JNC TJ8400 2001-3

オンライン
臨界安全監視システムの
開発研究(V)

(核燃料サイクル開発機構　委託研究成果報告書)

2001年2月

摂南大学　工学部

JNC TJ8400 2001-3
2001年2月

オンライン臨界安全監視システムの開発研究（V）

山田　澄*

要　旨

核燃料加工・再処理システムの臨界安全をより一層確実なものとするための未臨界度モニタリングシステムの必要性と重要性に鑑みて、本研究では、必要な場所に容易に設置できる安価で即応性の良い臨界安全監視システムを開発することを目的として、平成３年から、５年間の基礎研究と４年間の開発研究を行ってきた。本報告はこれら一連の研究を総括するものである。

核燃料加工施設で観測される中性子信号あるいはガンマ線モニタからの信号は基本的にランダム時系列であることに着目し、ランダム時系列解析手法の一つである逐次型自己回帰移動平均（ＡＲＭＡ）モデル同定手法をオンライン未臨界度推定に応用した安価で即応性の良い臨界安全監視システムを開発するための知見を蓄積することが本研究の目的である。

平成３年からの５年間にわたる基礎研究では、1）定常時系列にたいして、未臨界度を推定するのに最適な逐次型自己回帰移動平均（ＡＲＭＡ）モデル同定アルゴリズムの検討、2）未臨界度を精度よく推定するために必要となる信号前処理法の検討、3）時間的に変化する未臨界度を、できるだけ時間遅れを発生せずに追跡できる逐次型自己回帰移動平均（ＡＲＭＡ）モデル同定アルゴリズムの開発、4）これらの手法を臨界安全監視システムに応用するために必要となるシステム構成・要素モジュールの基本概念についての検討、5）核燃料サイクル開発機構のＤＣＡ実験データの解析等を行い、逐次型自己回帰移動平均（ＡＲＭＡ）モデル同定手法を用いた臨界安全監視システムの実用化に必要となる知見を蓄積した。

平成９年からの４年間にわたる開発研究では、1）それまでに蓄積した成果を実際の臨界安全監視システムとして構築するために必要となるモジュールに付いての検討、2）解析表示モジュールの中でＡＲＭＡモデル同定と平行して信号解析を行う冗長系として重要な要素となるディジタル適応フィルタモジュールの開発と実データ解析、3）ガンマ線信号を用いたFeynman-α法による未臨界度推定モジュールと、その信号取得に必要となる高速パルス信号処理系の提案、4）電源雑音を除去するノッチフィルタの設計法と実際のソフトウエアの開発、5）逐次型ＡＲＭＡモデル同定プログラムを始めとする一連のプログラムのワークステーションへの移植作業、等が行われた。

なお、本報告では内容の一貫性を保つため、未臨界原子炉のＡＲＭＡモデルと逐次型ＡＲＭＡモデル同定アルゴリズムを２，３章で述べると共に、ワークステーションに移植されたソフトウエア及び今年度再開発したディジタル適応フィルタモジュールのソースリストと使用法を付録として添附した。

※　本報告書は、摂南大学が核燃料サイクル開発機構の委託により実施した研究の成果である。
※　開発機構担当部課室：東海事業所　安全管理部　安全研究グループ

＊摂南大学　工学部　電気工学科

R & D Study on On-line Criticality Surveillance System (V)

Sumasu Yamada*

Abstract

In view of necessity and importance of criticality surveillance systems for ensuring the safety of nuclear fuel manufacturing and reprocessing plants, 5-year basic studies and 4-year R & D studies on an on-line criticality surveillance system were carried out since 1991. This report is a summary of these series of studies.

Noticing that the signal from a neutron detector is random in principle, these series of studies aimed to accumulate knowledge for developing an inexpensive criticality surveillance system with quick response based on the Auto-Regressive Moving Average (ARMA) model identification algorithm.

During five-year basic studies on criticality surveillance system since 1991, we obtained knowledge required for developing a criticality surveillance system based on the ARMA model identification algorithm through 1) studies on recursive ARMA model identification algorithms most appropriate for estimating subcriticality form time series data under a steady state condition, 2) studies on pre-processing of signal from neutron detectors, 3) developing a new recursive ARMA model identification algorithm with small time delay to estimate time-dependent subcriticality, 4) proposing a basic concept for the elements required for an on-line criticality surveillance system, and 5) numerical analysis of data from the DCA experiments.

During next four-year R & D studies on a criticality surveillance system since 1996, we 1) proposed modules required for a no-line criticality surveillance system, 2) revealed effectiveness of a adaptive digital filter (ADF) algorithm, as an important redundancy to the recursive ARMA model identification algorithm to be used in the signal processing module through numerical analysis of real data, 3) proposed a module of the Feynman-α method over γ ray signal and a fast signal processing module for γ ray signal, 4) developed a line-noise removal filter(Notch filter) and revealed its effectiveness for the DCA data corrupted with power-line noise, and 5) transplanted the recursive ARMA model identification program (ARMAX1) and other programs developed on the MS-DOS into those on the UNIX.

For self-consistency, this report is incorporated with the basic theory of ARMA model of a subcritical reactor and the ARMA model identification algorithm, as well as source programs and operation manuals of the NEOADF algorithm and other programs developed in these studies.

Work performed by Setsunan University under contract with Japan Nuclear Fuel Cycle Development Institute
JNC Liaison : Safety Study Group, Environmental and Safety Division, Tokai Work.

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

目　次

1 緒言

わが国における核燃料サイクルを確立するため、平成１７年の本格的作業開始を目指して使用済核燃料再処理工場の建設工事が現在進められている。このような核燃料再処理施設の安全性確保には安価で即応性の良い臨界安全監視システムの開発が以前より望まれていた。それにも関わらず、平成１１年９月３０日、不幸にも核燃料加工施設ＪＣＯにおいて臨界事故が発生し、一般社会に大きな不安を呼び起こしたことは記憶に新しく、益々臨界安全監視システムの必要性が高まっている。

本研究ではすでに平成３年から臨界安全監視の必要性と重要性に鑑みて、先ず５年間の基礎研究を行い、続いて更に４年間の開発研究を行ってきた。本報告はこれらの一連の研究を総括するものである。

核燃料再処理系の安全を確保する上で、沈澱、凝縮あるいは他の装置からの予想以上の核分裂性物質の流入等、再処理系において未臨界度変化の可能性がある所には、何らかの臨界安全監視システムを設けることが不可欠である。そのため、核燃料再処理系の未臨界度を精度良く推定して臨界安全性を確保しようとして、炉物理の分野で用いられている反応度測定法を用いる研究や、新たな手法の開発研究が進められている。これらの内、パルス中性子法、中性子相関法とも呼ばれるFeynman-α、Rossi-α法等は中性子検出からのパルス信号の統計的性質に着目した方法であり、Mihalczo法は中性子検出器信号の揺らぎのパワースペクトルに着目するいわゆる炉雑音解析手法に基づくものである。これらは全て原理的には一点炉近似動特性方程式を基礎とするもので、仮定の成り立つ範囲で反応度の絶対測定を行うことができ、反応度較正を必要としないと言う点で臨界安全監視法として望ましい特徴を有している。しかし、パルス中性子法はパルス中性子を発生する大型の装置を必要とする点で、臨界安全監視システムとして採用するには経済的問題があり、また、Mihalczo法では中性子源としてCf-252を内蔵する特殊な中性子検出器と通常の中性子検出器２つを用いなければならない。更に、Mihalczo法では、未臨界度を求めるのにこれらの検出器出力信号の揺らぎから計算される４種類のスペクトルを高速フー

リエ変換（Fast Fourier Transform, ＦＦＴ）を使って計算するため、結果を出すためには膨大なデータ処理が必要となる。即ち、Mihalczo法も計算機を含めた測定装置一式にかなりの費用を必要とする。従って、臨界事故発生の可能性がある所全てにこの様な高価な監視システムを設けることは燃料再処理のコストに大きな影響を及ぼす。その上、これらの方法はいずれもデータ処理がオフラインであるため、ある程度以上のデータが計算機内に蓄積されるまで未臨界度を推定することができず、刻々変化するシステムの状態監視に遅れを生じることになる。

使用済燃料貯蔵体系や燃料再処理システムの臨界安全性をより一層確実なものとするためには、必要に応じて容易に設置できる安価で即応性の良い未臨界度モニタリングシステムの開発が必要である。このような施設で観測される中性子信号あるいはガンマ線モニタからの信号がランダム時系列である。そこで、本研究ではランダム時系列解析手法の一つである逐次型-自己回帰移動平均（ＡＲＭＡ）モデル同定アルゴリズムをオンライン未臨界度推定に応用して、安価で即応性の良い臨界安全システムの開発できないかどうかを検討することからスタートした。一連の研究では、この手法の有効性が計算機によるシミュレーションデータ解析や、近畿大学原子炉の実データ解析、あるいは旧動燃動力炉核燃料開発事業団の重水臨界集合体を用いた実験データの解析などを通して、この手法を基本とする臨界安全監視システム構成の検討、そこに組み込まれるモジュールの検討、さらには臨界安全監視システムの高機能化なのための検討を行ってきた。

研究経過と成果を述べるためと、本報告書のまとまりと報告書としての一貫性を考慮し、２章には原子炉の離散時間モデルである自己回帰移動平均（ＡＲＭＡ）モデルの導出、３章では本研究で未臨界度推定に用いる逐次型ＡＲＭＡモデル同定アルゴリズムを述べた後で、４章で研究経過と成果を示す。

なお、本年度は昨年度に続いて、これまでにMS-DOS上で開発された一連のソフトウエアをワークステーション上に移植する作業を行っており、開発されたソフトウエアのソーステキストを添付する。また、適応ディジタルフィルタ（ＡＤＦ）に関するソフトウエアの検証並びに拡張を行っており、NEOADFプログラムのリストおよび、その使用マニュアルを付録として添付している。

2 未臨界度推定の原理

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。

$$\frac{d\delta N}{dt}=\frac{k(1-\gamma\beta)-1}{l}\delta N+\lambda\delta C+r_1 \qquad (2.1.a)$$

$$\frac{d\delta C}{dt}=\frac{k\gamma\beta}{l}\delta N-\lambda\delta C+r_2 \qquad (2.1.b)$$

ここで $r(t)=[r_1(t)\ \ r_2(t)]^T$ は平均値零の白色不規則力で、その分散は

$$\left\langle r_1(t)r_1(t)^T\right\rangle=D\delta(t-\tau)$$

で与えられる（＜・＞は期待値を表す）。ここで、Ｄは拡散定数行列で、拡張 Shcottky 公式

$$D=Q+FE+EF^T \qquad (2.2)$$

によって与えられる。ただし、

$$F=\begin{bmatrix}\dfrac{k(1-\gamma\beta)-1}{l} & \lambda\\ \dfrac{k\gamma\beta}{l} & -\lambda\end{bmatrix} \qquad (2.3)$$

$$E=diag[Sl/(1-k)\quad S\gamma\beta/(1-k)\lambda] \qquad (2.4)$$

$$Q=\frac{Sk(1-\gamma\beta)}{l<\nu_0>(1-k)}\begin{bmatrix}\langle\nu_0(\nu_0-1)\rangle & \langle\nu_0\nu_1\rangle\\ \langle\nu_0\nu_1\rangle & \langle\nu_1(\nu_1-1)\rangle\end{bmatrix} \qquad (2.5)$$

であり、k,β,λ,l,γ は慣用のものである。νはウラン２３５の核分裂により生じる中性子数であり、ν０＝ν（１－γβ）、ν１＝νγβ、Ｓは外部中性子源の強さ(n/s)を表す。

ここで、$x(t)=[\delta N(t)\quad \delta C(t)]^T$ とし、観測雑音として分散がＳの白色雑音 $s(t)$ を

仮定すると、式(2.1)は

$$\frac{dx(t)}{dt} = Fx(t) + r(t) \qquad (2.6.a)$$

$$y(t) = Hx(t) + s(t) \qquad (2.6.b)$$

と書ける。なお、$H=[1 \quad 0]$である。(2.6)をサンプリング時間Δtで離散時間化すると、

$$x(n+1) = \Phi x(n) + f(n+1) \qquad (2.7.a)$$

$$x(n+1) = \Phi x(n) + f(n+1) \qquad (2.7.b)$$

と書ける。ここで、

$$x(n) = x(n\Delta t) \qquad (2.8)$$

$$y(n) = y(n\Delta t) \qquad (2.9)$$

$$\Phi = \exp(F\Delta t) \qquad (2.10)$$

であり、また$f(n)$、$v(n)$は

$$f(n) = \int_{(n-1)\Delta t}^{n\Delta t} r(\tau)\exp\{F(n\Delta t - \tau)\}d\tau \qquad (2.11)$$

$$v(n) = \int_{(n-1)\Delta t}^{n\Delta t} s(\tau)\exp\{F(n\Delta t - \tau)\}d\tau \qquad (2.12)$$

により与えられるものである。なお、観測雑音の分散をdとすると、離散化された雑音源f(n)、V(n)の分散は

$$E[f(m)f(n)^T] = V\delta_{mn} \qquad (2.13)$$

$$E[v(m)v(n)^T] = R\delta_{mn} \qquad (2.14)$$

ただし、

$$V = \int_0^{\Delta t} \exp(F\tau)\cdot D\cdot\exp(F^T\tau)d\tau \qquad (2.15)$$

$$R = \int_0^{\Delta t} \exp(F\tau)\cdot S\cdot\exp(F^T\tau)d\tau \qquad (2.16)$$

によって与えられる。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

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

$$=\frac{\exp(\eta_+\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_- & \lambda\\ k\gamma\beta/l & -\lambda-\eta_-\end{bmatrix}-\frac{\exp(\eta_-\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_+ & \lambda\\ k\gamma\beta/l & -\lambda-\eta_+\end{bmatrix} \tag{2.17}$$

と書ける。ただし、$\alpha=[1-k(1-\gamma\beta)]/l$である。

同様にして、Vは

$$\begin{aligned}V=&\frac{\exp(2\eta_+\Delta t)-1}{2\eta_+(\eta_+-\eta_-)^2}(F-\eta_-I)D(F^T-\eta_-I)\\&-\frac{\exp((\eta_++\eta_-)\Delta t)-1}{(\eta_++\eta_-)(\eta_+-\eta_-)^2}(F-\eta_-I)D(F^T-\eta_+I)\\&-\frac{\exp((\eta_++\eta_-)\Delta t)-1}{(\eta_++\eta_-)(\eta_+-\eta_-)^2}(F-_+I)D(F^T-\eta_-I)\\&+\frac{\exp(2\eta_-\Delta t)-1}{2\eta_-(\eta_+-\eta_-)^2}(F-\eta_+I)D(F^T-\eta_+I)\end{aligned} \tag{2.18}$$

で与えられる。

(2.7.a)(2.7.b)に適当な物理定数と白色時系列 $f(t)=[f_1(t)\quad f_2(t)]^T]$、$v(n)$を用意することにより、未臨界原子炉のシミュレーションデータを作成することが出来る。

この様にして、物理モデルは Markov-Langevin 方程式(2.7.a)(2.7.b)で記述できたが、我々が直接観測出来るのは$\{y(n)\}$だけであって、$f(n)$、$v(n)$は観測できないためこのままではモデルパラメータΦを求めることは出来ない。

そこで、$\{y(n):k=\cdots,n-3,n-2,n-1,n\}$が与えられたときの$x(n)$の最適推定値$x(n|n)$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$\{y(n):k=\cdots,n-3,n-2,n-1,n\}$の張るデータ空間へ現在のデータ$y(n)$を射像することで、

$x(n|m)$　：m時点までの観測データがら得られるn時点での状態変数ベクトル$x(n)$の最適推定値

$y(n|n-1)$　：n-1時点までの観測データから得られるn時点での状態変数ベクトル$y(n)$の最適推定値とし、イノベーションν(n)を

$$v(n)=y(n)-y(n|n-1) \tag{2.19}$$

と定義すると、物理モデルは等価的に

$$x(n+1|n+1)=\Phi x(n|n)+Kv(n+1) \tag{2.20.a}$$

$$y(n)=Hx(n|n)+(I-HK)v(n) \tag{2.20.b}$$

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$[k_1\ k_2]^T$で、２×２行列Ｐに関するリカッチ（Riccatti）型方程式：

$$P=\Phi(P-KHP)\Phi^T+V \tag{2.21}$$

の一意解

$$P=\begin{bmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{bmatrix} \tag{2.22}$$

によって

$$K=PH^T\Gamma^{-1} \tag{2.23}$$

で与えられる。Γはイノベーションの分散で

$$P=HPH^T+R \tag{2.24}$$

で与えられる。

最後に、(2.17.a)(2.17.b)より $x(n+1|n+1)$を消去し、$v(n)$から$y(n)$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される原子炉のＡＲＭＡ（２，２）モデルは

$$G(z)=\frac{1+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \tag{2.25}$$

となる。このとき、ＡＲＭＡ（２，２）モデルの各パラメータは

$$a_1=-(\Phi_{11}+\Phi_{22}) \tag{2.26.a}$$

$$a_2=\Phi_{11}\Phi_{22}-\Phi_{12}\Phi_{21} \tag{2.26.b}$$

$$b_1=\{P_{12}\Phi_{12}-P_{11}\Phi_{22}-R(\Phi_{11}+\Phi_{22})\}/(P_{11}+R) \tag{2.26.c}$$

$$b_1=(P_{11}\Phi_{22}-P_{12}\Phi_{21})R/(P_{11}+R) \tag{2.27.d}$$

で与えられ、これらは全て１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等のみから算出できる。

なお、ここで注意しなければならないことは、式(2.25)のパラメータの内、分母の自己回帰パラメータａ１、ａ２は遷移行列Φの要素のみから成り立っており、原子炉の物理パラメータとサンプリング周波数から計算される定数であるが、分子の移動平均パラメータであるｂ１、ｂ２はリカッチ方程式の解にＰを含んであり、この値は実際には観測雑音の分散が与えられない限り計算できない。

従って、もし原子炉から観測される時系列をＡＲＭＡ（２，２）過程

$$y(n)=-a_1y(n-1)-a_2(n-2)+e(n)+b_1e(n-1)+b_2e(n-2)$$

としてモデル同定すれば、この自己回帰パラメータから原子炉の物理定数を逆算出来ることになる

3　未臨界度推定基本アルゴリズム

まず、基本的にはカルマンフィルタアルゴリズムと同等な逐次推定手法である Recursive Prediction Error Method（ＲＰＥ法）、Psuedo Linear Regression Method （ＰＬＲ法）、Extended Least Squares Method（ＥＬＳ法）の推定手順、及び計算プログラム上での処理の方法などについて説明する。

次に、ＲＰＥ法のもつ推定値の偏りを補正する際用いられる白色化フィルタの安定性を保証するための Projection Algorithm について計算プログラム上での実現方法も含めて説明する。最後に、非定常時系列データにおけるパラメータ推定の追従速度を速めるアルゴリズムとして開発された P-Matrix Activation Method （ＰＭＡ法）、及び P-Matrix Diagonal Activation Method （ＰＭＤＡ法）について説明する。

３．１　Recursive Prediction Error Method（ＲＰＥ法）

一入力・一出力システムの $u(t)$入力を、出力を $y(t)$とし、観測不可能な
外乱を $e(t)$とすると、

$$y_j + \sum_{k=1}^{p} a_{k,j} y_{j-k} = \sum_{k=0}^{r} b_{k,j} u_{j-k} + \sum_{k=1}^{q} c_{k,j} e_{j-k} \qquad (3.1)$$

をシステムのＡＲＭＡ（p,q,r）モデルと呼ぶ。

$u(t)$、$y(t)$とが観測可能で、$e(t)$が観測不可能な白色時系列外乱である場合、a_iを自己回帰（Auot-Regressive:AR）パラメータ、b_jを外部入力パラメータ（eXogeneous:X）、c_kを移動平均パラメータ（Moving Average:ＭＡ）と呼び、これらのパラメータを逐次的に求める一手法がＲＰＥ法である。入力が無い場合には、システムはいわゆる自己回帰移動平均（ＡＲＭＡ）モデルとなる。従って、本報告で用いるアル

ゴリズムは（3.1）式で$u(t)$を零とした場合に相当する。

時間推移演算子を用いると、式（3.1）は次のように書ける。

$$A(z)y(t)=B(z)u(t)+C(z)e(t) \qquad (3.2)$$

ただし、

$$A(z)=1+a_1z^{-1}+a_2z^{-2}+\cdots+a_pz^{-p}$$

$$B(z)=1+b_1z^{-1}+b_2z^{-2}+\cdots+b_rz^{-r}$$

$$C(z)=1+c_1z^{-1}+c_2z^{-2}+\cdots+c_qz^{-q}$$

ここで、$e(t)$が観測出来たとして、観測データベクトルを

$$\phi^T(t+1)=\begin{bmatrix}-y(t),-y(t-1),\cdots,-y(t-p+1),\\ u(t),u(t-1),\cdots,u(t-r+1),\\ e(t),e(t-1),\cdots,e(t-q+1)\end{bmatrix} \qquad (3.3)$$

と定義し、さらに、これから推定しようとするパラメータベクトルを

$$\theta^T=[a_1,a_2,\cdots,a_p,b_1,b_2,\cdots,b_r,c_1,c_2,\cdots,c_q] \qquad (3.4)$$

と定義すると、残差（式誤差）$v(t)$

$$v(t)=y(t)-\theta^T(t)\phi(t) \qquad (3.5)$$

が定義される。そこで、この$v(t)$2乗の平均値、即ち$v(t)$の分散

$$V_N(\theta)=\frac{1}{N}\sum_{t=1}^{N}[y(t)-\theta^T(t)\phi(t]^2 \qquad (3.6)$$

を最小にするような最適パラメータを逐次的に求める方法が逐次型ＡＲＭＡＸモデル同定アルゴリズムである。

このことは$V_N(\theta)$が最小、すなわち残差の分散を最小にするパラメータを推定することになる。従って、最適パラメータは

$$\frac{\partial}{\partial\theta}V_N(\theta)=0 \qquad (3.7)$$

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

$$\hat{\theta}(N)=\left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1}\sum_{t=1}^{N}\phi(t)y(t) \qquad (3.8)$$

と表される。ここで、逐次計算を行うために式(3.21)に於いて、N→tと置き換え、

$$\hat{\theta}(t)=\left[\sum_{k=1}^{t}\phi(k)\phi^T(k)\right]^{-1}\sum_{k=1}^{t}\phi(k)y(k) \qquad (3.9)$$

と書き、新たに(p+r+q)×(p+r+q)マトリックスである$R(t)$を

$$R(t)=\sum_{k=1}^{t}\phi(k)\phi^T(k) \qquad (3.10)$$

と定義し、更にこの逆行列の平均値行列Pを

$$P=\overline{R}^{-1}(t)=\frac{1}{t}R^{-1}(t) \qquad (3.11)$$

と定義すると、この行列はパラメータの誤差の共分散行列を表していることになることが、カルマンフィルタの理論から説明できる。

ここで、確立時系列データの平均相関行列$\overline{R}^{-1}(t)$は計算に取り込むデータ数が増加するに従って一定の値に収束する。従って、その逆行列Pも収束し非常に小さな値に落ち着く。この行列Pは推定パラメータの修正量を調整する係数であり、この行列の収束は推定パラメータの

収束を意味する。従って、一般的に、行列Pの収束と共にパラメータの修正量が減少し新たに読み込むデータに対する修正感度は減少し、パラメータの更新と言う観点からは、充分行列Pが収束した状態でシステムのパラメータ変化が発生した場合、パラメータの修正が少なく、結果的にパラメータの変化の追跡に大幅な遅れを生じることになる。

さて、これまでは説明を簡単にするために$e(t)$は観測できるものと仮定してきたが、実際には観測不可能なランダム変数であり、$e(t)$は計算過程で推定しながらパラメータ推定を行わなければならない。そのため、$e(t)$の推定値として先に定義した式誤差$v(t)$を用いる。この様にして、逐次型ＡＲＭＡモデル同定アルゴリズムを得る。

以下に逐次計算のアルゴリズムについて示す。
説明の過程で用いられる各シンボルは Table 3.1 に示されるものである。

ＲＰＥ法のアルゴリズム

観測される時系列データ$y(t)$、$u(t)$を新たに取得する度に、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。

$$\hat{y}(t)=\hat{\theta}^T(t-1)\phi(t) \qquad (3.12)$$

［ＳＴＥＰ２］
Prediction Error e_pの計算。

$$e_p=y(t)-\hat{y}(t) \qquad (3.13)$$

［ＳＴＥＰ３］
$\hat{\theta}(t)$の更新。

$$\hat{\theta}(t)=\hat{\theta}(t-1)+L(t-1)\cdot e_p \tag{3.14}$$

［ＳＴＥＰ４］

式誤差（残差）$v(t)$の推定$\hat{v}(t)$値計算。

$$\hat{v}(t)=y(t)-\hat{\theta}^T(t)\phi(t) \tag{3.15}$$

［ＳＴＥＰ５］

式誤差の非白色性による推定値の偏りを補正。$\frac{1}{\hat{C}(z)}$による白色化

$$\tilde{y}(t)=y(t)-\sum_{i=1}^{p}K_t\hat{c}_i(t)\tilde{y}(t-i) \tag{3.16}$$

$$\tilde{u}(t)=u(t)-\sum_{i=1}^{r}K_t\hat{c}_i(t)\tilde{u}(t-i) \tag{3.17}$$

$$\tilde{v}(t)=v(t)-\sum_{i=1}^{q}K_t\hat{c}_i(t)\tilde{v}(t-i) \tag{3.18}$$

［ＳＴＥＰ６］

$\phi(t)$及び$\varphi(t)$の更新。

$$\phi^T(t+1)=\begin{bmatrix}-y(t),-y(t-1),\cdots,-y(t-p+1),\\ u(t),u(t-1),\cdots,u(t-r+1),\\ e(t),e(t-1),\cdots,e(t-q+1)\end{bmatrix} \tag{3.19}$$

$$\varphi^T(t+1)=\begin{bmatrix}-\tilde{y}(t),-\tilde{y}(t-1),\cdots,-\tilde{y}(t-p+1),\\ \tilde{u}(t),\tilde{u}(t-1),\cdots,\tilde{u}(t-r+1),\\ \tilde{v}(t),\tilde{v}(t-1),\cdots,\tilde{v}(t-q+1)\end{bmatrix} \tag{3.20}$$

以降$\varphi(t)$はこの更新されたものである。

Table 3.1 Notations used in RPE method.

Symbo l	Discription	Form
$\hat{\theta}(t)$	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
$\varphi(t)$	Revised time series vector	(p+r+q) Row-vector
$e_p(t)$	Prediction Error	variable
$\lambda(t)$	Forgetting Factor	variable
K_t	Contraction Factor	variable
$\tilde{y}(t)$	Revised value of $y(t)$	variable
$\tilde{u}(t)$	Revised value of $u(t)$	variable
$\tilde{v}(t)$	Revised value of $v(t)$	variable

［ＳＴＥＰ７］

マトリックス $P(t)$ の更新。

$$P(t)=\frac{P(t-1)}{\lambda(t)}-\frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda(t)+\varphi^T(t)P(t-1)\varphi(t)}\frac{1}{\lambda(t)} \qquad (3.21)$$

［ＳＴＥＰ８］

ゲインベクトル $L(t)$ の更新。

$$L(t)=P(t)\varphi(t) \qquad (3.22)$$

［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、以下に示すような二つのパラメータを導入しているから原子炉の物理定数を逆算出来ることになる。

① 忘却係数：(Forgetting Factor)

忘却係数λは［ＳＴＥＰ７］のＰマトリックスの更新時に作用するパラメータで、これはＰマトリクスの更新時に一段階前の推定値が次回の推定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の値は先にも説明したとおり、使用するデータが増加するに逆比例して小さくなる性質を持っており、パラメータ修正ゲインＬ(t)は［ＳＴＥＰ８］の式(3.22)で与えられるようにＰに比例している。従って、λ＝１の時はＰマトリックスの更新に対して影響を与えることはない。また、０＜λ＜１の時はＰマトリックスの修正量を大きくして新規の情報を重んじるよう作用する。この場合、λ＝１の時と比べて等価的にＰ行列の値の減少が少なくなり、推定パラメータの更新量を大きくする。

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より小さい値にすることによってＡＲＭＡパラメータの変化量を大きくすることにより速く真値に近づける。そしてＰマトリックスが収束した後は、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を同定するときには、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、$\lambda(t)$は次式

$$\lambda(t)=(\lambda_0-1)\mu_\lambda^{t-1}+1 \qquad (3.23)$$

のように定義している。

②　縮約係数：(Reflection Coefficient)

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ＭＡモデルで表し、その逆フィルタを白色化フィルタ

として時系列データを補正する過程が[ＳＴＥＰ５]である。しかし、この過程では、データ量が少ないときや、システムが過渡変化時では$e(t)$の白色化フィルタを構成する推定パラメータである$\hat{C}(z)$自身の精度が悪くなり、このパラメータを用いて白色化フィルタを構成すると、結果的にパラメータの収束を損なう場合を生じる。従って、推定パラメータが大きく変動するような同定初期段階などでは、白色化を強く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補正段階で縮約係数K_tを作用させ、同定段階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予め、K_tの初期値を$K_0=0$とし、更新のためのファクターとして
μ_k $(0\leq\mu_k\leq1)$を与え、K_tを

$$K_t=1-\mu_k^t \qquad (3.24)$$

のように定義した。

なお、K_t＝１の時は、完全な白色化を行い、K_t＝０の場合には白色化を行わない。また、K_t＝１の時パラメータ推定の誤差評価関数は近似的に　度関数となり、K_t＝０の場合には誤差評価関数は最小２乗関数となる。ＲＰＥ法において、K_t＝０とするとＰＬＲ法、K_t＝０、λ(n)＝１とするとＥＬＳ法、K_t＝１、λ(n)＝１とすればＲＭＬ法となる。

ベクトル及びマトリクスの初期値

本研究で用いた計算プログラムでは計算時間の短縮化とメモリの節約を実現するために、基本的にはＰマトリックスの構成をＵＤＬ分解マトリックスを用いて表現しており、実際の計算プログラムでは［ＳＴＥＰ７］及び［ＳＴＥＰ８］での計算方法が、式(3.12)、(3.13)とは異なる。ただし、内容は全く同じである。また、本章第５節で詳しく述べるＰＭＡ法の一種であるＰＭＤＡ法を用いるときには、ＵＤＬ分解マトリックスは用いずにＰマトリックス自身を用いている。その対応関係を Table 3.2 に示す。

最後に、上記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について述べる。特に、パラメータベクトル$\hat{\theta}(0)$に関しては、先験的な知識がありその近似値が既知である場合には、以後の同定を容易にするような値をセットすることにより速くて正確な同定を行うことが可能である。しかし、本研究では一般的な初期設定を用いた。その内容は以下の通りである。

Table 3.2 Formation of P-Matrix

P-Matirx Activation	method of P-matrix computation
No use	UDL decomposition of P-matrix
PMA method	UDL decomposition of P-matrix
PMDA method	Direct P matrix calculation

$$\theta^T(0)=[0,0,\cdots,0],\quad L^T(0)=[0,0,\cdots,0],\quad P(0)=\alpha I$$

（αは任意の正数、I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。本研究では、特殊な要因がない限りα＝１０００を用いている。

３．２ Pseudo Linear Regression Method（ＰＬＲ法）

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$\varphi(t)=\phi(t)$、となるものである。この場合、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には推定値の真値への収束は悪くなり推定値に偏りを生じてしまう。その一方で、推定ＭＡパラメータ$\hat{C}(z)$の精度及び安定性に依存しないためＭＡパラメータが安定しない非定常状態の時系列データを同定する場合などには有効な手段である。

3.3 Extended Least Squares Method（ＥＬＳ法）

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が常に１としたものである。従ってＰＬＲ法の特殊なタイプであると言え、ＲＰＥ法の中では最もシンプルな同定法であり計算量も最も少なくてすむ反面、同定対象依存性が強いなどの欠点をもつことが報告されている。

3.4 ＲＰＥ法におけるProjection Algorithm（ＲＭＬ法）

ＲＰＥ法のアルゴリズムにおいて縮約係数K_tを１、忘却係数λを１としたものである。ＲＰＥ法のアルゴリズムにおいて本章第１節でも述べたが、［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とが、白色化フィルタを形成するための推定パラメータ$\hat{C}(z)$が発散フィルタを形成しかねない。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$\hat{v}(t)$の白色化を行う場合、白色化フィルタを形成するための推定パラメータ$\hat{C}(z)$が安定、すなわち、

$$\hat{C}(z)=0 \qquad (3.25)$$

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ルーチンが必要となる。本研究で用いた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実現している。

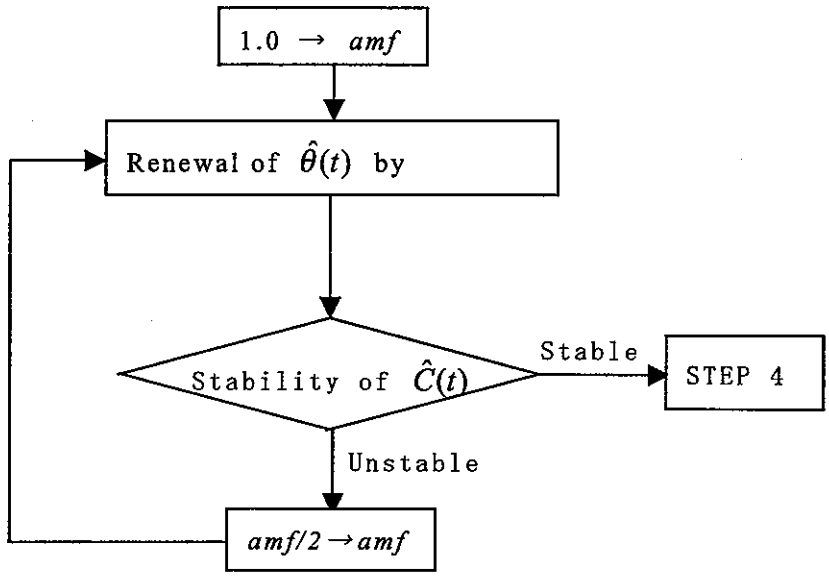
［ＳＴＥＰ３’］

$$\hat{\theta}(t)=\hat{\theta}(t-1)+L(t-1)\cdot e_p\cdot amf \qquad (3.26)$$

ここでは調整係数(AdjustMent Factor)として前頁にのようなフローチャートに従って変更される。　本研究で用いた計算プログラムでは、$\hat{C}(z)$の安定性判別に Suhur Cohn の判定条件を用いている。

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。



3．5　P-Matrix Activation Method（ＰＭＡ法）
& P-Matrix Diagonal Activation Method（ＰＭＤＡ法）

システムのパラメータが変動する非定常時系列データに対するＲＰＥ法の追従性能の改良に従来から取り組んできた。その手法としてＲＰＥアルゴリズムにおけるＰマトリックスの活性化法を提案してきた。ここでは、Ｐマトリックスの活性化の方法として、従来型のＰＭＡ法と本研究で新たに提案するＰＭＤＡ法について述べる。

まず、Ｐマトリックスの意味について説明する。尚、ここで用いる記号については Table 3.1 に示されているもの、及び３．１節のＲＰＥ法のアルゴリズムの説明で用いたものと同じである。

定常時系列のパラメータ推定の原理は、

$$y(t)=\theta^T\phi(t)+v(t)$$

が成立するという仮定の下で、残差$v(t)$の分散

$$V_N(\theta)=\frac{1}{N}\sum_{t=1}^{N}[y(t)-\theta^T(t)\phi(t]^2 \qquad (3.27)$$

を最小にするような最適パラメータを逐次的に求める方法が逐次型ＡＲＭＡＸモデル同定アルゴリズムである。
このことは$V_N(\theta)$が最小、すなわち残差の分散を最小にするパラメータを推定することになる。従って、最適パラメータは

$$\frac{\partial}{\partial\theta}V_N(\theta)=0 \qquad (3.28)$$

となるように選べばよい。この様なθを$\hat{\theta}(N)$は、

$$\hat{\theta}(N)=\left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1}\sum_{t=1}^{N}\phi(t)y(t) \qquad (3.29)$$

と表される。ここで、逐次計算を行うために式(3.29)に於いて、Ｎ→ｔと置き換え、

$$\hat{\theta}(t)=\left[\sum_{k=1}^{t}\phi(k)\phi^T(k)\right]^{-1}\sum_{k=1}^{t}\phi(k)y(k) \qquad (3.30)$$

と書き、新たに(p+r+q)×(p+r+q)マトリックスである$R(t)$を

$$R(t)=\sum_{k=1}^{t}\phi(k)\phi^T(k) \qquad (3.31)$$

と定義し、更にこの逆行列の平均値行列Pは

$$P=\overline{R}^{-1}(t)=\frac{1}{t}R^{-1}(t) \qquad (3.32)$$

で与えられた。

実際には、Ｐマトリックスは前述のＲＰＥ法のアルゴリズムの［Ｓ

ＴＥＰ］に従って計算されるものであるが、その物理的な意味は上述したものである。Ｐマトリックスは確率時系列データの相関行列であるマトリックス$\overline{R}(t)$の逆行列であるから、推定が進むにつれて$\overline{R}(t)$マトリックスはある一定値に収束し、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。式（3.14)、(3.22)から分かるように、Ｐマトリックスは推定パラメータ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。統計的に、Ｐマトリックスが収束すると新規の情報に対して感度が鈍くなる。従ってＰマトリックスが収束した場合、パラメータが変動する非定常時系列データにおいてそのパラメータの変化の追跡に大幅な遅れを生じる。

そこで、外部からＰマトリックスの収束を抑制するのが活性化法である。

ＰＭＡ法は、基本アルゴリズムとしてＲＰＥ法によりパラメータ推定を行いながら、一定間隔でＰマトリックスの全要素に対して定数を掛けることによりパラメータ変更感度を活性化する手法である。

ＰＭＤＡ法はＰＭＡ法の特殊なタイプである。ＰＭＡ法がＰマトリックスの全要素に定数を掛けるのに対して、ＰＭＤＡ法ではＰマトリックスの主対角要素に対してのみ一定間隔で定数を掛けるものである。この方法を用いることによって同一活性化量でより高い感度が得られる。ただし、ＰＭＤＡ法はＰマトリックスの構成においてメモリの節約と高速計算を実現するＵＤＬ分解マトリックス表現が不可能なためＰＭＡ法に比べてメモリ及び計算時間を多く必要とする。

Ｐマトリックスの活性化は、本章第１節のＲＰＥ法のアルゴリズムの説明での［ＳＴＥＰ９］での周辺処理の一環として以下の手続きに従って行う。ここで、A_{act}は活性化量を表し、$A_{act}>1$である。

ＰＭＡ法

$$P(t)=\begin{cases} A_{act}\cdot P(t) & : t=nT_{act} \\ P(t) & : otherwise \end{cases}$$

ここで、T_{act}は活性化間隔を表し、nは正の整数である。

ＰＭＤＡ法

$$diag[P(t)]=\begin{cases} A_{act}\cdot diag[P(t)] & : t=nT_{act} \\ diag[P(t)] & : otherwise \end{cases}$$

Ｐ行列活性化法を採用することで、逐次計算におけるパラメータ修正量を強制的に大きくすることにより、システムの過渡変化時におけるパラメータ推定の追従性能は高めようとするものである。しかし、本来Ｐマトリックスは、式(3.27)における評価関数を最小にするという条件で更新されるもので、活性化という外部からのファクターで乱すことになり、定常状態における推定パラメータの統計的変動は大きくなるという欠点がある。従って、Ｐ行列活性化法を導入する場合は、「適正な追従速度の要求」・「定常時におけるパラメータ変動の許容量」を総合的に考慮した上で、活性化量及び活性化間隔を設定する必要がある。この様な考え方に近い方法として、Ｐ行列が０に収束するのを防ぐためにＰ行列に予め定めてルールに従う時間依存の定数を加える方法も提案されているが、加える定数の算定法が明らかでなく、これも試行錯誤的に定めなくてはならない。

4 研究経過と成果

4.1 平成3年度(1991年度)

使用済燃料貯蔵体系や燃料再処理システムの臨界安全性をより一層確実なものとするためには、必要に応じて容易に設置できる安価で即応性の良い未臨界度モニタリングシステムの開発が必要であるとの観点から、本研究では安価で即応性の良い臨界安全システムの開発を行うことを最終目的として、時系列解析におけるシステム同定手法の一つである逐次型-自己回帰移動平均(ARMA)モデル同定アルゴリズムをオンライン未臨界度推定に応用することを提案し、計算機によるシミュレーションデータと近畿大学原子炉からの実データを用いてオンライン未臨界度推定のための基本アルゴリズムの検討を行い、以下の結果を得た。

① 定常状態にある未臨界原子炉から得られる中性子信号揺らぎの離散時間データを計算機シミュレーションにより作成し、逐次型自己回帰移動平均ARMAモデル同定アルゴリズムを用いてこの時系列をARMAモデル同定した。このとき推定されたARMAパラメターは真のモデルのARMAパラメターに収束しないが、同定されたARMAモデルの伝達関数の高周波側折れ点周波数に対応する極から理論通り未臨界度を推定することができることが確認された。即ち、時系列データのパワースペクトル密度(PSD)の高周波側折れ点周波数に対応する極はARMAモデルの極に収束する。

② 定常状態でARMA(1、1)モデル同定を行う場合、ELS法を用いる方が最良い。このアルゴリズムは計算量が最も少なく、計算時間が短くて済む。未臨界度の推定時間を短縮するため、忘却係数が初期値から急速に1に近づくように設定された忘却作用の強い忘却係数を使う($\lambda(0)=0.95$、$\lambda_0=0.99$) PLR法を用いてもよいが、この場合はARMA(2、2)モデル同定を行う方がよい。

③ データ量が少ない場合はARMA(2、2)モデルのPLR法を用いるのがよい。ARMA(1、1)モデル同定に比べて計算量は増えるが、高速にパラメターを推定することができる。忘却係数は忘却作用の弱いλ

(0)=0。99、λ_0=0。99 を用いる必要がある。

④ 白色雑音を多量に含む状態では、ＡＲＭＡ（３、３）によるＲＰＥ法を用いるのがよい。ただし、このアルゴリズムは推定に必要なデータ数が多く、また推定精度もよくない。このアルゴリズムは尤度関数を評価関数とするのでＰＬＲ法に比べて計算量はかなり多く、モデル次数が高いため計算量はＡＲＭＡ（１、１）モデルの９倍、ＡＲＭＡ（２、２）モデルの２倍となる。白色雑音が多い場合には忘却係数はゆっくり１に近づく忘却作用の弱いもの（λ(0)=0。95、λ_0=0。999）を用いる。

⑤ 白色雑音がない場合、未臨界度を１０％以内のエラーで推定するのに必要なデータ数は１４００から３１００データ程度である。しかし、白色雑音が混入している場合はこれより増える。

⑥ 未臨界度の初期値設定をする事によってＡＲＭＡモデル同定の初期に発生する異常なパラメータ変動を避けることができる。しかし、初期値を設定すると、設定された未臨界度と異なる未臨界度の推定は初期値を設定しない場合よりも推定速度は遅くなることが多い。

⑦ 中性子信号揺らぎを離散化するためのサンプリング周波数は、ＡＲＭＡモデル伝達関数の高周波側折れ点周波数（未臨界度と１対１に対応する極）のおよそ１０倍から１００倍の範囲であれば、本手法が適用できる。

⑧ 近畿大学原子炉シミュレーションもでるから作成されたシミュレーションデータ解析では、０。０１Ｈｚにカットオフ周波数をもつ１次の低周波カットフィルターを通した場合、最も未臨界度推定速度が速い。

⑨ 未臨界度が７￠～７０￠の近畿大学原子炉から取得した実際の時系列データを用いた場合、アルゴリズムに適当な初期値設定を行うことにより、３万データの使用により１４￠の分解能で未臨界度の推定ができた。

⑩ 数Ｈｚから十数Ｈｚに伝達関数の高周波側折れ点をもつ近畿大学原子炉では、時系列の平均値を除去するため、０。１Ｈｚにカットオフ周波数を持つ低周波カットフィルターに時系列データを通した後、原子炉モデルとしてＡＲＭＡ（２、２）モデルを用いて未臨界度推定を行うのがよい。ＡＲＭＡ（１、１）モデルでは未臨界度の推定値に偏りが生じ、推定された未臨界度が実際の未臨界度より深く推定される。

以上の結果を総合すると、逐次型自己回帰移動平均ＡＲＭＡモデル同定アルゴ

リズムアルゴリズムを用いた未臨界度推法では、
（１）解析に用いる逐次型ＡＲＭＡモデル同定アルゴリズム、
（２）原子炉のＡＲＭＡモデルの次数、
（３）推定値に対する初期値設定、
（４）Ａ／Ｄ変換のサンプリング周波数、
（６）低周波除去フィルター、
（７）信号に混入している観測雑音の割合、
等、非常に多くの要因が複雑に絡み合っており、最適なＡＲＭＡモデル同定アルゴリズムの決定が簡単ではないことを示している。

ここでは触れていないが推定未臨界度の収束の速さは推定を開始する初期データにも依存することが観測されており、今後この問題も含めて上記の要因に付いて更に詳しい検討が必要である。

４．２　平成４年度（１９９２年度）

近畿大学原子炉の定数を用いて作成されたシミュレーションデータのＡＲＭＡモデル同定に用いたアルゴリズムを、そのまま近畿大学原子炉から実際に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、測定系に未知の外乱が加わる上、検出器からの信号は、増幅器、前処理用フィルター、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、信号の質がシミュレーションデータとは本質的に異なってくるため考えられる。

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その信号が持つ特性を表現しようとするため、色々な場面で混入する雑音によって生じ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しているものと考えられる。したがって、実際に原子炉から観測される信号から未臨界度を推定するためには、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、９２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、主にＲＰＥ法を用いてＡＲＭＡモデル同定し、以下の結論を得た。

4.2.1 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、その観測信号の平均値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、未臨界度は実際よりも浅く推定される。

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される未臨界度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未臨界度の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、未臨界度推定精度は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。

4.2.2 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。

4.2.3 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。

4.2.4 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。

4.2.5 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。

4.2.6 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。

4.3 平成5年度（1994年度）

オンラインARMAモデル同定アルゴリズムを用いて非定常状態での未臨界度推定を行うための知見を得るため、計算機シミュレーションにより最適未臨界度推定アルゴリズムの検討を行うと共に、未臨界状態で運転中の近畿大学原子炉に挿入されたCIC中性子検出器からの出力揺らぎも併せて解析し以下の結論をえた。

前年度までの研究により得られた結果をそのまま非定常状態に適用しただけでは、パラメータの変化を追跡する能力の点で十分ではない。これは、用いているアルゴリズムが本質的には非定常状態に対応できるはずであるが、実際には定常時系列に対して有効なアルゴリズムとなっているため、一定時間パラメータ推定を行った後ではパラメータ修正係数が非常に小さい値に収束しているため、それまでの情報を基にして作成されたモデルによって推定される予測値と実際の観測値との誤差が相当大きくならないと、パラメータの修正に効果が現れない。従って、非定常状態が予め予測される場合には、推定パラメータの修正係数が常にある値以上となるように工夫して、システムの変化を素早くパラメータの修正に反映出来るようなアルゴリズムが要求される。即ち、非定常状態のパラメータ推定に適したアルゴリズムに改める改善が必要であることが分かった。

そこで、我々は非定常時系列に対応できアルゴリズムとしてP－行列活性化法を提案し、この方法は未臨界度が変化する非定常状態でのパラメータ推定に

有効であることを示した。しかし、Ｐ－行列活性化法を用いても未臨界度が浅い状態から深い状態に変化する場合、その逆の場合に比べてパラメータ追跡能力の改善が顕著ではなかった。

以下に、実データ解析により得られた結論をまとめて述べる。

4.3.1 計算機シミュレーションによる解析結果

① 変動する未臨界度を追跡するアルゴリズムとしてＰ－行列活性化（P-matrix Activation：ＰＭＡ）法が有効である。

② シミュレーションでは、未臨界度が浅い方向に変化する場合は尤度関数を評価関数とするＲＭＬ法が、深い方向に変化する場合は最小２乗関数を評価関数とするＥＬＳ法が優れており、未臨界度の変化を総合的に追跡するためにはＥＬＳ法が優れているといえる。

③ 同定次数の問題については、ＡＲＭＡ（１、１）モデルを用いた場合、推定値の統計的変動が少ないが、この場合、速やかに変動を検出する能力にやや欠ける。

④ ＡＲＭＡ（２、２）モデルによる推定では比較的未臨界度の変動の検出能力がある。また、定常状態の未臨界度の推定では、未臨界度を浅い方向に過大評価することはなく、正常な未臨界度推定能力を持つ。しかし、ＡＲＭＡ（１、１）モデル同定に比べ推定値のばらつきが大きく、また、時系列データの変化が急激な場合には、一時的に誤った未臨界度を推定する場合がある。

⑤ ＡＲＭＡ（３、３）モデルによる未臨界度推定では、ＡＲＭＡ（２、２）モデル同定と同じ程度の未臨界度変化追跡能力を持つが、推定値のばらつきがＡＲＭＡ（２、２）モデルによる場合よりさらに大きくなり、また、浅い未臨界度から深い未臨界度に急激に変化す場合には未臨界度推定が出来なくなることはある。

⑥ 以上の結果を総合すると、特に浅い方向への未臨界度変化を高速に検出する必要のある臨界安全監視システムに対しては、パラメータ推定のモデルとしてはＡＲＭＡ（２、２）モデルを用い、活性化間隔１００データ、活性化量１。０１としてＰ－行列で活性化法をＥＬＳ法に用いることが最適である。

⑦ 推定値のばらつきが小さく安定した未臨界度推定を行いたい場合には

上記のＰ－行列活性化法を用いたＡＲＭＡ（１、１）モデル同定がよい。この場合、５０００から１万データの追跡遅れが発生する事に留意する必要がある。この程度の追跡遅れを認めるならＡＲＭＡ（１、１）モデル同定に対し、活性化間隔１００データ、活性化量１。０１で活性化したＥＬＳ法を用いれば計算量も少なく、従って最近のパーソナル計算機を用いて、安定した未臨界度の推定ができるものと期待される。

4.3.2 近畿大学原子炉による実データ解析による結果

⑧ 未臨界度が変化する非定常状態において、未臨界度の変化に対する推定値の追従性能の向上には、高周波サンプリングが有効であることが確認された。

⑨ サンプリング周波数を高くすることは、未臨界度が浅い方から深い方へと変化するときに推定値が追従しきれずに浅い方に偏りを生じるUnder Estimation 現象を抑制する効果があることが確認された。

⑩ 非定常状態の未臨界度推定にでは、未臨界度の変化の方向によって最適なＡＲＭＡモデル同定アルゴリズムが異なることが実データの場合にも生じることが確認された。未臨界度が深い方から浅い方へ変化に対して最適なアルゴリズムであっても、未臨界度が浅い方から深い方へ変化する場合には必ずしも最適ではない。同様のことがその逆の場合にも生じる。従って、あらゆる状況に対して最大公約数的に対処できるアルゴリズムを選択する必要があるという結論を得た。

⑪ 基本アルゴリズムとしてはＥＬＳ法を選択し、本研究で新たに用いたＰ－行列対角要素活性化法（ＰＭＤＡ法）を採用すれば、上記⑩での要請を満足でき、Over Estimation 現象、Under Estination 現象を抑制することができた。

4.3.3 今後の検討課題

最期に、本研究の結論から導かれる今後の課題と展望について述べる。

これまで、逐次型自己回帰移動平均ＡＲＭＡモデル同定アルゴリズムに関して、理論的研究(アルゴリズムも含む)とシミュレーションが中心のであった。しかし、提案せれているアルゴリズムを実際のシステム同定の場に持ち込むと、シミュレーション時系列データを解析した場合には遭遇しない様な問題が発生する場合があり、また、その問題がアルゴリズムの根幹に関わる問題であることが有り得る。

１．　信号処理と実験データ

近年、信号処理の分野では、急速な勢いでアナログからデジタルへの世代交代が進められており、その周辺技術も充実してきている。本研究にとって信号処理上最も大きな問題はアナログ信号処理の部分である。その意味では、ディジタル信号処理の技術の向上は歓迎すべき傾向であるといえる。特に、最近の動向で最も注目すべきことは、アナログ信号のデジタル化にエリアシング防止用のフィルタを用いずにオーバーサンプリングによって、かなり高い周波数領域にわたって信号を取り込み、その後にデジタルフィルタによって必要な波形整形をするものである。デジタルフィルタはアナログフィルタに比べて、経年変化や温度変化による特性変化がなく、しかも精度の高い急峻なフィルタ特性を有するものが得られることから、この方法を用いることによりナイキスト周波数近傍で歪の少ない時系列データを得ることが期待できる。

これらの状況を考慮して、今後、信号処理技術の高度化に対応し、より実プラントに近い未臨界システム、すなわち、高未臨界体系での実験研究を重ねる必要があると思われる。

２．　アルゴリズム

アルゴリズムに関しては、本研究では主にＡＲＭＡ（２、２）モデルを中心に研究を進めてきたが、依然としてシステム同定に用いるモデル次数の選択はかなり重要な問題として残っている。

ＡＲＭＡ（１、１）モデルによる未臨界度推定では、比較的揺らぎの少ない未臨界度推定が行えることが分かったが、今回用いたアルゴリズムとの組み合わせでは必ずしも総合的に見て最も優れた未臨界度推定法と言うことが出来ない点、信号処理も含めて今後も更に検討を続ける

べきテーマの一つである。また、今回解析に用いなかったアルゴリズムの組み合わせ等まだまだ多くの問題が残されている。

3．　未臨界度が深くなる場合の未臨界度変化の追跡

未臨界度が深くなる場合は、未臨界度が浅くなるときほど高速には未臨界度の変化追跡できないのはなぜかについては今回は全く触れていない。今後、逐次計算型ＡＲＭＡモデル同定アルゴリズムを未臨界度モニタリングシステムとして使用するためには、なにがこの様な問題を引き起こす原因かを詳しく検討する必要がある。

4．　パラメータの収束速度

Ｐマトリックスの初期設定において、一般的には推定パラメータの偏りを少なくするためにはαとして十分に大きな値を設定することが望ましいとされているが、我々の研究では、時系列によっては必ずしもそうではない場合もある。また、解析を始める最初のデータによって、パラメータの収束速度に著しい差を生じることも我々の研究で明らかになっており、何時でもどの様なデータから解析を始めてもパラメータ収束速度が変わらないようなアルゴリズムの研究も今後の課題であろう。

4．4　平成6年度（1995年度）

オンラインＡＲＭＡモデル同定アルゴリズムを用いて未臨界度が浅い状態から深い状態に変化する場合の非定常状態での未臨界度推定を最適に行うアルゴリズムに関する知見を得るため、前年度の解析で得た知見を基にして、アルゴリズムとして最も一般的なＲＰＥ法に対しパラメータサーベイを前年度と同じデータに対して行うと共に、旧動燃事業団のＤＣＡ実験から得られた中性子検出器信号出力揺らぎも併せて解析し、以下の結論をえた。

1．　一般的な逐次型ＡＲＭＡモデル同定アルゴリズムでは、パラメータの変化を追跡する能力の点で十分ではない。これは、用いているアルゴリズムが本質的には定常状態にある時系列の解析に適したものであり、定常時系列に対して有効なアルゴリズムとなっている。すなわち、パラメータ推定

を開始してから一定時間定常状態が続いた状態では、パラメータ修正係数が非常に小さい値に収束し、パラメータのばらつきが押さえられる。従って、パラメータ推定を開始してから一定時間定常状態が続いた後でシステムパラメータの変化が生じた場合、パラメータ修正係数が非常に小さい値であるため、それまでの定常状態を表すモデルが推定する予測値と実際の観測データとの誤差が相当大きくならないとパラメータの修正に効果が現れない。これが、パラメータの変動の追跡能力を低下させる原因である。そこで、状態変化が予め予測されるような場合にたいして、パラメータ修正係数を常にある値以上となるよう工夫し、システムの状態変化を素早くモデルパラメータの修正に反映出来るようなアルゴリズムが要求される。しかし、修正係数をある一定値に保つと定常な状態にも関わらず推定パラメータの変動が大きくなり安定したパラメータ推定ができなくなる。

そこで、昨年度は非定常時系列に対応できアルゴリズムとしてパラメータの修正ゲインを支配するパラメータの分散行列（Ｐ行）に一定間隔で定数を掛けてＰ行列を列活性化する方法（Ｐ行列活性化法）を提案し、この方法が非定常状態でのパラメータ推定に有効であることを示した。

しかし、未臨界度が浅い状態から深い状態に変化する場合、Ｐ－行列活性化法を用いても、その逆の場合に比べてパラメータ追跡能力の改善が顕著ではなかった。この問題に対して、今年度は、Ｐ行列の対角要素のみを活性化する手法を提案し、昨年度と同じデータに対してこの手法を用いた結果、昨年度見られたような問題が生じないことを示すことができた。すなわち、逐次型ＡＲＭＡモデル同定アルゴリズムとして最も一般的なＲＰＥ法に関しては、アルゴリズムの中で用いるパラメータを適切に選び、併せてＰ行列対角要素活性化法を用いれば、未臨界度が浅い状態から深い状態に変化する場合に対しても変動する未臨界度の推定が有効に行えることが確認された。また、ＲＰＥ法の特殊な場合であるＥＬＳ法とＰ行列対角要素活性化法の併用も非常に有効なアルゴリズムであることが確認できた。これらの結果と、昨年度の成果を総合すると、非定常状態の未臨界度推定においは、基本アルゴリズムとしてはＥＬＳ法を選択し、本研究で新たに用いたＰ－行列対角要素活性化法（ＰＭＤＡ法）を採用すれば、一般的な非定常状態に於いて、Over Estimation 現象、Under Estination 現象を起こすことなく適切な未臨界度推定が行えることが分かった。

２．　本年度、動燃事業団・大洗工学センターのＤＣＡ実験装置を用いて行われた未臨界定常運転時の中性子信号揺らぎデータを解析した結果、未臨界度が０～＄１１。２に対し、周波数スペクトル折れ点周波数は数Ｈｚから数十Ｈｚの間に分布しており、中性子信号揺らぎをデータレコーダで記録する場合、遮断周波数が５００Ｈｚ程度の比較的シャープな特性を持つローパスフィルターを用いればよい。この様なローパスフィルタを用いることにより、ＤＡＴデータレコーダで記録された信号は、計算機を通してエリアシングを生じることなく４８分１分周サンプリングできるため、比較的長時間の連続データを収集することが可能となるばかりでなく、その後の信号処理も非常に簡単になる。

今後の検討課題

最後に、本研究の結論から導かれる今後の課題と展望について述べる。これまで、本研究では主に近畿大学原子炉から得られた実験データを中心に解析を行ってきたため、比較的臨界に近い未臨界状態の未臨界度推定しか行っていない。このことは、本研究で検討してきたアルゴリズムが比較的Ｓ／Ｎ比の高い時系列に対して有効であることを示したもので、実用化を考えると今後は比較的深い未臨界度あるいはＳ／Ｎ比の悪い状態での実験データを解析して、本手法が有効であるかどうか検討する必要がある。また、結論でも述べたようにここで用いているアルゴリズムは、どちらかと言えば定常時系列のモデル同定アルゴリズムを無理に非定常状態に適用できるように工夫しているものであり、この点に関する検討が必要である。

良く知られているように、システムのパラメータが変動する非定常問題に対しては、制御工学では適応フィルタの概念が良く用いられている。しかし、この手法では、推定パラメータの修正法が逐次型ＡＲＭＡモデル同定手法とは全く異なり、望ましい値と推定値の差（誤差）に直接一定の修正係数を掛けるパラメータ修正法を用いており、従って、データの変動に即応しすぎて推定されたパラメータの統計的ばらつきが大きくなり過ぎることと、更に、パラメータ修正係数が逐次型ＡＲＭＡモデル同定手法のように自動的に決まらないため、この修正係数の決定を試行錯誤で決めなくてはならない点が問題である。ただ、即応性が良いと言う点で、適応フィルタを未臨界度推定に用いる研究も重要な

テーマであろう。

４．５　平成７年度（１９９５年度）

核燃料再処理工場等においては安価で即応性の良い臨界安全監視システムの開発を目的として過去４年間にわたって逐次型ＡＲＭＡモデル同定手法の安全監視システムへの応用と言う観点から各種の検討を行ってきた。そこで、この年度では、それまでに得られた知見を実際の臨界安全監視システムとしてシステム化するためには、システム構成をどの様にすれば良いかを検討した。しかし、ここでは実際にハードウエアとして設計する際に必要となる詳細な検討ではなく、システムとして構成するための基本概念についてのみの検討を行い、実際にハードウエアとして設計するための詳細な検討に関しては今後の課題とするものとした。

まず、システムの構成を考えるに当たってそのコンセプトを列挙する。

① 高信頼性

臨界安全監視システムによる異常通報は、プラントのシャットダウン等のトリガーになることを考慮した場合、誤警報は極力避ける必要がある。従って高い信頼性を確保するためにある程度の冗長性を有したシステムとする。

② 雑音対策

本手法が中性子雑音を対象としている性質上、その他の有色雑音を遮断する必要がある。そこで雑音の影響を受けやすいアナログレベルでの信号系は極力短くする。この時アナログ処理に関わるコンポーネントは検出器の近傍に設置することになるためモジュール構造にして小型化する。またデジタル信号系においても、より高い対雑音性を有する光情報伝送技術を活用する。

③ 高拡張性

取得した信号は、閉鎖されたシステムで扱うのではなく、様々な

手法・アルゴリズムを有機的にリンクさせ多角的に最良の未臨界度推定が行えるようにすると共に、新しい技術に即応できる様な柔軟性のあるシステムとする。

上記のコンセプトを踏まえた上で、検出部から解析部までシステム全体の系統の概要について考察した。また、システム内の信号ライン上で特に未臨界度推定性能に深く関わる構成ユニットである中性子検出器と前置信号処理装置についても考察を加えた。

最後にシステムの最終段となる解析装置でおこなう解析内容について信号の有機的リンクという観点から、その方法と拡張性について述べている。

4.5.1 システムの基本構成

本研究に基づくオンライン臨界安全監視システム基本の構成は、大きく次の５つのパートに分類される。

① 中性子検出部
② 信号前置処理装置
③ 光送信装置
④ 光受信装置
⑤ 解析・表示部

4.5.1-1 中性子検出部

一般的に臨界安全性がもっとも要求されるのは大量の核分裂物質が滞留する可能性のある溶解槽と考えられる。その点を考慮して本章では、システムの監視対象を溶解槽に限定して議論する。

本システムでは、中性子揺らぎから体系の未臨界度を推定するため、溶解槽には中性子検出器を配置する必要がある。中性子検出器には基本的にＣＩＣ(Compensated Ionization Chamber)を想定している。ただし、必ずしも他の選択肢がないわけではなく検出器の種類等に関する議論を４。２節で詳述することとする。

中性子検出器は、信号の有機的リンクを実現し多角的な解析によって安全性を保証する目的から複数の検出器を配置する

① 溶解槽中央部

中性子の検出感度が高く、本手法に基づいた未臨界度推定の基礎原理である一点炉近似動特性方程式に対する空間依存性の最も小さい位置である。

② 溶解槽中央周辺部

対象位置に複数点配置することにより、システムの信頼性を高めるとともに、それぞれの信号の相互相関に基づくデータの処理等を行うことにより多角的な信号解析に寄与する。

③ 溶解槽外周部

上記に対する補助的な役割を果たすものであり、検出器としてはＣＩＣではなく、γ線検出器を設置することになる。本研究の過程で、即発中性子に誘因されるγ線は統計的に中性子揺らぎと同様のメカニズムを保ちながら揺らいでいおり、さらに研究の余地を残してはいるものの未臨界度推定に利用できることを確認している。γ線検出法では空間依存性が無視できることや炉外設置にできるということでメンテナンスが格段に容易になるなどメリットも大きい。

4.5.1-2 前置信号処理装置

検出部から出力された信号は、信号ケーブルにより前置信号処理装置に導かれる。前置信号処理装置はさらに、アナログ信号処理パート・Ａ／Ｄ変換パート・デジタル信号処理パートの３つのパートに分けられる。各パートの機能の概略は、

① アナログ信号処理パート

検出器から導かれた信号をデジタル化する前段処理を行うパートである。検出部における検出器が電流タイプもしくはパルスタイプのいずれかにおいて処理が若干異なるが、基本的には、増幅・シェーピング・アンチエリアシング処理を行う。

これらの信号前処理部の特性は臨界安全を監視する対象の変化に対応出来きなくてはならない。そこで、これらの処理を行う部分の特性を遠隔制御できる機能を有するタイプのものを用いる。

② Ａ／Ｄ変換パート

信号をデジタル化する要件を満たされたアナログ信号をデジタル信号に変換するパートである。解析部で相互相関解析等を行うことを考慮して一つの検出器からの入力に対して一つのＡ／Ｄ変換器を割り当てて同時サンプリングを行う。また、解析サンプリング周波数に対して８倍程度のオーバーサンプリングを行うことにより前段のアナログ信号処理パートで行われるアンチエリアシング処理の負荷を減らし、元信号の位相特性を保証する。

③ デジタル信号処理パート

Ａ／Ｄ変換パートでオーバーサンプリングされたデジタル信号をデジタルローパスフィルタとデシメータによって解析サンプリング周波数に変換するパートである。

なお、アナログ処理パート及び、デジタル処理パートの各パーツにはプログラマブルタイプのものを用いることにより、解析部のコンピュータシステムからコントロールバスを通じて特性を最適なものに変更できる。

本研究のこれまでの経験から、「信号のクオリティー」が未臨界度推定性能に大きな影響を及ぼすことが確認されており、前置信号処理装置の果たす役割は大きい。

4.5.1-3 光送信装置

前置処理装置から出力されたデジタル信号は、信号ケーブルにより光送信装置に導かれる。光送信装置では、各チャンネルのデジタルデータの多重化を施した後、光パルスに変換して、光ファイバケーブルへ出力する。

本システムではここまでは管理区域内に設置されることになり、管理区域外と光ファイバケーブルにより結合される。

4.5.1-4 光受信装置

光送信装置から出力された光パルス信号は、光ファイバケーブルにより光受信装置に導かれる。光受信装置では、光パルスを電気信号に変換した後、多重

化されていた信号を再び分離する。

4.5.1-5 解析・表示部

メインコンピュータは、光受信装置との間に専用のバスとインターフェースを有し、光受信装置で分離されたそれぞれのチャンネルのデジタル信号を共有メモリ空間にストアする。この段階で信号の取得は完了し、そのデータを基に未臨界度推定の他各種の解析が行われる。

また、再処理工場内のプラントコンピュータとの接点を有することで、ホスト系へ警報を発信できる。更に、所内ＬＡＮにより解析データは各部署で閲覧及び再解析に供する事を可能とする。

4.5.2 中性子信号検出系

中性子信号検出部及び時系列データ発生部は、まず中性子検出器出力信号の揺らぎ成分を取り出し、このアナログ信号をディジタル時系列データ化するための信号処理部であり、その構成要素は中性子検出に何を用いるかによりかなりの差が出る。良く知られているように中性子検出器には電流型とパルス型があり、何れを用いてもそれぞれ利点と欠点を持っている。

電流検出器型では電流電圧変換プリアンプを用いることにより、直接、非常に簡単に中性子信号揺らぎを取り出すことができる。しかし、中性子検出器として例えば電流電離箱を用いる場合には、高圧ＤＣ電源としてプラス電源とマイナス電源の両方を用意しなければならないこと、また、信号揺らぎに電源系ノイズ、特に商用電源周波数の混入が避けがたい。そのため、現場に設置される中性子信号検出系毎に試行錯誤方式で雑音成分を最小にするための最適なアース方法・アースポイントを探索する必要がある。この様な方法を用いても必ずしも商用電源周波数を完全には除去仕切れない場合が多く、現実的な処理法としてアナログ信号処理系に商用電源周波数ノイズのみを除去するためのフィルタを挿入する必要がある。このことは、本研究が目的の一つとして挙げている安価で多くの現場に容易に設置できるシステムと言う観点からすると好ましくない特徴である。一方、 Fission Chamber、^{3}Ｈｅ、あるいはＢＦ$_{3}$等のようなパルス型の中性子検出器の場合、高圧電源はプラス電源のみでよく、信号に商用電源周波数の混入を心配する必要はないが、Ｆｉｇ．４－３ に示すようにパルス型信号処理系としてまず電流電圧変換リニアーアンプ、ｎ－γ波形弁

別器、シングルチャンネルアナライザ、波形整形アンプ、そして、パルスレートをアナログ信号に変換するためのレイトメータが必要となり、電流型検出器に比べて多くの高価な装置を用意する必要がある。従ってパルス中性子検出系も、本研究の目的とする安価なシステムと言う観点からは必ずしも好ましいとは言いがたいが、現場での調整が必要ないと言う点では電流電離箱より優れていると言える。また、 Fission Chamber は中性子検出感度が高く、比較的中性子束の低い燃料処理系での中性子検出系として有効な検出器である。

いずれの方式を採用するかは、中性子検出器感度、用いる各要素の価格、システムの安定度、雑音の混入の程度等色々な観点から検討されなければならない。

Table 4.1 Comparison of Neutron Detectors

中性子揺らぎ信号検出部の比較		
	電流型中性子検出系	パルス型中性子検出系
利点	電流電離箱と電流電圧変換増幅器を通して直ちに中性子信号揺らぎ成分を抽出できる。	1. ＤＣ高圧電源はプラス電源のみでよい。 2. 電源系雑音の混入を心配する必要がなく、現場での調整が必要ない。 3. 感度の高い検出器の利用が可能である
欠点	1. ＤＣ高圧電源にプラス・マイナス両電源が必要となる 2. 電源系からの雑音、特に商用電源周波数の混入が避けられない。 3. 商用電源周波数成分の除去はシステムの設置場所での現場調整が不可欠となる。	リニアーアンプ、ｎ－γ弁別、シングルチャンネルアナライザ、波形成型器、レートメータなど、中性子信号揺らぎ成分を取り出すため非常に多く設備が必要となり、電流型検出器に比べて高価なシステムになる可能性がある。

4.5.3 前置信号処理装置

前置信号処理装置については4.5.1-2で説明したように、３つの主要パート

から構成される。この構成内容からも分かるように前置信号処理装置は、アナログ信号とデジタル信号との接点として機能している。ここでは、信号のクオリティー維持について前置信号処理装置に要求されるべき内容を提示した上で、それぞれのパートの構成要素について説明する。

4.5.3-1 前置信号処理装置と信号のクオリティー

本手法による未臨界度推定は、手段としてデジタル信号値を用いて解析を行っているが、解析の対象としているのはアナログ量としての検出器信号であることは言うまでもない。検出器からの原信号自身は元来微弱であり、更にその信号のなかで情報源として利用できるのは平均値回りで存在する揺らぎの成分のみであることを考慮すると、アナログレベルでの信号を如何にデリケートに扱えるかが未臨界度推定の性能に大きく寄与してくる。前置信号処理装置に要求されるのは、原信号の揺らぎ成分だけを限りなく忠実にデジタル値に置き換えることにある。しかし、この要求を完全に満足することは次の要因から実質上不可能である。

① 外乱によるクオリティーの低下

もっとも顕著に観測される外乱としては、商用電源の周波数が検出器信号に重畳してくることにある。接地対策を施すことによってかなりの部分は改善されるが影響をゼロにすることはきわめて困難である。

② フィルタリングによるクオリティーの低下

アンチエリアシングフィルタや揺らぎ成分だけを抽出するためのハイパスフィルタに信号を通過させるというゲイン特性上の要請を満たすことにより反面的に位相特性等に歪みを生じる。

前置信号処理装置としては、これらの点を考慮して原信号のクオリティーの低下を最小限に押さえるような設計を行う必要がある。

4.5.3-2 アナログ信号処理パート

アナログ信号処理パートは、カレントアンプ・エリミネーションフィルタ・

ハイパスフィルタ・ローパスフィルタの４種のユニットから構成されている。それぞれの働きと要求すべき特性について以下に示す。

① カレントアンプ

カレントアンプはＣＩＣからの信号が最初に導かれるユニットである。カレントアンプには、低ノイズでかつ10^9以上の高ゲインが要求される。

② エリミネーションフィルタ

エリミネーションフィルタは、パッシブな方法であるアース対策によって除去しきれない信号に重畳してきた商用電源の周波数をアクティブに除去するものである。ＡＲＭＡ同定の立場からは避けたい方法でもあるのでアース対策により無視できるレベルにまでノイズを除去できる場合には、Ｆｉｇ．４－４に示してあるようにスイッチングによりバイパスさせることも可能とする。

③ ハイパスフィルタ

ハイパスフィルタは、信号のバイアス成分を除去して平均値回りの揺らぎを抽出するものである。基本的にはＤＣ成分のみを除去するためのもであるが、これまでの研究から得られたＡＲＭＡモデル同定しきれない遅発中性子の寄与による低周波成分の要素も除去するべきであるという見地に立ってカットオフ周波数を考慮しなければならない。また周波数特性は、ＡＲＭＡモデル同定時にハイパスフィルタの特性を同定しないように低次（１次）のバターワース特性を有したものを用いる。

④ ローパスフィルタ

ローパスフィルタは、アンチエリアシングフィルタとして機能する。一般にアナログ信号レベルで完全なアンチエリアシングを行おうとすると、急峻な減衰特性を確保するために多連立のチェビシェフ特性フィルタを使用する事になるが、その場合カットオフ周波数近傍での位相特性は非常に悪いものとなる。そこで、原信

号を忠実に再現するという立場から本システムでは、後段のA／D変換部でオーバーサンプリングを行いデジタル信号処理パートでデジタルフィルタリングとデシメーションを行うという前提の下で低次のバターワース特性のローパスフィルタを用いる。

また最近では、プログラマブル型の半導体デバイスがワンチップで供給されていることを考慮して、アナログ信号処理パートはメインコンピュータからコントロールバスを通じて、論理信号でその特性値をコントロールできるシステムとする。この際、スイッチングゲートにはフォトカプラなどのデバイスを用いることにより外部雑音とのアイソレーションにも配慮する必要がある。

4.5.3-3　A／D変換パート

アナログ信号処理パートで整形された信号は、一つのチャンネルにつき一つのA／D変換器によりデジタル変換される。このような方式にするのは、後段の解析部で相互相関をとること等を考慮して、信号の同時性を確保するためである。

サンプリング周波数は、解析に供するサンプリング周波数の８倍程度のオーバーサンプリングを行う。本システムでは解析時のナイキスト周波数が最大10kHzと予想されることから解析サンプリング周波数は20kHzとなり、その８倍のオーバーサンプリングを行うと160kHzでサンプリングを実施することになる。

量子化に関しては16bitの直線量子化を行う。これまでの研究の結果よりＡＲＭＡ同定を行うに当たっては12bit以上の精度でがあれば十分であることを確認している。

4.5.3-4　デジタル信号処理パート

A／D変換された信号はデジタル信号処理パートに導かれる。デジタル信号処理パートはデジタルローパスフィルタとデシメーションフィルタの２種のユニットから構成されている。それぞれの働きと要求すべき特性について以下に示す。

① デジタルローパスフィルタ

デジタルローパスフィルタはオーバーサンプリングされたデジタルデータを次に説明するデシメーションにより解析サンプリング周波数に下げてもエリアシングが生じないように解析時のナイキスト周波数以上の周波数成分を除去するものである。

4.3.2項でも述べたように、エリアシングの防止をアナログ信号処理パートに担保させずにデジタル信号処理パートで行うのは、アナログ時に急峻な減衰特性を有したフィルタを通過させると信号の位相特性が極度に悪くなるからであった。本パートでもアンチエリアシングフィルタとして機能させることから急峻な減衰特性を有することには変わりはないが、フィルタの伝達関数が零点と原点の極しか有しない

ＦＩＲ(Finite Inpulse Response)型のフィルタとすることにより位相直線性を実現できる。

② デシメーションフィルタ

デシメーションフィルタは、オーバーサンプリングされたデジタル信号を実際に解析に供するサンプリング周波数に変換するためのものである。実現方法としてはデジタル信号を間引き抽出する事により実施する。従って、８倍オーバーサンプリングされたデータの場合には実在するデジタル８個毎に１個のデータを抽出することになる。

4.5.4 解析・表示部の概要

メインコンピュータでの解析は、ＲＰＥ法を用いたＡＲＭＡ同定による未臨界度推定をメインとして、それを多角的にサポートするために、自己相関法によるスペクトラム推定、観測雑音レベルが高いときに有効な相互相関法によるスペクトラム推定を行い、ＲＰＥ法のアルゴリズムの破綻等が生じていないこと等を確認するシステムを作る。また近年、研究の進んできた適応フィルタを

用いたＡＲＭＡ同定による未臨界度推定を導入し、特に急激な反応度添加に対して迅速な対応がとれる体勢を確保する。更に、ＡＲＭＡ同定手法には初期仮説が存在するため、その仮説が崩れていないことを保証するための統計計算プロセスも補助ストリームとして機能する。システムの最終目標として、これらの機能が有機的にリンクし、ＲＰＥ法で用いる諸設定値の自動修正等を自動的に行えるシステムとする。以下にそれぞれの解析ストリームについて要点を述べる。

4.5.4-1　ＲＰＥ法によるＡＲＭＡモデル同定

このストリームは、本手法による未臨界度推定の中核となる部分であることは言うまでもない。ここでは、解析処理における実際の流れについて順を追って説明する。

STEP 1: ＲＰＥ法によるＡＲＭＡモデル同定

設定された解析サンプリング周波数に同期して、メモリーからデータを読み出して第３章で説明したアルゴリズムで処理を行う。仮に解析サンプリング周波数が２０ｋＨｚであるとすると、１秒間あたり２万点のパラメータが推定される。

STEP 2: パラメータの間引き

１秒間あたり２万点という多量のデータを管理保存することは容易ではないため、実際人間がトレンドを合理的に監視できるレベルまで推定パラメータを間引いた上で再処理系のシステム履歴として保存する。異常が発生した時には、流量計などの他のプラントパラメータと保存したシステムパラメータから得られる様々な情報を照らし合わせることによって、システムがどのような経過をたどったのかを推測しプラントの安全管理に寄与するための一手段となる。

STEP 3: システムポール(極)の算出

推定パラメータのＡＲ部よりシステムポールを計算する。この際、解が２種類算出されるが、システムの周波数特性の高周波側の折

れ点を表す方のみを残し、もう一方は棄却する。

STEP 4: 未臨界度変換式へのシステムポールの代入
システムポールから未臨界度への変換を行う。（変換式は第２章で説明した原理に従って未臨界度ρとシステムポールの対応テーブルを作成し、更にネブィレ補間多項式等によりなる変換式を予め構築しておく）

STEP 5: 算出された未臨界度のプロッティングと保存
未臨界度の時間トレンドをディスプレー上にプロット表示するとともに保存する。

4.5.4-2 自己相関法によるスペクトラム推定

このストリームは、ＡＲＭＡ同定法という物理モデルに基づいたパラメトリックな推定方法に対して、自己相関とフーリエ変換によって信号のパワースペクトラム密度を直接的に導出するストリームである。ＲＰＥ法を用いたＡＲＭＡ同定ストリームと異なり、本ストリームではその過程に『平均化』の概念が入ってくるため、逐次的に状態を推定したり、非定常状態に対応することはできないが、生の信号がもっている周波数特性を直接観測できるという特性を有している。

4.5.4-3 相互相関法によるスペクトラム推定

このストリームは、自己相関法によるスペクトラム推定と同様にノンパラメトリックな推定方法であり、相互相関とフーリエ変換によって信号のクロスパワースペクトラム密度を直接的に導出するストリームである。基本的な考え方は自己相関による方法と同一であるが、相互相関を用いることにより観測空間の無相関雑音を除去することが可能なためＳ／Ｎ比が悪い状態であっても信号の周波数特性を観測し易いという特性を有している。

4.5.4-4 適応フィルタ(ＡＤＦ)によるＡＲＭＡ同定

ＡＲＭＡモデル同定アルゴリズムは定常系のパラメータ同定を基本として

いるため、未臨界度の変動追跡速度に問題があることは既に述べたことである。そのため我々は、Ｐ行列対角要素活性化法を提案しその有効性を示したが、本来、逐次型ＡＲＭＡモデル同定アルゴリズムはパラメータの変動を追跡するためのアルゴリズムではない。これに対し、信号処理の分野で近年注目を集めている適応ディジタルフィルタ（Ａdaptive Ｄigital Ｆilter：ＡＤＦ）は、ＡＲＭＡモデルそのものを直接同定する手法ではないが、非定常系の観測信号の最適推定値を与えるモデルを最小２乗法によってオンライン同定する手法であり、未臨界度推定のように時間的に変動することが問題となるシステムのパラメータ推定に適しているように思われ、未臨界度追跡速度の改善が期待される。しかし、ＡＤＦによるＡＲＭＡ過程のシステム同定を行う場合には、パラメータの修正係数はＡＲＭＡモデル同定アルゴリズムの場合とは異なり、自動的に計算されるものではなく、入力信号の分散に応じて経験的に与える必要がある。従って、信号の分散からトライアルエラー方式で実験的、経験的に修正係数の最適値を見付けなければならない。更に、一端修正係数を決定してしまえば、その後はその値は変更されることはなく、推定されるパラメータは観測された信号の統計的変動を反映して常にある分散を持って変動することになり、ＡＲＭＡモデル同定のように定常状態でパラメータが収束すると言うことがない。従って、同定される対象システム自身のパラメータ変動の追跡能力は高いが推定されるパラメータの統計的変動の大きな手法であることを承知してＡＤＦを利用するれば、ＡＤＦも臨界安全監視システムの補助として十分利用価値のある手法であろう。

適応フィルタの基本的な動作は単純で、信号ｘ（ｋ）を適応フィルタに入力したときの出力が希望する信号ｄ（ｋ）に等しくなるように、つまり、両者の誤差ｅ（ｋ）ができるだけ小さくなるようにフィルタの特性、具体的には、そのインパルス応答を徐々に変えていくだけである。そのインパルス応答を変化させる方法が適応アルゴリズムと呼ばれるものである。

最も広く使われている適応アルゴリズムは、**ＬＭＳアルゴリズム**（least-mean-square algorithm）と呼ばれているものであり、２乗平均誤差が最小になるように、システムの特性を決めていくものである。アルゴリズムの理論的説明は付録に譲る。

4.5.4-5　統計量計算プロセス

このストリームは、ＡＲＭＡ同定法が信号に対して要求する統計的初期仮説を検定するという目的、検出部から光伝送系にいたるまでの本システムの機能的な異常を早期に発見するという目的、そして再処理系の異常判断に関する一つのトリガーとしての３つの目的を有するストリームである。

計算する統計量としては、データの平均値・分散であり、またこれらを基にカイ二乗検定やヒストグラムの作成を行う。

4 研究経過と成果

4.6 平成8年度(1996年度)

核燃料再処理工場等に用いられる安価で即応性の良い臨界安全監視システムの開発を目的として、過去5年間にわたって逐次型ARMAモデル同定手法の安全監視システムへの応用と言う観点から各種の検討を行ってきた。

これらの成果を実際のシステムとして構築することがこれからの課題であり、本年度はこれらの知見を実際の臨界安全監視システムとしてシステム化するために必要となるモジュールに付いて検討を行うと共に、平成8年3月に旧動燃事業団・大洗工学センターのDCA実験装置において行われた炉雑音実験データを解析し、これまでの手法の有効性と限界を確認すると共に以下の結論を得た。

ここで提案しているシステムの設計基本思想は高信頼性、耐雑音性、そして高拡張性に置いている。従来、この様な特定の目的を持ったシステムでは、高速信号処理、警報発生の遅れの軽減、また、使用計算機の負荷の軽減、システム全体の小型化等の目的で、解析アルゴリズムはROM化され、システムは固定化される傾向にあった。従って、一度システムが完成してしまうとその変更は容易ではなかったが、近年パーソナル計算機、或いはエンジニアリングワークステーションの性能は飛躍的に向上し、その発展と変革は目を見張るものがある。この様にめざましい発展を遂げている計算機に着目すると、信号解析、表示、警報信号発生に、それらのアルゴリズムをROM化したICチップを用いるよりも、高性能の計算機を用意し、各種の信号解析ソフトウエアや、必要に応じて他のシステム同定アルゴリズムを組み合わせた総合的信号処理システムとすることが望ましい。この様な構成にすることにより、ソフトウエアや解析技術の進歩を即座にシステムに取り入れることが可能となるばかりでなく、特別なシステムを付加することなく、逐次計算型ARMAモデル同定法に基づく未臨界度推定法による結果に対し、補間的あるいは参考情報を容易に提供することが可能となり、しかも、時代と共に進歩・発展する柔軟な臨界安全監視システムを構築することができる。

そこで、臨界安全監視システムは計算機を中心とするシステム構成とし、

1．　臨界安全監視システムは高い信頼性を確保することが最も重要でありそのためシステムは冗長性を有し、また、本手法が中性子雑音を対象としている性質上、外部からの有色雑音を遮断するシステムとなっていなければならないこと、

2．　雑音の影響を受けやすいアナログ信号系は極力短くする必要があること、そのため、アナログ処理に関わる信号処理要素は中性子検出器の近傍に設置することになるためモジュール構造として、極力小型化すること、デジタル信号系においても、より高い対雑音性を有する光情報伝送技術を活用すること、

3．　取得した信号は、閉鎖されたシステムで扱うのではなく、様々な手法・アルゴリズムを有機的にリンクさせ多角的に最良の未臨界度推定が行えるようにすると共に、新しい技術に即応できる様な柔軟性のあるシステムとすること、等を考慮し、

①　中性子信号検出モジュール

②　信号前置処理モジュール

③　光送信モジュール

④　光受信モジュール

⑤　解析表示モジュール

からなるシステム構成を提案した。性子信号を検出し、アナログ／ディジタル変換を行うシステム構成は中性子検出器に何を用いるかによって変わる。しかし、今回のＤＣＡ実験データの解析結果はパルス型の中性子検出器の方が、電流型よりも有効であることを示している。電流型の場合には、ハムノイズを除去するための適当なノッチフィルタを設計しなくてはならない。

1．　従来は、同軸ケーブルなどを用いて信号の伝送を行ってきたが、近年、光ケーブルを用いた信号の伝送技術が主流となっている。そこで、本システムでは光伝送制御システムの利用を提案する。この技術は大量の情報の伝送を可能にするばかりでなく各モジュールの遠隔制御を可能にする。また、この方法は、耐電磁誘導障害、耐火性に富み、システムの信頼性を高める。

2．　信号解析部として、システムを固定化してしまうＲＯＭチップを用いるよりも、近年目覚ましい進歩を示している高性能計算機と各種の信号解析ソフトウエアを組み合わせた総合信号解析処理システムを提案した。この手法はソフトウエアや解析技術の進歩を即座にシステムに取り入れることが可能となるばかりでなく、特別なシステムを付加することなく、時代と共に進歩・発展する柔軟な臨界安全監視システムを構築することができる。

この様に、本システムの設計基本思想は高信頼性、耐雑音性、そして高拡張性に置いており、臨界安全監視システムは計算機を中心とするシステム構成として、(1)中性子信号検出部及びアナログ／ディジタル変換部、(2)ディジタル信号の光伝送制御システム、(3)信号解析及び表示警報信号発生部の３つの部分システムから構成される。これらのモジュールは基本的に平成７年度の報告書にまとめたものの具体化について述べたものであり、ここではその詳細は割愛する。

以下に、１９９６年３月１１、１２日の両日に、動燃事業団・大洗工学センターのＤＣＡ実験装置を用いて行われた中性子雑音測定実験の解析により得られた結果をまとめておく。

この実験装置では未臨界度を重水の水位で調整できるため非常に深い未臨界度から、臨界に近い状態まで任意の未臨界度における実験が可能であり、また、重水の注入方法を調整することにより非定常状態を模擬することができる。

4.6.1　解析結果から得られた知見

ＤＣＡ実験装置を用いて行われた未臨界定常運転時の実効増倍係数が0.991～0.881に対する中性子信号揺らぎデータを解析し、以下の知見を得た。

1．重水の水位１２００ｍｍに対応する実効増倍係数が０．８８９以下ではＡＲＭＡモデル同定では有意な情報が得られない。特に、水位１０００ｍｍの場合には全くＡＲＭＡ同定ができないほどスペクトルは白色である。

2．Ｃｆ中性子源を含む中性子検出器出力は中性子源の影響が非常に大きく、最も条件の良い実効増倍係数が０．９９１の場合でも周波数スペクトルは殆ど平坦で、未臨界度推定には利用できない。

３．非定常運転時に対するＩＤ００９～ＩＤ００１２に於ける未臨界度では、ＩＤ００９とＩＤ０１２はそれぞれ定常状態の条件は全く同じであるにも　かかわらずＡＲＭＡ同定結果に大きな差異が認められる。今回の解析では　この原因を明確にすることが今後の課題である。

４．ＩＤ０１３～１６の中性子検出器の電流モード実験では、非常に大きな電源ノイズが混入しており、今回は２０Ｈｚ以上のスペクトル成分を除去するローパスフィルタを用いてＡＲＭＡモデル同定を行ったが、一応満足できる結果を得た。しかし、実際の再処理系では未臨界度に対応する折れ点周波数は数百Ｈｚとなることが予測され、特定の周波数を特に鋭い特性で除去するノッチフィルタが必要になる。

５．本解析では、データの前処理として０．１Ｈｚに折れ点周波数を持つハイパスフィルタを用いたが、この場合オリジナルデータをこのフィルタに通した場合、始めの数千データには過渡現象が現れており、フィルタの特性によってはこの様な過渡現象を考慮しなくてはならないことが判明した。

4.6.2 データ取り込みのための知見

①　ＤＡＴテープに記録されているデータは４８ＫＨｚサンプリング周波数であるから、４８KHzを分周レートで割った周波数しかサンプリング周波数として指定することが出来ないため、オリジナルテープからディジタル信号を計算機に取り込む場合の間引き数の指定がエリアシングの問題に関連して重要な問題となっているが、実際には適当なローパスフィルタを用いて信号前処理を行うのでこの点については問題はない。

②　非定常の状態でのＡＲＭＡモデル同定で重要な課題は、ＤＣ成分の除去であり、今回はハイパスフィルタを用いたが非常に低い周波数に折れ点のあってシャープな特性を持つディジタルフィルタの設計は非常に難しく、この点は今後の問題である。しかし、このノイズをそのままにしてＡＲＭＡ同定を行い、ある程度順当な結果が得られており、今後更に検討を行う予定である。

③　今回の実験で行ったパルス型中性子検出器の電流モード信号取得では電源周波数のノイズが大きなエネルギーで混入しており、ＡＲＭＡ同定のための信号前処理が非常に困難であった。このことから、実際のシステムではパルス型の中性子検出が適当であると言える。

４．７　平成９年度（１９９７年度）

平成９年度は、過去６年間の研究によって得られた知見を基に、実際の臨界安全監視システムとしてシステム化するために必要となるシステムの構成要素モジュールに関する検討をおこなった。

前年度、平成８年度に提案した解析表示モジュールの中で、ＡＲＭＡモデル同定と平行して信号解析を行う冗長系として重要な要素となるアダプティブフィルタ（ＡＤＦ）アルゴリズムモジュールの開発と、このモジュールを用いた実データ解析を行い、ＡＤＦアルゴリズムの未臨界度推定手法としての有効性を検討し、以下の成果を得た。

4.7.1 ＡＤＦによるＡＲＭＡモデル同定法

ＡＤＦによるシステム同定において、その推定パラメータが修正係数に大きく依存することは、ＡＤＦの原理から明白である。本研究においては、修正係数は３段階に分けて同定を行ったが、その値が大きい時には推定されるパラメータは入力データの統計的性質の変化に素早く追従するが、その反面推定されたパラメータの統計的変動が大きく、逆に修正係数を小さくすると統計的変動は小さくなりほぼ収束しているが、一方、実際にシステムパラメータに変動がある場合には追従が遅くなっており、このことは定常状態、非定常状態を問わずすべてのＡＤＦのパラメータ推定結果について言える。

しかし、推定されたパラメータ並びに、その結果から得られる未臨界度はいずれの場合においても統計的変動が非常に大きい。この統計的変動は逐次型ＡＲＭＡモデル同定アルゴリズムによる未臨界度推定の場合に比べて非常に大きく、ＡＤＦアルゴリズムによる結果をそのまま未臨界度推定値とするにはやや問題がある。従って、推定結果を平滑化する二次的なアルゴリズムを付加することが重要になる。そこで、本研究では、ＡＤＦによる推定パラメータに５００次の単純移動平均、更に、その平滑化されたパラメータの再移動平均、５００データ毎の相加平均などの平滑化を行った。これらには大きな差がなくいずれも統計的変動は小さくなっているが、しかし、相加平均を取れば平均を取った個数（５００個）の単位で情報の遅れが生じる。

相加平均では、５００点づつの平均であるから表示データ数がかなり少なくなるが、計算アルゴリズムが至って簡単でデータ数も少ないため、演算して表

示するのに計算を必要とせず、未臨界度の概算を知るのにはよい方法であろう。

4.7.2　未臨界度

未臨界度については、中間の修正係数を使ったＡＤＦのパラメータ同定結果を使って求めた。推定されたパラメータを持つ伝達関数のポールの値はパラメータの統計的変動と比べると、移動平均を取らなくても統計的変動が比較的少なくほぼ一定値に収束し、かつ、速く追従している。

未臨界度において、同じ修正係数を用いた場合、ＡＲＭＡ（２，２）モデル同定の結果をＡＲ（１，１）モデルの場合と比べると、同定の初期ではＡＲＭＡ（２，２）モデルの方が収束が早いが、統計的変動に関してはＡＲＭＡ（２，２）モデルの方が推定値の揺らぎが大きい。これは基のデータに含まれる情報がＡＲＭＡ（１，１）モデルでは、２つのパラメータに宿約されるのに対し、ＡＲＭＡ（２，２）モデルでは、４つのパラメータに分散されるためではないかと考えられる。

また、移動平均を施した結果を見ると、ＡＲＭＡ（２，２）モデルの場合の方が、未臨界度がやや小さめになっている。これはＡＲＭＡ（２，２）モデルの方がパラメータ収束が早く、少ないデータで定常状態に収束したためと考えられ、最終的な推定値にはそれほど大きな差は生じていない。

未臨界度に一回の移動平均による平滑化を行った場合、明らかにＡＲＭＡ（１，１）モデルの場合の方が統計的変動が少ない。

以上の結果を総合的に判断するとて、計算量が少なくて済むＡＲＭＡ（１，１）モデル同定はＡＤＦアルゴリズムによる未臨界度推定法として有効な有効な手法であると結論できる。

4.7.3　今後の課題

（１）修正係数の選択

本研究においては、ＡＲ部とＭＡ部の修正係数は同じ値を用いている。修正係数が大きい時においては、修正係数が小さい時に比べ推定パラメータがまったく違う値になってしまうことがあった。これはＡＲ部とＭＡ部の修正係数が最適に選択されていなかったことによる可能性もある。

ここで報告したプログラムではＡＲ部とＭＡ部の修正係数をそれぞれ独自に設定できるように作成されており、今後、ＡＲ部とＭＡ部の修正係数を変え

て、ＡＤＦによるパラメータ推定の速さや統計的変動にどの様な効果があるかを検討すべきであろう。

一般にＡＤＦアルゴリズムでは修正係数の最適な選択は試行錯誤的に、信号の分散の何％にするかを決める必要がある。この問題に対しては、信号レベルが時間と共に変動する場合は、分散を信号レベルの変化に応じて計算し、分散の変動に応じて修正係数を変化させる手法も提案されており、今後、この様な手法の検討を行う必要がある。また、ＬＭＳ法はパラメータが収束した状態で、修正係数を変えずに使用し続けると還ってパラメータが発散するなどが生じることが報告されており、この問題に関しては、ＡＲＭＡモデル同定アルゴリズムで用いられている忘却係数の導入の方法が提案されている。この様な手法の検討も今後行っていく必要があろう。

（２）移動平均による統計的変動の削減

ＡＤＦによるパラメータ推定値、並びに未臨界度推定値の統計的変動を押さえる最も簡単な手法として本研究では移動平均を用い、所期の目的をある程度達成できることが分かったが、逐次型ＡＲＭＡモデル同定アルゴリズムに基づく方法に比べて、やはり統計的変動が大きい。これは、それぞれの手法の本質的な特性によるものであって、必ずしもどちらが優れていると言い切ることのできない問題である。ＡＤＦアルゴリズムの即応性の良さを有効に利用し、且つ、統計的変動を少なくするには、非常に緩やかに変動する信号（推定されたパラメータ）に何らかの外乱が混入しているモデルを考えて、この信号をローパスフィルタに通すことにより統計的変動を減らす方法などを考える必要があるであろう。

これらの結果は、未臨界度の統計的変動を小さくするために移動平均による平滑化を導入することにより、ＡＤＦアルゴリズムによるＡＲＭＡモデル同定が充分未臨界度推定法として有効であることを示している。

４．８　平成１０年度（１９９８年度）

臨界安全監視システムは高い信頼性と新しい手法を容易に取り込むことの

できる柔軟性が必要である。この様な観点から１９９７年にはシステムの基本構成を報告した。ここでは、この基本デザインに新たに加えるべきシステムと手法について報告すると共に、１９９６年のＤＣＡ実験データの解析結果を報告した。

１：本報告書では、先ず、ガンマ線信号を用いた Feynman-α法による未臨界度推定モジュールと、その信号取得に必要となる高速パルス信号処理系の追加を提案した。ガンマ線信号に対して Feynman-α法を用いることにより原子炉の減衰定数を推定することが可能であるこが最近報告されており、このモジュールを追加することは、計数率の広い範囲にわたって臨界安全監視システムの信頼性を高めるものと考えられる。

２：前年度、ＡＲＭＡモデル同定と平行して信号解析を行う冗長系として最小２乗法に基づくアダプティブフィルタ（ＡＤＦ）アルゴリズムモジュールを報告したが、この手法は有色雑音が混入する信号に対してパラメータの収束が遅くなることが知られており、ここでは近年注目されているブロックアルゴリズムの基礎理論と、これを実現するための跳躍アルゴリズムについて報告した。

３：アナログ信号系では不可避の電源雑音を除去するノッチフィルタの設計法と実際のソフトウエアについて報告した。このフィルタは処理されるデータのサンプリング周波数と電源周波数に対して設計されるもので、電源周波数の雑音のみならず、その高調波成分並びに直流成分も同時に除去することができるので、ＡＲＭＡモデル同定やＡＤＦを適用する際に必ず必要となるハイパスフィルタが不要となる。

４：上記ノッチフィルタをサンプリング周波数４００Ｈｚ、電源周波数５０Ｈｚとして設計、これを１９９６年のＤＣＡ実験データＩＤ０１３に適当し、その有効性を確認した。

以下に、再考されたモジュールについて述べる。

臨界安全監視システムによる異常検出には、極力誤警報を避ける必要がある。従って高い信頼性を確保するため冗長性を有したシステムとすることが基本である。本研究の出発点は、中性子検出器揺らぎ信号を対象としていたが、臨

界安全監視システムとしての機能の点から、中性子検出器信号揺らぎのみならず、中性子パルス信号を直接利用する手法を考慮したシステム構成を提案してきた。

平成８年度の報告書では、雑音の影響を受けやすいアナログレベルでの信号系は極力短くし、アナログ処理に関わるコンポーネントは検出器の近傍に設置すると共に、モジュール構造にして小型化する一方、ディジタル信号系においても、より高い耐雑音性を有する光情報伝送技術を活用する構成としている。さらに、取得した信号は、閉鎖されたシステムで扱うのではなく、様々な手法・アルゴリズムを有機的にリンクさせ多角的に最良の未臨界度推定が行える計算機システムとして、新しい技術に即応できる柔軟性のあるシステムを提案した。即ち、高信頼性、雑音対策、高拡張性がシステム設計の基本コンセプトである

既に本委託研究初年度に於いて、オンライン臨界安全システムの構成モジュールについてその基本概念を考察し報告したが、その後の研究や調査により新たに分かったことを加えて、改めて臨界安全システムの構成に付いて報告する。

ここでは、新たにシステムとして改良すべき点について幾つか述べる。

1)　燃料再処理系ではガンマ線は避けることのできないの雑音信号であるが、この情報を有効利用することがオンライン臨界安全監視システムの信頼性を高めることは疑う余地がない。ガンマ線による炉雑音解析に関する論文は理論的解析の分野では既に１９７０年ころより報告されていたが、実際にガンマ線のみを計測することが非常に困難であることから、実験的考察を行った論文は見られなかった。しかし、我々は近畿大学原子炉を用いてガンマ線による炉雑音解析に関連する基礎研究を進めた結果から、ガンマ線検出器信号の利用が有効であるとの感触を得ており、今後、実際に臨界安全監視システム作成に当たってはガンマ線信号パルス系モジュールが重要になってくるものと考えられる。そこで、以下のようなモジュール構成を提案する。

4.8.1　センサーモジュール

システムの信頼性の向上と冗長性の確保を行うため中性子信号検出モジュールと並列にガンマ線検出器システムとして、 ガンマ線検出モジュール　を追加するべきであろう。

核燃料再処理系ではガンマ線は中性子検出に対しては、どちらかと言え

ばやっかいなノイズとも考えられるが、万一、核分裂数が増加した場合には当然高エネルギーガンマ線が伴って発生するので、安全監視システムとしては積極的にこの情報を利用すべきであろう。 近畿大学で近年行ってきた研究でも、ガンマ線パルス信号の時間領域での統計的解析から原子炉の減衰計数を推定できることが確認されている。また、九州大学では中性子検出器信号、並びにガンマ線信号をパルス信号として直接計算機ハードディスクに記録する装置を開発中で、既にその有効性を確認する実験が行われたと聞いている。ガンマ線信号は中性子信号に比べて燃料再処理系の構造物の影響を受けにくく、積極的にガンマ線情報を利用する方法をオンライン臨界安全監視システム追加すべきである。この様なシステムを追加することに伴い、信号処理系に関しても、高速パルス系信号処理モジュールを追加する必要が出てくる。ガンマ線計測は中性子計測よりも容易ではないかと思われる。

核燃料再処理系では信号の効率よい取得が重要な課題となることが予想される。この問題に対しては、シンチレータと光ファイバを組み合わせた放射線計測法が有効である。すでに、この方法によってガンマ線計測を行っている研究があり、また、放射線の空間分布測定にこの手法を用いた研究も報告されている。従来の放射線計測では、そのほとんどが一点での計測であり、広い面積にわたる情報を収集するシステムには向いていない。しかし、臨界安全監視システムでは理論的に一点炉近似の理論を基とする未臨界度を推定し、警報を出すシステムであるから中性子検出であれ、ガンマ線計測であれ、いずれにしても出来るだけシステム全体の計測情報を集めることが重要になる。ここでシンチレータと光ファイバを組み合わせた放射線計測法を改良すれば燃料再処理系の面としての放射線計測が可能になり、従来の点としての放射線計測では不可能であったような低計数率場合にも、未臨界度推定が可能になる可能性が出てくるものと期待される。

4.8.2 信号前置処理モジュール

１９９７年３月の報告書では前置信号処理モジュールはさらに、アナログ信号処理パート・Ａ／Ｄ変換パート・デジタル信号処理パートの３つのパートを提案したが、中性子検出器パルス信号処理系、並びにガンマ線検出器信号をパルス信号として取り扱う高速パルス信号処理系を新たに付けする必要がある。

シンチレータで発生した光は光電増倍管に集められ、高速パルスを取り出すため信号は増幅される前のAnodeから直接とり出す。こうすることにより非常に短いパルス信号を取り出すことが出来るためパルス系でのデッドタイムの問題が無視できるようになり、高計数率の場合にも対応できるようになる。

アノードから取り出されたパルス信号は Fast Amp で増幅されたあと、Fast Discriminatorにより適当な閾値以上のパルスを取り出し、パルス波高、パルス発生時刻記録系に送られる。

パルス波高・パルス発生時刻記録系は、この装置に入力として送られてくるパルス信号の振幅をＡＤ変換して取り込むと同時に、そのパルスが到着した時刻も同時に記録する装置で、パルスの波高をＡＤ変換するＡＤ変換部分と、ＦＩＦＯメモリバッファを持ちパルスの波高とパルスの到着時刻を記録するための信号処理部分から構成される。

4.8.3　解析表示モジュール

これまでの臨界安全監視システムでは、主として中性子検出器信号揺らぎを用いる手法について述べている。即ち、未臨界度が非常に深い場合には、中性子検出器信号揺らぎは殆ど白色となるため有意な未臨界度情報が得られないので、この様な状態では基本的にはパルスタイプの中性子検出器から一定時間内に観測されるパルス数の統計的揺らぎを用いた中性子相関法を利用すべきであり、比較的未臨界度が浅く中性子係数率が高く、逐次型ＡＲＭＡモデル同定を用いる解析モジュールについて述べた。また、未臨界度が深く中性子係数率が低く、周波数スペクトル解析手法が有効に働かない場合に用いる中性子相関法に付いてはその手法の原理のみを述べるにとどめた。

新たに提案する信号前置処理モジュールで追加せれるパルス波高・パルス発生時刻記録系はＤＯＳ／Ｖ系パソコンとＡＤ変換ボード及び信号パルス信号処理記録メモリバッファからなり、解析モジュールで使用される計算機の一部に組み込むことが可能である。近年、マルチＣＰＵの計算機が一般化しつつあり、解析モジュールもこの様な計算機を用いたマルチタスク処理システム構成とすることが考えられ、パルス波高・パルス発生時刻記録系はその一部となる。

先にも述べた様に、放射線検出センサーとして、中性子検出器、ガンマ線検出器（光ファイバを利用したシンチレーションカウンタなど）を用いることにより処理すべき情報量が２倍に増加し、再処理系に含まれる核燃料物質が少量

の場合から、相当量処理槽に存在する場合まで、あらゆる状況に対応出来る臨界安全監視システムが構築できるものと期待される。

(1-3-1) 低計数率領域

低計数率領域では、アナログ信号処理系は役に立たないので、当然パルス系の信号処理系を用い、未臨界度推定手法としては、パルス計測による時間領域の統計的解析手法である Feynman-α法を用いることになる。信号源としては、中性子信号の法が有利であるが、ガンマ線信号に関しても閾値を高エネルギー領域に設定すれば利用可能になるものと思われる。

中性子信号とガンマ線信号とを用いて未臨界度の推定を行うことは、冗長性と言う立場から考えると、非常に重要であろう。なお、Rossi-α法も考えられるが、計測に要する時間的な問題があり、応用するには問題があるのではないかと思われる。

(1-3-2) 中計数率領域

この領域では、パルス計測と電流計測の併用が望ましい。従って、Feynman-α法、ＦＦＴ法によるスペクトル解析と最小２乗法による伝達関数推定、オンライン ARMA モデル同定などの方法による推定結果を総合的に判断するシステム並列計算機の構成によって実現する。また、この領域では観測雑音レベルが高い場合に有効な相互相関法によるスペクトラム推定を行い、ＲＰＥ法のアルゴリズム破綻等が生じていないこと等を確認するシステムを作る。また近年、研究の進んできた適応フィルタを用いたＡＲＭＡ同定による未臨界度推定を導入し、特に急激な反応度添加に対して迅速な対応がとれる体勢を確保する。

(1-3-3) 高計数率領域

この領域では基本的にはＣＩＣ等の中性子検出器による ARMA モデル同定法を用いるが、今回提案している高速パルス系出処理されたガンマ線パルス信号を用いた減衰定数の推定を行い、また、同定される対象システム自身のパラメータ変動の追跡能力は高いが推定されるパラメータの統計的変動の大きな手法であることを承知してＡＤＦを利用等を並列に行い、最終的にはこれらの機能が有機的にリンクし、ＲＰＥ法用いる諸設定値の自動修正等を自動的に行え

るシステムとする。

４．９　平成１１年度（１９９９年度）

本研究に用いている基本的手法は、未臨界体系で観測される中性子信号揺らぎをオンライン計算処理して未臨界度を推定しようとする試みであり、オンラインＡＲＭＡモデル同定手法を安全監視システムに応用するためのシステム構成要素モジュールの開発、核燃料サイクル開発機構のＤＣＡ実験データ解析など実用化に必要となる知見の蓄積を行ってきたが、この間、Microsoft Windows を基本ソフトとするＤＯＳ/Ｖ系パーソナル計算機の爆発的な普及により、ＮＥＣを中心とするＭＳ-ＤＯＳ系の計算機がその地位を失い、今ではＭＳ-ＤＯＳ系の計算機はその姿を消しつつある。

本研究ではＭＳ-ＤＯＳ系計算機を用いたプログラム開発を行って北。しかし、これまでの研究成果を今後も有効に利用するためには、これらを商業主義に左右されないＯＳに基盤を置くワークステーションに移すことが重要な課題となってきた。また、本研究で提案している臨界安全監視システムの設計基本思想は高信頼性、耐雑音性、そして高拡張性にある。近年エンジニアリングワークステーションの性能は飛躍的に向上し、且つ価格もかなり安くなっており、この様にめざましい発展を遂げている計算機に着目すると、信号解析、表示、警報信号発生に、それらのアルゴリズムをＲＯＭ化したＩＣチップを用いるよりも、高性能の計算機を用意し、各種の信号解析ソフトウエアや、必要に応じて他のシステム同定アルゴリズムを組み合わせた総合的信号処理システムとすることが望ましい。この様な観点から、臨界安全監視用プログラムをワークステーションに移植することは非常に重要である。そこで、これまでＭＳ-ＤＯＳ系計算機上で開発された臨界安全監視用プログラムを順次年次計画を立ててワークステーション上に移植することとなった。平成１１年度は、臨界安全監視システムの中心的ソフトウエアである逐次型ＡＲＭＡモデル同定プログラム（ＡＲＭＡＸ１）の移植作業を行った。

なお、この報告書には逐次型ＡＲＭＡモデル同定アルゴリズムのプログラムりすと、並びにこのプログラムの使用マニュアルを添付した。

Appendix

A．適応ディジタルフィルタ（NEOADF）

A．1　プログラムの機能

ＮＥＯＡＤＦ．ＥＸＥは、計算条件を入力後、d_jを入力ファイル（２バイト／８バイトバイナリーデータ：原子炉より取得した中性子揺らぎのデータなど）から読み込んで、未知システム：$H(z)$のパラメータの推定値$(\hat{\phi}_{k,j},\hat{\theta}_{k,j})$と、ＡＤＦ：$G(z)$のパラメータの推定値$(\hat{a}_{k,j},\hat{b}_{k,j})$と、誤差$e_j$の分散$\sigma_e^{\,2}$を、出力ファイル（８バイトバイナリーデータ）に書き出し、必要であればそれとは別の出力ファイルに誤差e_jを書き出す。その他に計算条件が書かれたログファイルも出力する。

ＮＥＯＡＤＦ．ＥＸＥの出力ファイルはバイナリーであるが、変換プログラム（ＢＩＮ２ＣＳＶ．ＥＸＥ）を用いて、表計算ソフトやグラフ作成ソフトで利用可能な、カンマ区切りのＣＳＶ形式のＡＳＣＩＩテキストファイルに変換することができる。

A．2　計算条件の入力

’0.input binary type(2= 2byte_int, 8= 8byte_real) =’(bintyp : 2 or 8)

入力ファイルのデータが、２バイト整数バイナリーの時は、2 を、8バイト倍精度実数バイナリーの時は、8 を指定する。

’1.input binary data file name =’（i_file : 文字列)

入力ファイル名を、８文字以内＋ピリオド＋拡張子３文字という形式で指定する。

カレントディレクトリになければならない。

（全部で１２文字以内であれば、フルパスで指定することも可能。）

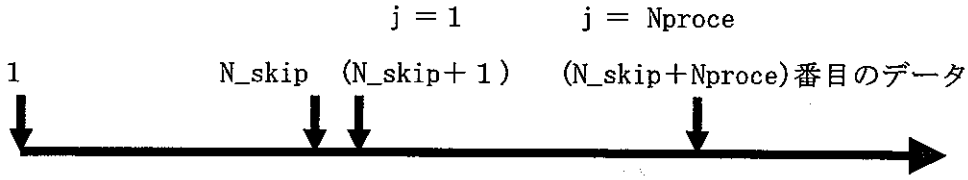
’2.number of data to be skipped =’（N_skip : 0～約21億の整数)

入力ファイルにおいて、１～N_skip 番目までのデータは使わずに読み飛ばす。

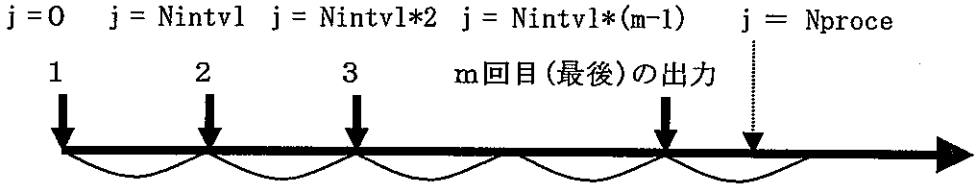
0 を指定すると、１番目のデータから計算に使われる。（図Ａ．１参照）

' 3. number of data to be processed =' (Nproce : 0～約21億の整数)

入力ファイルにおいて、(N_skip+1)～(N_skip+Nproce)番目の Nproce個のデータを計算に用いる。もちろん、(N_skip+Nproce)番目のデータが入力ファイルに存在しなければならない。（図Ａ．１、図Ａ．２参照）



図Ａ．１　入力ファイル(d_j)



図Ａ．２　データ出力の間隔

' 4. output data file name(8-byte real number binary) =' (o_file : 文字列)

出力ファイル名を、８文字以内＋ピリオド＋拡張子３文字という形式で指定する。既存のファイル名ではいけない。カレントディレクトリに作成される。

（全部で１２文字以内であれば、フルパスで指定することも可能。）

' 5. output interval of estimated parameters =' (Nintvl : 0～約21億の整数)

出力ファイルに、データを出力する間隔を指定する。1であれば、すべて出力できる。出力は、１回目にｊ＝０の時の初期値が出力され、その後、Nintvl回ごとに、j=Nproce以下の範囲で、くり返される。（図Ａ．２参照）

' 6. p(AR order of unknown system) =' (p : 0～127の整数)

未知システムをＡＲＭＡ(p,q)、または、ＡＲ(p)過程とした時の同定次数 pを指定する。

' 7. q(MA order of unknown system) =' (q : 0～127の整数)

未知システムをＡＲＭＡ(p,q)、または、ＭＡ(q)過程とした時の同定次数 qを指定する。ｐ＝ｑ＝0 であってはならない。

' 8.algorithm_1(0=White,1=Feintuch) =' (algo_1： 0 or 1)

$(\alpha_{k,j},\beta_{k,j})$を求めるアルゴリズムを指定する。0 のときは、Ｗｈｉｔｅのアルゴリズムを、1 のときは、Ｆｅｉｎｔｕｃｈのアルゴリズムを使う。

' 9.algorithm_2(0=White,1=Stochastic,2=Variable) =' (algo_2： 0 or 1 or 2)

修正係数(u_j,v_j)を求めるアルゴリズムを指定する。0 のときは、Ｗｈｉｔｅのアルゴリズムを、1 のときは、確率近似アルゴリズム(stochastic approximation algorithm)を、2 のときは、本研究で提案した variable gain constant algorithm を使う。(2.3.3, 2.3.4 参照)

'10.u0(adaptive gain constant for a[j]) ='

'11.v0(adaptive gain constant for b[j]) =' (v0 ＞0：正の倍精度実数)

このメニューは、9.でＷｈｉｔｅのアルゴリズムを指定した場合に現われる。このプログラムでは、u0＝v0としているので、v0のみ指定する。Ｗｈｉｔｅのアルゴリズムでは、u_j＝u0＝u＝一定、v_j＝v0＝v＝一定である。v0 は、収束条件をみたす最大値v_{max}より小さな値でないといけない。式 (2.42) , (2.43) より

$$v_{max}=\frac{2}{(N+1)\sigma_d^{\,2}} \qquad \text{(ARMA(p,q) } p\geqq q \text{ の時 N+1=p、} p<q \text{の時 N+1=q)}$$

ただし、$\sigma_d^{\,2}=\frac{1}{Nproce}\sum_{j=1}^{Nproce} d_j^{\,2}$ (A.1)

'12.k1(v[j]=k1*v0/(c1+j)) =' (k1 ＞0：正の倍精度実数)

このメニューは、9.で確率近似アルゴリズムを指定した場合に現われる。ここでv0 は、収束条件をみたす最大値v_{max}を計算した値が代入されている。 v[j=1]がv_{max}より小さくなるように、k1,c1を指定しなければならない。

$$v_j=\frac{k1*v0}{j+c1} \qquad \text{(A.2)}$$

'13.c1(v[j]=k1*v0/(c1+j)) =' (c1 ＞0：正の倍精度実数)

このメニューは、9.で確率近似アルゴリズムを指定した場合に現われる。

'14. c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =' (c2 >0：正の倍精度実数)

このメニューは、9.でvariable gain constant algorithmを指定した場合に現われる。

v[j=1]が v_{max} より小さくなるように、c2, r2, a2を指定しなければならない。

$$v_j = \frac{v0}{c2 + r2\left(1 - e^{-a2*j}\right)} \qquad (A.3)$$

'15. r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =' (r2 >0：正の倍精度実数)

このメニューは、9.でvariable gain constant algorithmを指定した場合に現われる。

1より大きな値でなければならない。(例、1.0D+4)

'16. a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =' (a2 >0：正の倍精度実数)

このメニューは、9.でvariable gain constant algorithmを指定した場合に現われる。

1より小さな値でなければならない。(例、1.0D-4)

'17. flag of the output e_file(0=No, 1=Yes) =' (e_mode： 0 or 1)

誤差 e_j をファイルに出力するかどうかを指定する。0 のときは、出力しない。

1 のときは、出力する。

'18. output e_file name(8-byte real number binary) = (e_file：文字列)

このメニューは、17.で 1(=Yes)を指定した場合に現われる。誤差 e_j を出力するファイル名を、８文字以内＋ピリオド＋拡張子３文字という形式で指定する。既存のファイル名ではいけない。カレントディレクトリに作成される。

（全部で１２文字以内であれば、フルパスで指定することも可能。）

A．３　プログラムの実行例

以下に、例を示す。入力する部分は下線部で表している。実際には入力部分の最後には、Ｅｎｔｅｒキーを押している。※にはコメントを書いている。

```
B:¥ADF>NEOADF
System Identification by Using ADF (Adaptive Digital Filter)
*** Please enter the next ***
```

```
 0. input binary type(2= 2byte_int,  8= 8byte_real) =
2
 1. input binary data file name =
id010.800
 2. number of data to be skipped =
0
 3. number of data to be processed =
150000
 4. output data file name(8-byte real number binary) =
test.bin
 5. output interval of estimated parameters =
10000
 6. p(AR order of unknown system) =
2
 7. q(MA order of unknown system) =
2
 8. algorithm_1(0=White, 1=Feintuch) =
0
 9. algorithm_2(0=White, 1=Stochastic, 2=Variable) =
0
 ### We recommend u0, v0<  3.803133336156471E-008                ※ v_max
10. u0(adaptive gain constant for a[j]) =
11. v0(adaptive gain constant for b[j]) =
5.0D-12
17. flag of the output e_file(0=No, 1=Yes) =
1
18. output e_file name(8-byte real number binary) =
test-e.bin                          ※ 17. で、0 とした場合は表示されない。

*** the conditions of this calculation ***
 0. input binary type(2= 2byte_int,  8= 8byte_real) =          2
 1. input binary data file name =id010.800
```

```
 2. number of data to be skipped =          0
 3. number of data to be processed =     150000
 4. output data file name(8-byte real number binary) =test.bin
 ### output file size[byte] =       1152
 5. output interval of estimated parameters =      10000
 6. p(AR order of unknown system) =          2
 7. q(MA order of unknown system) =          2
 8. algorithm_1(0=White, 1=Feintuch) =          0
 9. algorithm_2(0=White, 1=Stochastic, 2=Variable) =          0
10. u0(adaptive gain constant for a[j]) =  5.000000000000000E-012
11. v0(adaptive gain constant for b[j]) =  5.000000000000000E-012
 ### We recommend u0, v0<  3.803133336156471E-008                ※ v_max
17. flag of the output e_file(0=No, 1=Yes) =          1
18. output e_file name(8-byte real number binary) =test-e.bin
Please enter 0-18(correct), s(start), or q(quit)
s                          ※ 0～18のうち修正したい項目の番号を指定すると、
                                                    修正できる。
*** start this calculation ***
 ### log file name: test.LOG
*** ordinary termination ***
Stop - Program terminated.
```

A．4　ログファイルの形式

先の例では、test.logというファイルが出来た。その内容は次のようになっている。ログファイル名は、出力ファイル名test.binの拡張子をlogに変えたものとなる。

```
             2  #+ input binary type(2= 2byte_int, 8= 8byte_real)
id010.800       #+ input data file name
             0  #+ number of data to be skipped
        150000  #+ number of data to be processed
test.bin        #+ output data file name(8-byte real number binary)
         10000  #+ output interval of estimated parameters
             2  #+ p(AR order of unknown system)
```

```
             2  #+ q(MA order of unknown system)
             0  #+ algorithm_1(0=White,1=Feintuch)
             0  #+ algorithm_2(0=White,1=Stochastic,2=Variable)
 5.00000E-12  #+ u0(adaptive gain constant for a[j])
             1  #+ flag of the output e_file(0=No,1=Yes)
test-e.bin    #+ output e_file name(8-byte real number binary)
s             #+ for input re-direction (< filename.ini)
 5.00000E-12  #- v0(adaptive gain constant for b[j])
 3.80313E-08  #- u_max,v_max
             9  #- number of output data par record
            16  #- number of output records
          1152  #- output file size[byte]
a[1] a[2]
b[1] b[2]
phi[1] phi[2]
theta[1] theta[2]
e_var         #- output record form
```

ログファイルの、“s　　#+ for input re-direction (< filename.ini)”という行以前には、１４桁目から #+という記号があり、その行より下は、#-と書かれている。#+は、1～１２桁目までに、計算条件として用いた値が書かれている。#-は、付加的な情報である。出力バイナリーファイルには、$a_{1,j}\ a_{2,j}\ b_{1,j}\ b_{2,j}\ \phi_{1,j}\ \phi_{2,j}\ \theta_{1,j}\ \theta_{2,j}\ \sigma_e^{\,2}$ という順番で、9 (#- number of output data par record) 個のデータが、j＝0の時、j＝10000の時、j＝20000の時、……、j＝150000の時、の 16 (#- number of output records) 回出力されて、出力ファイルサイズは、8バイト×9×16＝1152(#- output file size[byte])バイトとなったことが分かる。

ログファイルの名前を、test.iniに変えて、入力リダイレクションを使って、

B:¥ADF>NEOADF < test.ini

としても実行できる。この場合、実行時にいちいち計算条件を手で入力する必要がなくなり便利である。これをバッチファイルに書いて複数の条件で計算させていくということもできる。ただし、１～１２桁以内に値がちゃんと書かれている必要がある。

条件によってログファイルの行数が変わる。

```
                1  #+ flag of the output e_file(0=No, 1=Yes)
    test-e.bin  #+ output e_file name(8-byte real number binary)
    s           #+ for input re-direction (< filename.ini)
```

という行は、0=Noのときは、次のようになる。

```
                0  #+ flag of the output e_file(0=No, 1=Yes)
    s           #+ for input re-direction (< filename.ini)
```

また、algorithm_2の行以下も、0 (=White)のときは次のようになるが、

```
                0  #+ algorithm_2(0=White, 1=Stochastic, 2=Variable)
      5.00000E-12  #+ u0(adaptive gain constant for a[j])
                0 #+ flag of the output e_file(0=No, 1=Yes)
```

1 (=Stochastic)のときは、次のようになり、

```
                1  #+ algorithm_2(0=White, 1=Stochastic, 2=Variable)
         1.0000    #+ k1(v[j]=k1*v0/(c1+j))
         11.000    #+ c1(v[j]=k1*v0/(c1+j))
                0  #+ flag of the output e_file(0=No, 1=Yes)
```

2 (=Variable)のときは、次のようになるので注意が必要である。

```
                2  #+ algorithm_2(0=White, 1=Stochastic, 2=Variable)
         10.000    #+ c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))
         10000.    #+ r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))
      1.00000E-04  #+ a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))
                0  #+ flag of the output e_file(0=No, 1=Yes)
```

A．5　出力ファイルの形式

ＡＲ(ｐ)の場合は、

b[1]--b[p] phi[1]--phi[p] e_var

すなわち、 $b_{1,j}--b_{p,j}\,\phi_{1,j}--\phi_{p,j}\,\sigma_e^2$

ＡＲＭＡ(ｐ，ｑ)、ｐ≧ｑ の場合は、

a[1]--a[q] b[1]--b[p] phi[1]--phi[p] theta[1]--theta[q] e_var

すなわち、$a_{1,j}--a_{q,j}\,b_{1,j}--b_{p,j}\,\phi_{1,j}--\phi_{p,j}\,\theta_{1,j}--\theta_{q,j}\,\sigma_e^2$

ＭＡ(ｑ)の場合は、

a[1]--a[q] b[1]--b[q] theta[1]--theta[q] e_var

すなわち、$a_{1,j}--a_{q,j}\,b_{1,j}--b_{q,j}\,\theta_{1,j}--\theta_{q,j}\,\sigma_e^{\,2}$

ＡＲＭＡ(ｐ，ｑ)、ｐ＜ｑ の場合は、

a[1]--a[q] b[1]--b[q] phi[1]--phi[p] theta[1]--theta[q] e_var

すなわち、$a_{1,j}--a_{q,j}\,b_{1,j}--b_{q,j}\,\phi_{1,j}--\phi_{p,j}\,\theta_{1,j}--\theta_{q,j}\,\sigma_e^{\,2}$

という順番で格納された８バイトバイナリーファイルである。ここで

$$\sigma_e^{\,2}=\frac{1}{N\,intvl}\sum_{k=1}^{N\,intvl}e_k^{\,2} \qquad (A.4)$$

であり、Nintvl回ごとに誤差e_jの二乗平均を計算したものである。

Ａ．６　ＣＳＶテキスト形式への変換

先程の例では、ログファイル（test.log）と出力ファイル（test.bin）と、誤差の出力ファイル（test-e.bin）が得られた。これから、ｂｉｎ２ｃｓｖ．ｅｘｅを用いて、ＡＳＣＩＩテキストファイル（test.csv）、（test-e.csv）に変換する例を示す。入力する部分は下線部で表している。

```
B:¥ADF>bin2csv
bin2csv (output binary to csv-text file converter)
*** Please enter the next ***
 1.input log file name =
test.log
 2.output csv-text file name =
test.csv
 3.target(0=o_file,1=e_file) =
0                          ※ 0 を選ぶと、出力ファイルを変換
                              1 を選ぶと、誤差の出力ファイルを変換する。
*** the conditions of the file conversion ***
 1.input log file name =test.log
 2.output csv-text file name =test.csv
 3.target(0=o_file,1=e_file) =          0
Please enter 1-3(correct), n(next-step), or q(quit)
n
```

```
*** next step ***                    ※ ログファイルから読み込んだ情報の表示
Nproce =       150000
o_file = test.bin
Nintvl =        10000
p =             2
q =             2
algo_2 =            0
e_mode =              1
e_file = test-e.bin

*** Please enter the next ***        ※ 出力ファイルの場合のみ
### j,phi[1],--,phi[p],theta[1],--,theta[q],a[1],--,a[q],b[1],--,b[p],e_var
 4.flag[phi,theta](1=yes,0=no) =     ※ (φ̂k,j, θ̂k,j)を出力するかどうか
1                                        を指定。1 にすると出力する。
 5.flag[a,b](1=yes,0=no) =           ※ (âk,j, b̂k,j)を出力するかどうか
1                                        を指定。1 にすると出力する。
 6.flag[e_var](1=yes,0=no) =         ※ σe² を出力するかどうかを指定。
1                                        1 にすると出力する。

*** the conditions of the file conversion ***
 4.flag[phi,theta](1=yes,0=no) =             1
 5.flag[a,b](1=yes,0=no) =           1
 6.flag[e_var](1=yes,0=no) =             1
Please enter 4-6(correct), s(start), or q(quit)
s

*** start this calculation ***
*** ordinary termination ***
Please enter r(re-try), or q(quit)
```

q

Stop - Program terminated.

以上の操作によりできたＣＳＶテキストファイルは、次のようになる。

Test.csv

j,phi(1),phi(2),theta(1),theta(2),a(1),a(2),b(1),b(2),e_var

0, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00, 0.0000E+00

10000, 4.0791E-01, 3.8137E-01,-9.8960E-02,-1.0596E-01, 9.8960E-02, 1.0596E-01, 3.0895E-01, 2.7541E-01, 1.6657E+07

（途中省略）

150000, 4.7829E-01, 4.3423E-01,-1.5154E-01,-2.4049E-01, 1.5154E-01, 2.4049E-01, 3.2675E-01, 1.9374E-01, 1.3590E+07

test-e.csv

j,e_var

0, 0.0000E+00

1,-3.8920E+03

2,-2.3910E+03

3,-8.8189E+02

（以下省略）

次のようなファイル（test-csv.ini）を作っておくと、入力リダイレクションを使って、

B:¥ADF>bin2csv < test-csv.ini

としても実行できる。test-csv.iniの内容は、次のようになっている。

test.log

test.csv

0

n

1

1

0

s

q

A．７　プログラムソースリスト

ここでは、ＮＥＯＡＤＦ．ＥＸＥのＦＯＲＴＲＡＮソースファイル(NEOADF.FOR)の内容を示す。基本的には、ＦＯＲＴＲＡＮ７７で書かれてあるが、多少その範囲を超える機能も使っている。たとえば、小文字、サブルーチン・関数名にアンダーバー(_)の使用、DOUBLE PRECISIONの代わりにREAL*8という宣言の使用、あるいは変数の宣言漏れが無いかを確かめるため、IMPLICIT NONE文を用いている。

COMMON文は使用せず、サブルーチン・関数間のやり取りには、全て引数を割り当てている。これは、Ｃ言語への移植を考慮したためであるが、ＦＯＲＴＲＡＮ７７にはＣ言語のようなグローバル変数がないという理由もある。

なお、変数の名称としては、それぞれcnt(=Count, Counter)、f_*(=関数)、pos(=Position)、buf*(=Buffer)、N*(=Number)、*_file(=File name)、*_cal,*_c(=Calculation)、ev*, e_var(Variance of error)というような意味で使っている。（＊は任意の文字を表すワイルドカード)

配列変数の添え字は、理論式と次のように対応させてある。（Ａｐｐｅｎｄｉｘ Ｂ参照）

ϕ_k ＝ phi(k)

θ_k ＝ theta(k)

d_j ＝ d(0)	d_{j-k} ＝ d(k)
e_j ＝ e(0)	e_{j-k} ＝ e(k)
y_j ＝ y(0)	y_{j-k} ＝ y(k)
$\alpha_{k,j}$ ＝ alpha(k,0)	$\alpha_{k,j-i}$ ＝ alpha(k,i)
$\beta_{k,j}$ ＝ beta(k,0)	$\beta_{k,j-i}$ ＝ beta(k,i)
$a_{k,j}$ ＝ a(k)	$a_{k,j+1}$ ＝ a_next(k)
$b_{k,j}$ ＝ b(k)	$b_{k,j+1}$ ＝ b_next(k)
j ＝ Ncount	σ_e^2 ＝ e_var

次頁以降にプログラムソースリストを載せておく。

NEOADF.FOR ソースリスト

```
************************************************************************
      PROGRAM NEOADF
************************************************************************
      IMPLICIT NONE
      CHARACTER*12 i_file,o_file,e_file
      INTEGER*4 N_skip,Nproce,Nintvl
      INTEGER*1 p,q,algo_1,algo_2,bintyp,e_mode
      REAL*8 u0,v0,k1,c1,c2,r2,a2

      CALL input(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,algo_1,
     $         algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)
      CALL main_c(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,algo_1,
     $         algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)
      CALL loglog(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,algo_1,
     $         algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)

      WRITE(6,*) '*** ordinary termination ***'

      STOP
      END


************************************************************************
      SUBROUTINE input(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,
     $               algo_1,algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)
************************************************************************
      IMPLICIT NONE
      CHARACTER*12 i_file,o_file,e_file
      INTEGER*4 N_skip,Nproce,Nintvl
      INTEGER*1 p,q,algo_1,algo_2,maxdim,bintyp,e_mode
      REAL*8 u0,v0,k1,c1,c2,r2,a2

      REAL*8 fmaxuv
      CHARACTER*2 flag
      INTEGER*4 f_leng,f_Nrec
      INTEGER*1 i

      PARAMETER (maxdim=127)


      WRITE(6,*) 'System Identification by Using ADF (Adaptive Digital F
     $ilter)'
      WRITE(6,*) '*** Please enter the next ***'

      WRITE(6,*) ' 0.input binary type(2= 2byte_int, 8= 8byte_real) ='
      READ(5,*) bintyp
      WRITE(6,*) ' 1.input binary data file name ='
      READ(5,'(A)') i_file
      WRITE(6,*) ' 2.number of data to be skipped ='
      READ(5,*) N_skip
      WRITE(6,*) ' 3.number of data to be processed ='
      READ(5,*) Nproce
      WRITE(6,*) ' 4.output data file name(8-byte real number binary) ='
      READ(5,'(A)') o_file
      WRITE(6,*) ' 5.output interval of estimated parameters ='
      READ(5,*) Nintvl
      WRITE(6,*) ' 6.p(AR order of unknown system) ='
      READ(5,*) p
      WRITE(6,*) ' 7.q(MA order of unknown system) ='
      READ(5,*) q
      WRITE(6,*) ' 8.algorithm_1(0=White,1=Feintuch) ='
      READ(5,*) algo_1
      WRITE(6,*) ' 9.algorithm_2(0=White,1=Stochastic,2=Variable) ='
      READ(5,*) algo_2
      IF (algo_2.EQ.0) THEN

         WRITE(6,*) ' ### We recommend u0,v0<',fmaxuv(bintyp,i_file,
     $              N_skip,Nproce,p,q)

         WRITE(6,*) '10.u0(adaptive gain constant for a[j]) ='
         WRITE(6,*) '11.v0(adaptive gain constant for b[j]) ='
         READ(5,*) v0
         u0=v0
      ELSEIF (algo_2.EQ.1) THEN
         v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
         u0=v0
         WRITE(6,*) '12.k1(v[j]=k1*v0/(c1+j)) ='
         READ(5,*) k1
```

```
        WRITE(6,*) '13.c1(v[j]=k1*v0/(c1+j)) ='
        READ(5,*) c1
      ELSEIF (algo_2.EQ.2) THEN
        v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
        u0=v0
        WRITE(6,*) '14.c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
        READ(5,*) c2
        WRITE(6,*) '15.r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
        READ(5,*) r2
        WRITE(6,*) '16.a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
        READ(5,*) a2
      ENDIF

      WRITE(6,*) '17.flag of the output e_file(0=No,1=Yes) ='
      READ(5,*) e_mode
      e_file=' '
      IF (e_mode.EQ.1) THEN
        WRITE(6,*) '18.output e_file name(8-byte real number binary) =
     $'
        READ(5,'(A)') e_file
      ENDIF

C     /* This Do Loop is for input re-direction ( < *.ini ) */
      DO 555 i=12,2,-1
        IF (i_file(i:i).EQ.'#') THEN
          i_file=i_file(1:i-1)
          GOTO 567
        ENDIF
  555 CONTINUE

  567 DO 666 i=12,2,-1
        IF (o_file(i:i).EQ.'#') THEN
          o_file=o_file(1:i-1)
          GOTO 678
        ENDIF
  666 CONTINUE
  678 DO 777 i=12,2,-1
        IF (e_file(i:i).EQ.'#') THEN
          e_file=e_file(1:i-1)
          GOTO 111
        ENDIF
  777 CONTINUE

C     /* start of infinite loop */
  111 CONTINUE
      IF (.TRUE.) THEN
        WRITE(6,*)
        WRITE(6,*) '*** the conditions of this calculation ***'
        WRITE(6,*) ' 0.input binary type(2= 2byte_int, 8= 8byte_real)
     $=',bintyp
        WRITE(6,*) ' 1.input binary data file name =',i_file
        WRITE(6,*) ' 2.number of data to be skipped =',N_skip
        WRITE(6,*) ' 3.number of data to be processed =',Nproce
        WRITE(6,*) ' 4.output data file name(8-byte real number binary
     $) =',o_file

        WRITE(6,*) ' ### output file size[byte] =',f_leng(p,q)*8*
     $               f_Nrec(Nproce,Nintvl)

        WRITE(6,*) ' 5.output interval of estimated parameters ='
     $,Nintvl
        WRITE(6,*) ' 6.p(AR order of unknown system) =',p
        WRITE(6,*) ' 7.q(MA order of unknown system) =',q
        WRITE(6,*) ' 8.algorithm_1(0=White,1=Feintuch) =',algo_1
        WRITE(6,*) ' 9.algorithm_2(0=White,1=Stochastic,2=Variable) ='
     $              ,algo_2
        WRITE(6,*) '10.u0(adaptive gain constant for a[j]) =',u0
        WRITE(6,*) '11.v0(adaptive gain constant for b[j]) =',v0
        IF (algo_2.EQ.0) THEN
          WRITE(6,*) ' ### We recommend u0,v0<',fmaxuv(bintyp,
     $                i_file,N_skip,Nproce,p,q)
        ELSEIF (algo_2.EQ.1) THEN
          WRITE(6,*) ' ### v[j=1]=',k1*v0/(c1+1.D0),
     $               '   v[j=10^6]=',k1*v0/(c1+1.D6)
          WRITE(6,*) '12.k1(v[j]=k1*v0/(c1+j)) =',k1
          WRITE(6,*) '13.c1(v[j]=k1*v0/(c1+j)) =',c1
        ELSEIF (algo_2.EQ.2) THEN
          WRITE(6,*) ' ### v[j=1]=',v0/(c2+r2*(1.D0-DEXP(-a2*1.D0)))
     $,              '   v[j> 1/a2]=',v0/r2
          WRITE(6,*) '14.c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =',c2
          WRITE(6,*) '15.r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =',r2
```

```
            WRITE(6,*) '16.a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) =',a2
         ENDIF

         WRITE(6,*) '17.flag of the output e_file(0=No,1=Yes) =',e_mode
         IF (e_mode.EQ.1) THEN
            WRITE(6,*) '18.output e_file name(8-byte real number binar
     $y) =',e_file
         ENDIF

C         CALL check

         WRITE(6,*) 'Please enter 0-18(correct), s(start), or q(quit)'
         READ(5,'(A)') flag
         IF (flag.EQ.'q'.OR.flag.EQ.'Q') THEN
            WRITE(6,*)
            WRITE(6,*) '*** quit this program ***'
            STOP
         ELSEIF (flag.EQ.'s'.OR.flag.EQ.'S') THEN
            WRITE(6,*)
            WRITE(6,*) '*** start this calculation ***'
            RETURN

         ELSEIF (flag.EQ.'0'.OR.flag.EQ.'1') THEN
            WRITE(6,*) ' 0.input binary type(2= 2byte_int, 8= 8byte_re
     $al) ='
            READ(5,*) bintyp
            WRITE(6,*) ' 1.input binary data file name ='
            READ(5,'(A)') i_file
            IF (algo_2.EQ.1.OR.algo_2.EQ.2) THEN
               v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
               u0=v0
            ENDIF
         ELSEIF (flag.EQ.'2') THEN
            WRITE(6,*) ' 2.number of data to be skipped ='
            READ(5,*) N_skip
            IF (algo_2.EQ.1.OR.algo_2.EQ.2) THEN
               v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
               u0=v0
            ENDIF
         ELSEIF (flag.EQ.'3') THEN
            WRITE(6,*) ' 3.number of data to be processed ='
            READ(5,*) Nproce
         ELSEIF (flag.EQ.'4') THEN
            WRITE(6,*) ' 4.output data file name(8-byte real number bi
     $nary) ='
            READ(5,'(A)') o_file
         ELSEIF (flag.EQ.'5') THEN
            WRITE(6,*) ' 5.output interval of estimated parameters ='
            READ(5,*) Nintvl
         ELSEIF (flag.EQ.'6') THEN
            WRITE(6,*) ' 6.p(AR order of unknown system) ='
            READ(5,*) p
            IF (p.EQ.0.AND.q.EQ.0) THEN
               q=1
            ENDIF
            IF (algo_2.EQ.1.OR.algo_2.EQ.2) THEN
               v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
               u0=v0
            ENDIF
         ELSEIF (flag.EQ.'7') THEN
            WRITE(6,*) ' 7.q(MA order of unknown system) ='
            READ(5,*) q
            IF (p.EQ.0.AND.q.EQ.0) THEN
               p=1
            ENDIF
            IF (algo_2.EQ.1.OR.algo_2.EQ.2) THEN
               v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
               u0=v0
            ENDIF
         ELSEIF (flag.EQ.'8') THEN
            WRITE(6,*) ' 8.algorithm_1(0=White,1=Feintuch) ='
            READ(5,*) algo_1
         ELSEIF (flag.EQ.'9') THEN
            WRITE(6,*) ' 9.algorithm_2(0=White,1=Stochastic,2=Variable
     $) ='
            READ(5,*) algo_2

            IF (algo_2.EQ.0) THEN

               WRITE(6,*) ' ### We recommend u0,v0<',fmaxuv(bintyp,
     $                    i_file,N_skip,Nproce,p,q)
```

```
            WRITE(6,*) '10.u0(adaptive gain constant for a[j]) ='
            WRITE(6,*) '11.v0(adaptive gain constant for b[j]) ='
            READ(5,*) v0
            u0=v0
         ELSEIF (algo_2.EQ.1) THEN
            v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
            u0=v0
            WRITE(6,*) '12.k1(v[j]=k1*v0/(c1+j)) ='
            READ(5,*) k1
            WRITE(6,*) '13.c1(v[j]=k1*v0/(c1+j)) ='
            READ(5,*) c1
         ELSEIF (algo_2.EQ.2) THEN
            v0=fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
            u0=v0
            WRITE(6,*) '14.c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) c2
            WRITE(6,*) '15.r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) r2
            WRITE(6,*) '16.a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) a2
         ENDIF

      ELSEIF (flag.EQ.'10'.OR.flag.EQ.'11') THEN
         IF (algo_2.EQ.0) THEN
            WRITE(6,*) ' ### We recommend u0,v0<',fmaxuv(bintyp,
     $                  i_file,N_skip,Nproce,p,q)
            WRITE(6,*) '10.u0(adaptive gain constant for a[j]) ='
            WRITE(6,*) '11.v0(adaptive gain constant for b[j]) ='
            READ(5,*) v0
            u0=v0
         ENDIF
      ELSEIF (flag.EQ.'12') THEN
         IF (algo_2.EQ.1) THEN
            WRITE(6,*) '12.k1(v[j]=k1*v0/(c1+j)) ='
            READ(5,*) k1
         ENDIF
      ELSEIF (flag.EQ.'13') THEN
         IF (algo_2.EQ.1) THEN
            WRITE(6,*) '13.c1(v[j]=k1*v0/(c1+j)) ='
            READ(5,*) c1
         ENDIF
      ELSEIF (flag.EQ.'14') THEN
         IF (algo_2.EQ.2) THEN
            WRITE(6,*) '14.c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) c2
         ENDIF
      ELSEIF (flag.EQ.'15') THEN
         IF (algo_2.EQ.2) THEN
            WRITE(6,*) '15.r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) r2
         ENDIF
      ELSEIF (flag.EQ.'16') THEN
         IF (algo_2.EQ.2) THEN
            WRITE(6,*) '16.a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j))))) ='
            READ(5,*) a2
         ENDIF

      ELSEIF (flag.EQ.'17'.OR.flag.EQ.'18') THEN
         WRITE(6,*) '17.flag of the output e_file(0=No,1=Yes) ='
         READ(5,*) e_mode
         e_file=' '
         IF (e_mode.EQ.1) THEN
            WRITE(6,*) '18.output e_file name(8-byte real number b
     $inary) ='
            READ(5,'(A)') e_file
         ENDIF

      ELSE

      ENDIF

      GOTO 111
      ENDIF
C     /* end of infinite loop */


      RETURN
      END
```

```
************************************************************************
      INTEGER*4 FUNCTION f_leng(p,q)
************************************************************************
      IMPLICIT NONE
      INTEGER*1 p,q

      IF (p.GE.q) THEN
         IF (q.EQ.0) THEN
C           /* when AR(p) */
            f_leng=2*p+1
         ELSE
C           /* when ARMA(p,q) p>=q */
            f_leng=2*p+2*q+1
         ENDIF
      ELSE
         IF (p.EQ.0) THEN
C           /* when MA(q) */
            f_leng=3*q+1
         ELSE
C           /* when ARMA(p,q) p<q */
            f_leng=3*q+p+1
         ENDIF
      ENDIF

      RETURN
      END


************************************************************************
      INTEGER*4 FUNCTION f_Nrec(j,Nintvl)
************************************************************************
      IMPLICIT NONE
      INTEGER*4 j,Nintvl

      f_Nrec=(j/Nintvl)+1

      RETURN
      END


************************************************************************
      REAL*8 FUNCTION fmaxuv(bintyp,i_file,N_skip,Nproce,p,q)
************************************************************************
      IMPLICIT NONE
      CHARACTER*12 i_file
      INTEGER*4 N_skip,Nproce
      INTEGER*1 p,q,bintyp

      INTEGER*4 i,pos
      INTEGER*2 i_temp
      REAL*8 d_temp,sum

      sum=0.0D0

      OPEN(UNIT=1,FILE=i_file,ACCESS='DIRECT',RECL=bintyp,ERR=777,
     $     STATUS='OLD')

      DO 111 i=1,Nproce
         pos=N_skip+i
         IF (bintyp.EQ.2) THEN
            READ(1,REC=pos) i_temp
            d_temp=i_temp*1.0D0
         ELSEIF (bintyp.EQ.8) THEN
            READ(1,REC=pos) d_temp
         ENDIF
         sum=sum+d_temp*d_temp
 111  CONTINUE

      CLOSE(1)

      IF (p.GE.q) THEN
C        /* when AR(p) or ARMA(p,q) p>=q */
         fmaxuv=2.0D0/(p*sum/Nproce)
      ELSE
C        /* when MA(q) or ARMA(p,q) p<q*/
         fmaxuv=2.0D0/(q*sum/Nproce)
      ENDIF

      RETURN

 777  WRITE(6,*) ' ### error: can''t open input data file'
```

```
      STOP

      END

************************************************************************
      SUBROUTINE form(p,q)
************************************************************************
      IMPLICIT NONE
      INTEGER*1 p,q

      IF (p.GE.q) THEN
         IF (q.EQ.0) THEN
C           /* when AR(p) */
C            'b[1] --- b[p]'
C            'phi[1] --- phi[p]'
C            'e_var'
            IF (p.EQ.1) THEN
               WRITE(3,'(A)') 'b[1]'
               WRITE(3,'(A)') 'phi[1]'
            ELSEIF (p.EQ.2) THEN
               WRITE(3,'(A)') 'b[1] b[2]'
               WRITE(3,'(A)') 'phi[1] phi[2]'
            ELSEIF (p.GE.3 .AND. p.LE.9) THEN
               WRITE(3,'(A,I1,A)') 'b[1] --- b[',p,']'
               WRITE(3,'(A,I1,A)') 'phi[1] --- phi[',p,']'
            ELSEIF (p.GE.10 .AND. p.LE.99) THEN
               WRITE(3,'(A,I2,A)') 'b[1] --- b[',p,']'
               WRITE(3,'(A,I2,A)') 'phi[1] --- phi[',p,']'
            ELSE
               WRITE(3,'(A)') 'b[1] --- b[p]'
               WRITE(3,'(A)') 'phi[1] --- phi[p]'
            ENDIF

            WRITE(3,'(A)') 'e_var        #- output record form'

         ELSE
C           /* when ARMA(p,q) p>=q */
C           'a[1]--a[q]'
C           'b[1]--b[p]'
C           'phi[1]--phi[p]'
C           'theta[1]--theta[q]'
C           'e_var'
            IF (q.EQ.1) THEN
               WRITE(3,'(A)') 'a[1]'
            ELSEIF (q.EQ.2) THEN
               WRITE(3,'(A)') 'a[1] a[2]'
            ELSEIF (q.GE.3 .AND. q.LE.9) THEN
               WRITE(3,'(A,I1,A)') 'a[1] --- a[',q,']'
            ELSEIF (q.GE.10 .AND. q.LE.99) THEN
               WRITE(3,'(A,I2,A)') 'a[1] --- a[',q,']'
            ELSE
               WRITE(3,'(A)') 'a[1] --- a[q]'
            ENDIF

            IF (p.EQ.1) THEN
               WRITE(3,'(A)') 'b[1]'
               WRITE(3,'(A)') 'phi[1]'
            ELSEIF (p.EQ.2) THEN
               WRITE(3,'(A)') 'b[1] b[2]'
               WRITE(3,'(A)') 'phi[1] phi[2]'
            ELSEIF (p.GE.3 .AND. p.LE.9) THEN
               WRITE(3,'(A,I1,A)') 'b[1] --- b[',p,']'
               WRITE(3,'(A,I1,A)') 'phi[1] --- phi[',p,']'
            ELSEIF (p.GE.10 .AND. p.LE.99) THEN
               WRITE(3,'(A,I2,A)') 'b[1] --- b[',p,']'
               WRITE(3,'(A,I2,A)') 'phi[1] --- phi[',p,']'
            ELSE
               WRITE(3,'(A)') 'b[1] --- b[p]'
               WRITE(3,'(A)') 'phi[1] --- phi[p]'
            ENDIF

            IF (q.EQ.1) THEN
               WRITE(3,'(A)') 'theta[1]'
            ELSEIF (q.EQ.2) THEN
               WRITE(3,'(A)') 'theta[1] theta[2]'
            ELSEIF (q.GE.3 .AND. q.LE.9) THEN
               WRITE(3,'(A,I1,A)') 'theta[1] --- theta[',q,']'
            ELSEIF (q.GE.10 .AND. q.LE.99) THEN
               WRITE(3,'(A,I2,A)') 'theta[1] --- theta[',q,']'
            ELSE
```

```
            WRITE(3,'(A)') 'theta[1] --- theta[q]'
         ENDIF

         WRITE(3,'(A)') 'e_var       #- output record form'

      ENDIF
   ELSE
      IF (p.EQ.0) THEN
C        /* when MA(q) */
C        'a[1]--a[q]'
C        'b[1]--b[q]'
C        'theta[1]--theta[q]'
C        'e_var'
         IF (q.EQ.1) THEN
            WRITE(3,'(A)') 'a[1]'
            WRITE(3,'(A)') 'b[1]'
            WRITE(3,'(A)') 'theta[1]'
         ELSEIF (q.EQ.2) THEN
            WRITE(3,'(A)') 'a[1] a[2]'
            WRITE(3,'(A)') 'b[1] b[2]'
            WRITE(3,'(A)') 'theta[1] theta[2]'
         ELSEIF (q.GE.3 .AND. q.LE.9) THEN
            WRITE(3,'(A,I1,A)') 'a[1] --- a[',q,']'
            WRITE(3,'(A,I1,A)') 'b[1] --- b[',q,']'
            WRITE(3,'(A,I1,A)') 'theta[1] --- theta[',q,']'
         ELSEIF (q.GE.10 .AND. q.LE.99) THEN
            WRITE(3,'(A,I2,A)') 'a[1] --- a[',q,']'
            WRITE(3,'(A,I2,A)') 'b[1] --- b[',q,']'
            WRITE(3,'(A,I2,A)') 'theta[1] --- theta[',q,']'
         ELSE
            WRITE(3,'(A)') 'a[1] --- a[q]'
            WRITE(3,'(A)') 'b[1] --- b[q]'
            WRITE(3,'(A)') 'theta[1] --- theta[q]'
         ENDIF

         WRITE(3,'(A)') 'e_var       #- output record form'

      ELSE
C        /* when ARMA(p,q) p<q */
C        'a[1]--a[q]'
C        'b[1]--b[q]'
C        'phi[1]--phi[p]'
C        'theta[1]--theta[q]'
C        'e_var'
         IF (q.EQ.1) THEN
            WRITE(3,'(A)') 'a[1]'
            WRITE(3,'(A)') 'b[1]'
         ELSEIF (q.EQ.2) THEN
            WRITE(3,'(A)') 'a[1] a[2]'
            WRITE(3,'(A)') 'b[1] b[2]'
         ELSEIF (q.GE.3 .AND. q.LE.9) THEN
            WRITE(3,'(A,I1,A)') 'a[1] --- a[',q,']'
            WRITE(3,'(A,I1,A)') 'b[1] --- b[',q,']'
         ELSEIF (q.GE.10 .AND. q.LE.99) THEN
            WRITE(3,'(A,I2,A)') 'a[1] --- a[',q,']'
            WRITE(3,'(A,I2,A)') 'b[1] --- b[',q,']'
         ELSE
            WRITE(3,'(A)') 'a[1] --- a[q]'
            WRITE(3,'(A)') 'b[1] --- b[q]'
         ENDIF

         IF (p.EQ.1) THEN
            WRITE(3,'(A)') 'phi[1]'
         ELSEIF (p.EQ.2) THEN
            WRITE(3,'(A)') 'phi[1] phi[2]'
         ELSEIF (p.GE.3 .AND. p.LE.9) THEN
            WRITE(3,'(A,I1,A)') 'phi[1] --- phi[',p,']'
         ELSEIF (p.GE.10 .AND. p.LE.99) THEN
            WRITE(3,'(A,I2,A)') 'phi[1] --- phi[',p,']'
         ELSE
            WRITE(3,'(A)') 'phi[1] --- phi[p]'
         ENDIF

         IF (q.EQ.1) THEN
            WRITE(3,'(A)') 'theta[1]'
         ELSEIF (q.EQ.2) THEN
            WRITE(3,'(A)') 'theta[1] theta[2]'
         ELSEIF (q.GE.3 .AND. q.LE.9) THEN
            WRITE(3,'(A,I1,A)') 'theta[1] --- theta[',q,']'
         ELSEIF (q.GE.10 .AND. q.LE.99) THEN
            WRITE(3,'(A,I2,A)') 'theta[1] --- theta[',q,']'
```

```
         ELSE
            WRITE(3,'(A)') 'theta[1] --- theta[q]'
         ENDIF

         WRITE(3,'(A)') 'e_var        #- output record form'

      ENDIF
   ENDIF

   RETURN
   END


*************************************************************************
   SUBROUTINE loglog(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,
  $            algo_1,algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)
*************************************************************************
   IMPLICIT NONE
   CHARACTER*12 i_file,o_file,e_file
   INTEGER*4 N_skip,Nproce,Nintvl
   INTEGER*1 p,q,algo_1,algo_2,bintyp,e_mode
   REAL*8 u0,v0,k1,c1,c2,r2,a2

   INTEGER*1 i
   CHARACTER*12 l_file
   INTEGER*4 f_leng,f_Nrec
   REAL*8 fmaxuv

   l_file=o_file
   DO 111 i=12,1,-1
      IF (l_file(i:i).EQ.'.') THEN
         l_file(i+1:i+3)='LOG'
         GOTO 123
      ENDIF
111  CONTINUE

123  OPEN(UNIT=3,FILE=l_file,ERR=222,STATUS='NEW')
   WRITE(3,*) bintyp,'  #+ input binary type(2= 2byte_int, 8= 8byte_r
  $eal)'
   WRITE(3,'(2A)') i_file,'  #+ input data file name'
   WRITE(3,*) N_skip,'  #+ number of data to be skipped'
   WRITE(3,*) Nproce,'  #+ number of data to be processed'
   WRITE(3,'(2A)') o_file,'  #+ output data file name(8-byte real num
  $ber binary)'
   WRITE(3,*) Nintvl,'  #+ output interval of estimated parameters'

   WRITE(3,*) p,'  #+ p(AR order of unknown system)'
   WRITE(3,*) q,'  #+ q(MA order of unknown system)'
   WRITE(3,*) algo_1,'  #+ algorithm_1(0=White,1=Feintuch)'
   WRITE(3,*) algo_2,'  #+ algorithm_2(0=White,1=Stochastic,2=Variabl
  $e)'
   IF (algo_2.EQ.0) THEN
      WRITE(3,77) u0,'  #+ u0(adaptive gain constant for a[j])'
      WRITE(3,*) e_mode,'  #+ flag of the output e_file(0=No,1=Yes)'
      IF (e_mode.EQ.1) THEN
         WRITE(3,'(2A)') e_file,'  #+ output e_file name(8-byte rea
  $l number binary)'
      ENDIF
      WRITE(3,'(A)') 's           #+ for input re-direction (< fil
  $ename.ini)'
      WRITE(3,77) v0,'  #- v0(adaptive gain constant for b[j])'
   ELSEIF (algo_2.EQ.1) THEN
      WRITE(3,77) k1,'  #+ k1(v[j]=k1*v0/(c1+j))'
      WRITE(3,77) c1,'  #+ c1(v[j]=k1*v0/(c1+j))'
      WRITE(3,*) e_mode,'  #+ flag of the output e_file(0=No,1=Yes)'
      IF (e_mode.EQ.1) THEN
         WRITE(3,'(2A)') e_file,'  #+ output e_file name(8-byte rea
  $l number binary)'
      ENDIF
      WRITE(3,'(A)') 's           #+ for input re-direction (< fil
  $ename.ini)'
      WRITE(3,77) u0,'  #- u0(adaptive gain constant for a[j])'
      WRITE(3,77) v0,'  #- v0(adaptive gain constant for b[j])'
   ELSEIF (algo_2.EQ.2) THEN
      WRITE(3,77) c2,'  #+ c2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))'
      WRITE(3,77) r2,'  #+ r2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))'
      WRITE(3,77) a2,'  #+ a2(v[j]=v0/(c2+r2*(1-e^(-(a2*j)))))'
      WRITE(3,*) e_mode,'  #+ flag of the output e_file(0=No,1=Yes)'
      IF (e_mode.EQ.1) THEN
         WRITE(3,'(2A)') e_file,'  #+ output e_file name(8-byte rea
  $l number binary)'
```

```
        ENDIF
        WRITE(3,'(A)') 's           #+ for input re-direction (< fil
     $ename.ini)'
        WRITE(3,77) u0,'  #- u0(adaptive gain constant for a[j])'
        WRITE(3,77) v0,'  #- v0(adaptive gain constant for b[j])'
      ENDIF

      WRITE(3,77) fmaxuv(bintyp,i_file,N_skip,Nproce,p,q),
     $          '  #- u_max,v_max'

      WRITE(3,*) f_leng(p,q),'  #- number of output data par record'
      WRITE(3,*) f_Nrec(Nproce,Nintvl),'  #- number of output records'
      WRITE(3,*) f_leng(p,q)*8*f_Nrec(Nproce,Nintvl),'  #- output file s
     $ize[byte]'
      CALL form(p,q)

   77 FORMAT(1P,G12.5E2,A)


      WRITE(6,*) ' ### log file name: ',l_file
      CLOSE(3)
      RETURN

  222 WRITE(6,*) ' ### error: can''t open log-file'
      STOP

      END


************************************************************************
      SUBROUTINE main_c(bintyp,i_file,N_skip,Nproce,o_file,Nintvl,p,q,
     $                  algo_1,algo_2,u0,v0,k1,c1,c2,r2,a2,e_file,e_mode)
************************************************************************
      IMPLICIT NONE
      CHARACTER*12 i_file,o_file,e_file
      INTEGER*4 N_skip,Nproce,Nintvl
      INTEGER*1 p,q,algo_1,algo_2,maxdim,bintyp,e_mode
      REAL*8 u0,v0,k1,c1,c2,r2,a2

      PARAMETER (maxdim=127)

      REAL*8 a(1:maxdim),b(1:maxdim),alpha(1:maxdim,0:maxdim),
     $       beta(0:maxdim-1,0:maxdim),d(0:maxdim),y(0:maxdim),
     $       e(0:maxdim),phi(1:maxdim),theta(1:maxdim),
     $       a_next(1:maxdim),b_next(1:maxdim),u,v
      INTEGER*4 Ncount,o_cnt

      REAL*8 d_read,y_cal,e_cal
      REAL*8 e_var,ev_cal,ev_sum,sum2

      CALL init_c(a,b,alpha,beta,d,y,e,maxdim)

      o_cnt=Nintvl

      DO 111 Ncount=0,Nproce

         IF (Ncount.GT.0) THEN
            d(0)=d_read(bintyp,i_file,Ncount,N_skip,Nproce)
            y(0)=y_cal(b,d,a,y,maxdim,p,q)
            e(0)=e_cal(d(0),y(0))
         ENDIF

         IF (e_mode.EQ.1) THEN
            CALL e_outp(e(0),e_file,Ncount,Nproce)
         ENDIF

         sum2=ev_sum(e(0),o_cnt,Ncount)

         IF (o_cnt.EQ.Nintvl) THEN

C           /* when not MA(q) */
            IF (p.NE.0) THEN
               CALL ph_cal(phi,a,b,maxdim,p,q)
            ENDIF

C           /* when not AR(p) */
            IF (q.NE.0) THEN
               CALL th_cal(theta,a,maxdim,q)
            ENDIF

            e_var=ev_cal(sum2,Nintvl)
```

```
            CALL output(a,b,phi,theta,e_var,maxdim,p,q,o_file,
     $                  Ncount,Nintvl)

            o_cnt=0

         ENDIF

         o_cnt=o_cnt+1

C        /* when not AR(p) */
         IF (q.NE.0) THEN
            CALL al_cal(alpha,a,y,d,maxdim,p,q,algo_1)
         ENDIF

         CALL be_cal(beta,a,d,alpha,maxdim,p,q,algo_1)

         CALL uv_cal(u,v,u0,v0,algo_2,Ncount,k1,c1,c2,r2,a2)

C        /* when not AR(p) */
         IF (q.NE.0) THEN
            CALL a_cal(a_next,a,alpha,e(0),maxdim,q,u)
         ENDIF

         CALL b_cal(b_next,b,beta,a_next,e(0),maxdim,p,q,v)

         CALL shift(a,b,alpha,beta,a_next,b_next,y,d,e,maxdim,p,q)

 111  CONTINUE

      RETURN
      END


************************************************************************
      SUBROUTINE init_c(a,b,alpha,beta,d,y,e,maxdim)
************************************************************************
      IMPLICIT NONE
      INTEGER*1 maxdim
      REAL*8 a(1:maxdim),b(1:maxdim),alpha(1:maxdim,0:maxdim),
     $       beta(0:maxdim-1,0:maxdim),d(0:maxdim),y(0:maxdim),
     $       e(0:maxdim)

      INTEGER*1 i,j

      d(0)=0.0D0
      y(0)=0.0D0
      e(0)=0.0D0

      DO 111 i=1,maxdim
         a(i)=0.0D0
         b(i)=0.0D0
         d(i)=0.0D0
         y(i)=0.0D0
         e(i)=0.0D0
 111  CONTINUE

      DO 222 j=0,maxdim
         DO 333 i=1,maxdim
            alpha(i,j)=0.0D0
            beta(i-1,j)=0.0D0
 333     CONTINUE
 222  CONTINUE

      RETURN
      END


************************************************************************
      REAL*8 FUNCTION y_cal(b,d,a,y,maxdim,p,q)
************************************************************************
      IMPLICIT NONE
      INTEGER*1 maxdim,p,q
      REAL*8 b(1:maxdim),d(0:maxdim),a(1:maxdim),y(0:maxdim)

      INTEGER*1 k

      y_cal=0.0D0

      IF (p.GE.q) THEN
C        /* when AR(p) or ARMA(p,q) p>=q */
```

```
        DO 111 k=1,p
           y_cal=y_cal+b(k)*d(k)
 111     CONTINUE
        DO 222 k=1,q
           y_cal=y_cal+a(k)*y(k)
 222     CONTINUE

     ELSE
C       /* when MA(q) or ARMA(p,q) p<q*/
        DO 333 k=1,q
           y_cal=y_cal+b(k)*d(k)
 333     CONTINUE
        DO 444 k=1,q
           y_cal=y_cal+a(k)*y(k)
 444     CONTINUE
     ENDIF

     RETURN
     END


*************************************************************************
     REAL*8 FUNCTION e_cal(dd,yy)
*************************************************************************
     IMPLICIT NONE

     REAL*8 dd,yy

     e_cal=dd-yy

     RETURN
     END


*************************************************************************
     SUBROUTINE ph_cal(phi,a,b,maxdim,p,q)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 maxdim,p,q
     REAL*8 phi(1:maxdim),a(1:maxdim),b(1:maxdim)

     INTEGER*1 k

     IF (p.GE.q) THEN
C       /* when AR(p) or ARMA(p,q) p>=q */
        DO 111 k=1,q
           phi(k)=a(k)+b(k)
 111     CONTINUE
        DO 222 k=q+1,p
           phi(k)=b(k)
 222     CONTINUE
     ELSE
C       /* when ARMA(p,q) p<q*/
        DO 333 k=1,p
           phi(k)=a(k)+b(k)
 333     CONTINUE
     ENDIF

     RETURN
     END


*************************************************************************
     SUBROUTINE th_cal(theta,a,maxdim,q)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 maxdim,q
     REAL*8 theta(1:maxdim),a(1:maxdim)

     INTEGER*1 k

     DO 111 k=1,q
        theta(k)=-a(k)
 111  CONTINUE

     RETURN
     END


*************************************************************************
     SUBROUTINE al_cal(alpha,a,y,d,maxdim,p,q,algo_1)
```

```
************************************************************************
      IMPLICIT NONE
      INTEGER*1 maxdim,p,q,algo_1
      REAL*8 alpha(1:maxdim,0:maxdim),a(1:maxdim),y(0:maxdim),
     $      d(0:maxdim)

      INTEGER*1 k,i

      IF (p.GE.q) THEN
C        /* when ARMA(p,q) p>=q */
         DO 111 k=1,q

            alpha(k,0)=0.0D0

            IF (algo_1.EQ.0) THEN
C              /* when White's algorithm */
               DO 222 i=1,q
                  alpha(k,0)=alpha(k,0)+a(i)*alpha(k,i)
 222           CONTINUE
            ELSE
C              /* when Feintuch's algorithm */
            ENDIF

            alpha(k,0)=alpha(k,0)+y(k)

 111     CONTINUE

      ELSE
C        /* when MA(q) or ARMA(p,q) p<q */
         DO 333 k=1,p

            alpha(k,0)=0.0D0

            IF (algo_1.EQ.0) THEN
C              /* when White's algorithm */
               DO 444 i=1,q
                  alpha(k,0)=alpha(k,0)+a(i)*alpha(k,i)
 444           CONTINUE
            ELSE
C              /* when Feintuch's algorithm */
            ENDIF

            alpha(k,0)=alpha(k,0)+y(k)

 333     CONTINUE

         DO 555 k=p+1,q

            alpha(k,0)=0.0D0

            IF (algo_1.EQ.0) THEN
C              /* when White's algorithm */
               DO 666 i=1,q
                  alpha(k,0)=alpha(k,0)+a(i)*alpha(k,i)
 666           CONTINUE
            ELSE
C              /* when Feintuch's algorithm */
            ENDIF

            alpha(k,0)=alpha(k,0)+y(k)-d(k)

 555     CONTINUE

      ENDIF

      RETURN
      END


************************************************************************
      SUBROUTINE be_cal(beta,a,d,alpha,maxdim,p,q,algo_1)
************************************************************************
      IMPLICIT NONE
      INTEGER*1 maxdim,p,q,algo_1
      REAL*8 beta(0:maxdim-1,0:maxdim),a(1:maxdim),d(0:maxdim),
     $      alpha(1:maxdim,0:maxdim)

      INTEGER*1 k,i

      IF (p.GE.q) THEN
C        /* when AR(p) or ARMA(p,q) p>=q */
```

```
        DO 111 k=0,p-1

           beta(k,0)=0.0D0

           IF (algo_1.EQ.0) THEN
C          ..........  /* when White's algorithm */
              DO 222 i=1,q
                 beta(k,0)=beta(k,0)+a(i)*beta(k,i)
 222          CONTINUE
           ELSE
C             /* when Feintuch's algorithm */
           ENDIF

           beta(k,0)=beta(k,0)+d(k+1)

 111    CONTINUE

     ELSE
C       /* when MA(q) or ARMA(p,q) p<q*/
        DO 333 k=0,p-1

           beta(k,0)=0.0D0

           IF (algo_1.EQ.0) THEN
C             /* when White's algorithm */
              DO 444 i=1,q
                 beta(k,0)=beta(k,0)+a(i)*beta(k,i)
 444          CONTINUE
           ELSE
C             /* when Feintuch's algorithm */
           ENDIF

           beta(k,0)=beta(k,0)+d(k+1)

 333    CONTINUE

        DO 555 k=p,q-1
           beta(k,0)=-alpha(k+1,0)
 555    CONTINUE

     ENDIF

     RETURN
     END


*************************************************************************
     SUBROUTINE a_cal(a_next,a,alpha,ee,maxdim,q,u)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 maxdim,q
     REAL*8 a_next(1:maxdim),a(1:maxdim),alpha(1:maxdim,0:maxdim),u

     REAL*8 ee
     INTEGER*1 k

     DO 111 k=1,q
        a_next(k)=a(k)+u*ee*alpha(k,0)
 111  CONTINUE

     RETURN
     END


*************************************************************************
     SUBROUTINE b_cal(b_next,b,beta,a_next,ee,maxdim,p,q,v)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 maxdim,p,q
     REAL*8 b_next(1:maxdim),b(1:maxdim),beta(0:maxdim-1,0:maxdim),
    $       a_next(1:maxdim),v

     REAL*8 ee
     INTEGER*1 k

     IF (p.GE.q) THEN
C       /* when AR(p) or ARMA(p,q) p>=q */
        DO 111 k=0,p-1
           b_next(k+1)=b(k+1)+v*ee*beta(k,0)
 111    CONTINUE
     ELSE
```

```
C       /* when MA(q) or ARMA(p,q) p<q*/
        DO 222 k=0,p-1
           b_next(k+1)=b(k+1)+v*ee*beta(k,0)
 222     CONTINUE
        DO 333 k=p,q-1
           b_next(k+1)=-a_next(k+1)
 333     CONTINUE
     ENDIF

     RETURN
     END

*************************************************************************
     SUBROUTINE shift(a,b,alpha,beta,a_next,b_next,y,d,e,maxdim,p,q)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 maxdim,p,q
     REAL*8 a(1:maxdim),b(1:maxdim),alpha(1:maxdim,0:maxdim),
    $       beta(0:maxdim-1,0:maxdim),a_next(1:maxdim),
    $       b_next(1:maxdim),y(0:maxdim),d(0:maxdim),e(0:maxdim)

     INTEGER*1 k,i

     IF (p.GE.q) THEN
        IF (q.NE.0) THEN
C       /* when ARMA(p,q) p>=q */
           DO 111 k=q,1,-1
              a(k)=a_next(k)
              e(k)=e(k-1)
              y(k)=y(k-1)
              DO 222 i=q-1,0,-1
                 alpha(k,i+1)=alpha(k,i)
 222             CONTINUE
 111          CONTINUE
              DO 333 k=p-1,0,-1
                 DO 444 i=q-1,0,-1
                    beta(k,i+1)=beta(k,i)
 444             CONTINUE
 333          CONTINUE
        ELSE
C       /* when AR(p) */
        ENDIF

        DO 555 k=p-1,0,-1
           d(k+1)=d(k)
           b(k+1)=b_next(k+1)
 555     CONTINUE

     ELSE
C       /* when MA(q) or ARMA(p,q) p<q*/
           DO 666 k=q,1,-1
              a(k)=a_next(k)
              b(k)=b_next(k)
              y(k)=y(k-1)
              e(k)=e(k-1)
              d(k)=d(k-1)
              DO 777 i=q-1,0,-1
                 alpha(k,i+1)=alpha(k,i)
                 beta(k-1,i+1)=beta(k-1,i)
 777             CONTINUE
 666          CONTINUE
     ENDIF

     RETURN
     END

*************************************************************************
     SUBROUTINE uv_cal(u,v,u0,v0,algo_2,Ncount,k1,c1,c2,r2,a2)
*************************************************************************
     IMPLICIT NONE
     INTEGER*1 algo_2
     REAL*8 u,v,u0,v0,k1,c1,c2,r2,a2
     INTEGER*4 Ncount

     IF (algo_2.EQ.0) THEN
        v=v0
     ELSEIF (algo_2.EQ.1) THEN
        v=k1*v0/(c1+Ncount)
     ELSEIF (algo_2.EQ.2) THEN
```

```
        v=v0/(c2+r2*(1.0D0-DEXP(-a2*Ncount)))
      ENDIF

      u=u0
      u=v

      RETURN
      END

*************************************************************************
      REAL*8 FUNCTION d_read(bintyp,i_file,Ncount,N_skip,Nproce)
*************************************************************************
      IMPLICIT NONE
      CHARACTER*12 i_file
      INTEGER*4 Ncount,N_skip,Nproce
      INTEGER*1 bintyp

      INTEGER*4 N_bufr,pos,br_cnt,i
      PARAMETER (N_bufr=10000)
      REAL*8 d_bufr(1:N_bufr)
      INTEGER*2 i_bufr(1:N_bufr)

      SAVE br_cnt,d_bufr,i_bufr


      IF (Ncount.EQ.1) THEN
         br_cnt=1
         OPEN(UNIT=1,FILE=i_file,ACCESS='DIRECT',RECL=bintyp,
     $        STATUS='OLD')
      ENDIF

      IF (br_cnt.EQ.1) THEN
         DO 111 i=1,N_bufr
            pos=N_skip+Ncount+i-1
            IF (bintyp.EQ.2) THEN
               READ(1,REC=pos) i_bufr(i)
               IF (pos.EQ.N_skip+Nproce) THEN
                  GOTO 222
               ENDIF
            ELSEIF (bintyp.EQ.8) THEN
               READ(1,REC=pos) d_bufr(i)
               IF (pos.EQ.N_skip+Nproce) THEN
                  GOTO 222
               ENDIF
            ENDIF
 111     CONTINUE
 222  ENDIF

      IF (bintyp.EQ.2) THEN
         d_read=i_bufr(br_cnt)*1.0D0
      ELSEIF (bintyp.EQ.8) THEN
         d_read=d_bufr(br_cnt)
      ENDIF

      IF (br_cnt.EQ.N_bufr) THEN
         br_cnt=0
      ENDIF

      br_cnt=br_cnt+1

      IF (Ncount.EQ.Nproce) THEN
         CLOSE(1)
      ENDIF

      RETURN
      END


*************************************************************************
      SUBROUTINE output(a,b,phi,theta,e_var,maxdim,p,q,o_file,
     $                  Ncount,Nintvl)
*************************************************************************
      IMPLICIT NONE
      CHARACTER*12 o_file
      INTEGER*4 Ncount,Nintvl
      INTEGER*1 maxdim,p,q
      REAL*8 a(1:maxdim),b(1:maxdim),phi(1:maxdim),theta(1:maxdim)

      REAL*8 e_var
      INTEGER*4 pos,f_leng,f_Nrec
```

```
      INTEGER*1 i

      pos=f_Nrec(Ncount,Nintvl)

      IF (Ncount.EQ.0) THEN
         OPEN(UNIT=2,FILE=o_file,ACCESS='DIRECT',RECL=8*f_leng(p,q),
     $        STATUS='NEW',ERR=777)
      ELSE
         OPEN(UNIT=2,FILE=o_file,ACCESS='DIRECT',RECL=8*f_leng(p,q))
      ENDIF

      IF (p.GE.q) THEN
         IF (q.EQ.0) THEN
C           /* when AR(p) */
            WRITE(2,REC=pos) (b(i),i=1,p),(phi(i),i=1,p),e_var
         ELSE
C           /* when ARMA(p,q) p>=q */
            WRITE(2,REC=pos) (a(i),i=1,q),(b(i),i=1,p),
     $                       (phi(i),i=1,p),(theta(i),i=1,q),e_var
         ENDIF
      ELSE
         IF (p.EQ.0) THEN
C           /* when MA(q) */
            WRITE(2,REC=pos) (a(i),i=1,q),(b(i),i=1,q),
     $                       (theta(i),i=1,q),e_var
         ELSE
C           /* when ARMA(p,q) p<q */
            WRITE(2,REC=pos) (a(i),i=1,q),(b(i),i=1,q),
     $                       (phi(i),i=1,p),(theta(i),i=1,q),e_var
         ENDIF
      ENDIF

      CLOSE(2)

      RETURN

  777 WRITE(6,*) ' ### error: can''t open output file(o_file)'
      STOP

      END


************************************************************************
      REAL*8 FUNCTION ev_sum(ee,o_cnt,Ncount)
************************************************************************
      IMPLICIT NONE
      INTEGER*4 Ncount,o_cnt
      REAL*8 ee,sum2

      SAVE sum2

      IF (o_cnt.EQ.1 .OR. Ncount.EQ.0) THEN
         sum2=0.0D0
      ENDIF

      sum2=sum2+ee*ee

      ev_sum=sum2

      RETURN
      END


************************************************************************
      REAL*8 FUNCTION ev_cal(sum2,Nintvl)
************************************************************************
      IMPLICIT NONE
      INTEGER*4 Nintvl
      REAL*8 sum2

      ev_cal=sum2/Nintvl

      RETURN
      END


************************************************************************
      SUBROUTINE e_outp(ee,e_file,Ncount,Nproce)
************************************************************************
      IMPLICIT NONE
      CHARACTER*12 e_file
```

```
      REAL*8 ee
      INTEGER*4 Ncount,Nproce

      INTEGER*4 i,pos,N_bufe,be_cnt
      PARAMETER (N_bufe=10000)
      REAL*8 bufe(1:N_bufe)

      SAVE be_cnt,bufe

      IF (Ncount.EQ.0) THEN
         be_cnt=N_bufe
         OPEN(UNIT=4,FILE=e_file,ACCESS='DIRECT',RECL=8,STATUS='NEW',
     $        ERR=777)
      ENDIF

      bufe(be_cnt)=ee

      IF (be_cnt.EQ.N_bufe .OR. Ncount.EQ.Nproce) THEN
         IF (Ncount.EQ.0) THEN
            pos=1
            WRITE(4,REC=pos) bufe(be_cnt)
         ELSE
            DO 111 i=1,be_cnt
               pos=Ncount-be_cnt+1+i
               WRITE(4,REC=pos) bufe(i)
 111        CONTINUE
         ENDIF

         be_cnt=0
      ENDIF

      be_cnt=be_cnt+1

      IF (Ncount.EQ.Nproce) THEN
         CLOSE(4)
      ENDIF

      RETURN

 777  WRITE(6,*) ' ### error: can''t open output file(e_file)'
      STOP

      END


C     /* End of File */
```

B　ＦＦＴ計算プログラムリスト

B-1：　FFT_main.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/////////////////////////////////////////////////////////////////////////////
//                                                                           //
//  Power Spectrum Density Calculation Program by Direct FFT Method.         //
//                                                                           //
//                                                                           //
/////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* MAIN FUNCTION SETTING.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#define    FFT_MAIN

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_fft.h"
#include "VAR_fft.h"
#include "PRT_fft.h"

/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : void main(int argc, char **argv)                           */
/*  prototype   : none                                                       */
/*  Function    : Main function for fft program.                             */
/*  Return Value: none                                                       */
/*  Variable    : argc  ; (i  ) Number of variable on the command line.      */
/*                **argv; (i  ) Pointer of strings on the command line.      */
/*。。Revision。。 : 2001/01/01                                               */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  main(int argc, char **argv)
{

     int        check;

     if ( argc != 3 ) {
          Print_Usage();
          exit(0);
     }

     Title();

     if ( (G_fout_report = fopen(argv[2], "wt")) == NULL ) {
          printf( "Fatal Error: Cannot open the report file [%s]!! ¥n", argv[2] );
          exit(0);
     }

     check = AutoMain(argv);

     if ( check == SUCCESS ) {
          sprintf(G_reportbuff[0],
          "========================= NORMAL TERMINATION =========================¥n¥n");
    } else {
          sprintf(G_reportbuff[0],
          "========================= ABNORMAL TERMINATION =========================¥n¥n");
```

```
    }

    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    fclose(G_fout_report);
}

/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : void Print_Usage(void)                                  */
/* prototype   : PRT_fft.h                                               */
/* Function    : Display Usage                                           */
/* Return Value: none                                                    */
/* Variable    : none                                                    */
/*。。Revision。。 : 2001/01/01                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
void  Print_Usage(void) {

    printf( "Usage:¥n" );
    printf( "  fft.out  control_filename  report_filename¥n" );
    printf( "             control_filname: path of job file.¥n" );
    printf( "             report_filnema : path of report file.¥n¥n" );

}

/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : void Title(void)                                        */
/* prototype   : PRT_fft.h                                               */
/* Function    : Display Title                                           */
/* Return Value: none                                                    */
/* Variable    : none                                                    */
/*。。Revision。。 : 2001/01/01                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
void  Title(void) {

    printf("¥n¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("_/                                                          _/¥n");
    printf("_/ FFT: POWER SPECTRUM DENSITY CALCULATION PROGRAM BY DIRECT FFT METHOD _/¥n");
    printf("_/                                                          _/¥n");
    printf("_/     Version 1.01 : 2001/01/01 : transplantation from PC version.     _/¥n");
    printf("_/                                                          _/¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("¥n¥n");

}

/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : int AutoMain(void)                                      */
/* prototype   : PRT_fft.h                                               */
/* Function    : Execution Control                                       */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_fft.h]  */
/*               abnormal termination -> FAULT   [defined in DFN_fft.h]  */
/* Variable    : **argv; (i  ) Pointer of strings on the command line.   */
/*。。Revision。。 : 2001/01/01                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
int  AutoMain(char **argv)
{
    FILE        *fin_control;
    char        setup_filename[256];
    int         count, check;
    char        answer;


    /* Initialize */
```

```
    count = 0;

                                                                    /* Job control
file open */
    if ( (fin_control = fopen(argv[1], "rt")) == NULL) {
        printf( "Fatal Error: Cannot open the control file [%s]!! ¥n", argv[1] )
;
        return FAULT;
    }


     /* Roop process */
     while ( !feof(fin_control) ) {
          count++;
          fscanf(fin_control, "%s", setup_filename);
          if ( feof(fin_control) ) {
               break;
          }

          sprintf( G_reportbuff[0], "¥n¥n" );
          sprintf( G_reportbuff[1],
               "--- FFT Calculation ----------------------------------------------------------------------¥n");
       sprintf( G_reportbuff[2], "JOB No. %d¥n", count );
       sprintf( G_reportbuff[3], "Setup File Name = %s¥n¥n", setup_filename );
       W_Writing( CRT_DISK, G_reportbuff, 4, G_fout_report );

       if ( ReadFileCall(setup_filename) == FAULT) {
           continue;
       }
       check = FFT_Call();
     }
     fclose(fin_control);

     return check;

}
```

B-2: fft_cal.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/////////////////////////////////////////////////////////////////////////////
//                                                                         //
//  Power Spectrum Density Calcuration Program by Direct FFT Method.       //
//                                                                         //
//                                                                         //
/////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_fft.h"
#include "VAR_fft.h"
#include "PRT_fft.h"

/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Formula     : void FFT_Call(void)                                        */
/*  Prototype   : PRT_fft.h                                                  */
/*  Description : FFT calculation process routine.                           */
```

```
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_fft.h]      */
/*                abnormal termination -> FAULT    [defined in DFN_fft.h]      */
/*  Arguments   : none                                                        */
/*¡¡Revision¡¡  : 2001/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  FFT_Call(void)
{
     FILE            *fin_data, *fout;
     time_t               start_time, end_time;
     char            day_buff[32], time_buff[32];
     int                  i, j;
     int             full_memory_size, half_memory_size;
     long            add_seek;
     double               require_time;
     double               start_point_time, during_time, end_point_time;
     double               const_argument, argument, power_factor, window_power, minimum_frequency,
                          frequency;
     double               *real, *imagi, *psd, *wn_real, *wn_imagi;

     /* Infomation */
     start_point_time = G_datafile.skip_number / G_fft_calc.sampling_frequency;
     during_time      = G_fft_calc.block_number +
                                    ( G_fft_calc.block_number - G_fft_calc.overrap ) *
                                                  (G_fft_calc.averaging - 1 );
     during_time     /= G_fft_calc.sampling_frequency;
     end_point_time   = start_point_time + during_time;
     sprintf(G_reportbuff[0], "¥nSpan of calculation  = %.6f [sec] ---> %.6f [sec]¥n",
               start_point_time, end_point_time );
     sprintf(G_reportbuff[1], "Number of averaging  = %d¥n", G_fft_calc.averaging );
     W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);

     /* Memory assignment */
     full_memory_size = G_fft_calc.block_number;
     half_memory_size = full_memory_size / 2;
     real     = (double *)calloc(full_memory_size, sizeof(double));
     imagi    = (double *)calloc(full_memory_size, sizeof(double));
     psd      = (double *)calloc(full_memory_size, sizeof(double));
     wn_real  = (double *)calloc(half_memory_size, sizeof(double));
     wn_imagi = (double *)calloc(half_memory_size, sizeof(double));

     if (real == NULL || imagi == NULL || psd == NULL ||
                                    wn_real == NULL || wn_imagi == NULL) {
          sprintf(G_reportbuff[0], "Fatal conditions with Out of memory¥n" );
          W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
          return FAULT;
     }

     /* Preparation for calculation */
     const_argument = 2 * M_PI / (double)G_fft_calc.block_number;
     for (i = 0; i < (G_fft_calc.block_number / 2); i++) {
          argument = (double)i * const_argument;
          wn_real[i]  = cos(argument);
          wn_imagi[i] = sin(argument);
     }

     if (G_fft_calc.spectrum_type == PSD) {
          power_factor = G_fft_calc.sampling_frequency;
     } else {
          power_factor = 1.0;
     }

     add_seek = -1 * G_fft_calc.overrap * G_datafile.byte;

     /* File open and binaly data skipping */
```

```
    fin_data = fopen(G_datafile.name, "rb");
    fseek(fin_data, G_datafile.skip_number, SEEK_SET);

    /* Start time setting */
    start_time = Now_Time_Set(day_buff, time_buff);
    sprintf(G_reportbuff[0], "Start of Calculation = %s %s¥n", day_buff, time_buff);
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    /* Main calculation */
    for (i = 1; i <= G_fft_calc.averaging; i++) {
        if (i != 1) {
            fseek(fin_data, add_seek, SEEK_CUR);
        }
        for (j = 0; j < G_fft_calc.block_number; j++) {
            real[j] = Read_Data(fin_data, G_datafile.byte);
            imagi[j] = 0.0;
        }

        if ( G_fft_calc.dc_filter == ON ) {
            Mean_Delete(real, G_fft_calc.block_number);
        }

        window_power = Window_Function(G_fft_calc.window, real, G_fft_calc.block_number);

        Bit_Reversal(real, G_fft_calc.block_number);

        Butterfly(real, imagi, G_fft_calc.block_number, wn_real, wn_imagi, G_fft_calc.max_block);

        for (j = 0; j < G_fft_calc.block_number; j++) {
            psd[j] += ((real[j]*real[j] + imagi[j]*imagi[j])
                                                / window_power / power_factor);
        }

    }
    for (i = 0; i < G_fft_calc.block_number; i++) {
        psd[i] /= G_fft_calc.averaging;
    }

    /* Output of result */
    minimum_frequency = G_fft_calc.sampling_frequency / (double)G_fft_calc.block_number;

    fout = fopen(G_out_filename, "wt");

    if (G_out_filetype == PL3)
        fprintf(fout, "%d¥n", (G_fft_calc.block_number/2 - 1));

    for (i = 1; i < (G_fft_calc.block_number/2); i++) {
        frequency = minimum_frequency * i;
        fprintf(fout, "%e    %e¥n",frequency, psd[i]);
    }

    if (G_out_filetype == PL3) {
/*
        Frame_Set(psd, &d1_dummy, &d2_dummy);
        Footer();
*/
    }

    /* End time setting */
    end_time = Now_Time_Set(day_buff, time_buff);
    require_time = difftime(end_time, start_time);
    sprintf(G_reportbuff[0], "End of Calculation   = %s %s¥n", day_buff, time_buff);
    sprintf(G_reportbuff[1], "Requirement Time     = %f [sec]¥n", require_time);
    W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);
```

```
    /* Memory release and file close */
    free(real);
    free(imagi);
    free(psd);
    free(wn_real);
    free(wn_imagi);
    fclose(fin_data);
    fclose(fout);

    return SUCCESS;
}



/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : void Mean_Delete(double *real, int ndata)              */
/* Prototype   : PRT_fft.h                                              */
/* Discription : Logical check of various parameter.                    */
/* Return Value: none                                                   */
/* Variable    : *real; (i  ) pointer of array contained time serieas data.  */
/*               ndata; (i  ) size of array.                            */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
void  Mean_Delete(double *real, int ndata)
{
    int         i;
    double      mean;

    mean = 0.0;
    for (i = 0; i < ndata; i++)
        mean += real[i];

    mean /= (double)ndata;

    for (i = 0; i < ndata; i++)
        real[i] -= mean;
}



/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : double Read_Data(FILE *fin, int size)                  */
/* Prototype   : PRT_fft.h                                              */
/* Discription : Reading one data from the binary file.                 */
/* Return Value: The data which read from the binary file. When the data size*/
/*               is 2byte, the data is casted 8byte.                    */
/* Variable    : *fin; (i  ) file pointer to the binary file.           */
/*               size; (i  ) the data size.                             */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
double  Read_Data(FILE *fin, int byte)
{
    short       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;
}


/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : double Window_Function(int kind, double *real, int ndata)  */
/*  Prototype   : PRT_fft.h                                                  */
/*  Discription : Filtering data by specified window function and return to  */
/*                power.                                                     */
/*  Return Value: a value of power.                                          */
/*  Variable    : kind ; (i  ) type of window funciton.                      */
/*                *real; (i/o) pointer of array contained real part data.    */
/*                ndata; (i  ) size of array.                                */
/*  Revision    : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------- */
double  Window_Function(int kind, double *real, int ndata)
{
        int         i;
        double      argument, value, value_sum;


        argument = 2.0 * M_PI / (double)ndata;
        value_sum = 0.0;

        switch (kind) {
                case HANNING :
                        for (i = 0; i < ndata; i++) {
                                value = 0.5 * (1.0 - cos(argument * (double)i));
                                value_sum += (value*value);
                                real[i] *= value;
                        }
                        break;
                case HAMMING :
                        for (i = 0; i < ndata; i++) {
                                value = 0.54 - 0.64*cos(argument * (double)i);
                                value_sum += (value*value);
                                real[i] *= value;
                        }
                        break;
                case BLACKMAN :
                        for (i = 0; i < ndata; i++) {
                                value = 0.42 - 0.50*cos(argument * (double)i) +
                                                        0.08*cos(2 * argument* (double)i);
                                value_sum += (value*value);
                                real[i] *= value;
                        }
                        break;
                case AKAIKE :
                        for (i = 0; i < ndata; i++) {
                                value = 0.625 + 0.50*cos(argument * (double)i) -
                                                        0.125*cos(2 * argument* (double)i);
                                value_sum += (value*value);
                                real[i] *= value;
                        }
                        break;
        }


        return value_sum;
}
```

```
/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : void Bit_Reversal(double *real, int ndata)             */
/*  Prototype   : PRT_fft.h                                              */
/*  Discription : Bit reversal function used by FFT algorithm.           */
/*  Return Value: none.                                                  */
/*  Variable    : *real; (i/o) pointer of array contained real part data.*/
/*                ndata; (i  ) size of array.                            */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
void  Bit_Reversal(double *real, int ndata)
{

     int        i, j, msb;
     double     k, pool;


     j = 0;
     msb = ndata / 2;

     for (i = 0; i < (ndata-1); i++) {
          if ( i <= j) {
               pool    = real[i];
               real[i] = real[j];
               real[j] = pool;
          }
          k = msb;
          while (k <= j) {
               j -= k;
               k /= 2;
          }
          j += k;
     }
}
```

```
/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : void Butterfly(double *real,                           */
/*                               double *image,                          */
/*                               int ndata,                              */
/*                               double *wn_real,                        */
/*                               double *wn_image,                       */
/*                               int max_block )                         */
/*  Prototype   : PRT_fft.h                                              */
/*  Discription : Butterfly operation used by FFT algorithm.             */
/*  Return Value: none.                                                  */
/*  Variable    : *real    ; (i/o) pointer of array contained real part data.*/
/*                *imagi   ; (i/o) pointer of array contained imaginary part*/
/*                                 data.                                */
/*                *wn_real ; (i  ) pointer of array contained working real*/
/*                                 part.                                */
/*                *wn_imagi; (i  ) pointer of array contained working     */
/*                                 imaginary part.                      */
/*                max_block; (i  ) max_block.                           */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
void  Butterfly(double *real,
                double *imagi,
                int ndata,
                double *wn_real,
                double *wn_imagi,
                int max_block)
{
```

```
    int         i, j, k, msb, factor, dft_point, butterfly_loop, target1,
                target2;
    double      convert_wn_real, convert_wn_imagi,
                pool1_real, pool1_imagi, pool2_real, pool2_imagi;


    dft_point = 1;
    factor = ndata;

    for (i = 1; i <= max_block; i++) {
        dft_point *= 2;
        butterfly_loop = ndata / dft_point;
        factor /= 2;

        for (j = 0; j < butterfly_loop; j++) {

            for (k = 0; k < (dft_point/2); k++) {
                target1 = k + dft_point*j;
                target2 = target1 + dft_point / 2;
                convert_wn_real = wn_real[k*factor];
                convert_wn_imagi = wn_imagi[k*factor];
                pool1_real = real[target1];
                pool1_imagi = imagi[target1];
                pool2_real = real[target2];
                pool2_imagi = imagi[target2];
                real[target1] = pool1_real + convert_wn_real*pool2_real
                                           - convert_wn_imagi*pool2_imagi;
                imagi[target1] = pool1_imagi + convert_wn_imagi*
                                 pool2_real + convert_wn_real*pool2_imagi;
                real[target2] = pool1_real - convert_wn_real*pool2_real
                                           + convert_wn_imagi*pool2_imagi;
                imagi[target2] = pool1_imagi - convert_wn_imagi*
                                 pool2_real - convert_wn_real*pool2_imagi;
            }
        }
    }
}
```

B-3: fft_read.c

```
/* ------1--------2--------3--------4--------5--------6--------7--------8
/////////////////////////////////////////////////////////////////////////
//                                                                     //
//  Power Spectrum Density Calcuration Program by Direct FFT Method.   //
//                                                                     //
//                                                                     //
/////////////////////////////////////////////////////////////////////////
--------1--------2--------3--------4--------5--------6--------7------- */


/* ------+--------+--------+--------+--------+--------+--------+------- */
/* standard include defines.                                           */
/* ------+--------+--------+--------+--------+--------+--------+------- */


/* ------+--------+--------+--------+--------+--------+--------+------- */
/* private include defines.                                            */
/* ------+--------+--------+--------+--------+--------+--------+------- */
#include "DFN_fft.h"
#include "VAR_fft.h"
#include "PRT_fft.h"


/* ------+--------+--------+--------+--------+--------+--------+------- */
```

```
/*  Format      : int ReadFileCall(char *filename)                          */
/*  Prototype   : PRT_armax1.h                                              */
/*  Discription : Reading job file and setting parameter.                   */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_fft.h]    */
/*                abnormal termination -> FAULT    [defined in DFN_fft.h]   */
/*  Variable    : *filename; (i  ) pointer to filenmae buffer.              */
/*¡¡Revision¡¡  : 2000/01/01                                                */
/* ------+---------+---------+---------+---------+---------+------------- */
int  ReadFileCall(char *filename)
{
     FILE       *fin_setupfile, *fin_data, *fout_test;
     char       input_buff[256];
     int        i, j;
     int             sample;
     long       useable_number;


     /* Setup file open */
     if ( (fin_setupfile = fopen(filename, "rt")) == NULL ) {
          sprintf(G_reportbuff[0], "Cannot open setup file [ %s ].¥n", filename);
          sprintf(G_reportbuff[1], "Abort this job.¥n", filename);
          W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);
          return FAULT;
     }


     /* Initialize */
     memset(&G_datafile,     0x00, sizeof(S_DATAFILE));
     memset(&G_fft_calc,     0x00, sizeof(S_FFT_CALC));
     memset(G_out_filename,  0x00, 256);


     /* Reading parameter in setup file */
     for ( i = 0; i < 12; i++ ) {
          if ( Std_Reading(fin_setupfile, input_buff, i) == FAULT ) {
               sprintf(G_reportbuff[0], "Error: Format of setup file is unsuitable.¥n");
               W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
               goto ERROR_RECOVERY;
          }
          switch ( i ) {

               case 0: /* Reading record length of binary time series */
                    G_datafile.byte = atoi(input_buff);
                    if ( G_datafile.byte != 2 && G_datafile.byte != 8 ) {
                         sprintf(G_reportbuff[0],
                                        "Illegal value as record length of time series data.¥n");
                         goto ERROR_RECOVERY;
                    }
                    break;

               case 1: /* Reading data filename */
                    if ( (fin_data = fopen(input_buff, "rb")) == NULL ) {
                         sprintf(G_reportbuff[0],
                              "Cannot open the time series data file.¥n" );
                         goto ERROR_RECOVERY;
                     } else {
                         strcpy(G_datafile.name, input_buff);
                         G_datafile.number = Binary_File_Check(fin_data, G_datafile.byte);
                         fclose(fin_data);
                         if ( G_datafile.number == -1 ) {
                              sprintf(G_reportbuff[0],
                                   "Probably, [ %s ] is not binary data file of %d byte¥n",
                                   input_buff, G_datafile.byte);
                              goto ERROR_RECOVERY;
```

```
            }
            useable_number = G_datafile.number;
        }
        break;

    case 2: /* Reading number of skipping data */
        G_datafile.skip_number = atol(input_buff);
        if ( G_datafile.skip_number < 0 || G_datafile.skip_number > useable_number ) {
            sprintf(G_reportbuff[0], "Illegal value as number of skipping data.¥n");
            goto ERROR_RECOVERY;
        }
        useable_number -= G_datafile.skip_number;
        break;

    case 3: /* Reading sampling frequency */
        G_fft_calc.sampling_frequency = atof(input_buff);
        if ( G_fft_calc.sampling_frequency <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as sampling frequency.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 4: /* Reading 1 block data number */
        G_fft_calc.block_number = atoi(input_buff);
        sample = 1;
        for ( j = 1;  j <= 16; j++ ) {
            sample *= 2;
            if ( G_fft_calc.block_number == sample ) {
                G_fft_calc.max_block = j;
                break;
            }
        }
        if ( G_fft_calc.block_number <= 0 ||
                    G_fft_calc.block_number > useable_number ||
                                G_fft_calc.max_block == 0 ) {
            sprintf(G_reportbuff[0], "Illegal value as block number.¥n");
            goto ERROR_RECOVERY;
        }

        break;

    case 5: /* Reading overrap number */
        G_fft_calc.overrap  = atoi(input_buff);
        if ( G_fft_calc.overrap < 0 || G_fft_calc.overrap >= G_fft_calc.block_number ) {
            sprintf(G_reportbuff[0], "Illegal value as overrap number.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 6: /* Reading averaging number */
        if ( strcmp(input_buff, "MAX") == 0 ) {
            G_fft_calc.averaging = -1;
        } else {
            G_fft_calc.averaging  = atoi(input_buff);
            if ( G_fft_calc.averaging <= 0 ) {
                sprintf(G_reportbuff[0], "Illegal value as averaging number.¥n");
                goto ERROR_RECOVERY;
            }
        }
        break;

    case 7: /* Reading window function */
        if ( strcmp(input_buff, "HANNING") == 0 ) {
```

```
                    G_fft_calc.window = HANNING;
               } else if ( strcmp( input_buff, "HAMMING" ) == 0 ) {
                    G_fft_calc.window = HAMMING;
               } else if ( strcmp( input_buff, "BLACKMAN" ) == 0 ) {
                    G_fft_calc.window = BLACKMAN;
               } else if ( strcmp( input_buff, "AKAIKE" ) == 0 ) {
                    G_fft_calc.window = AKAIKE;
               } else {
                    sprintf(G_reportbuff[0],
                              "Illegal selection as window function.¥n");
                    goto ERROR_RECOVERY;
               }
               break;

          case 8: /* Reading Spectrum Type */
               if ( strcmp(input_buff, "PSD") == 0 ) {
                    G_fft_calc.spectrum_type = PSD;
               } else if ( strcmp( input_buff, "ESD" ) == 0 ) {
                    G_fft_calc.spectrum_type = ESD;
               } else {
                    sprintf(G_reportbuff[0],
                              "Illegal selection as spectrum type.¥n");
                    goto ERROR_RECOVERY;
               }
               break;

          case 9: /* Reading DC component filter setting */
               if ( strcmp(input_buff, "OFF") == 0 ) {
                    G_fft_calc.dc_filter = OFF;
               } else if ( strcmp( input_buff, "ON" ) == 0 ) {
                    G_fft_calc.dc_filter = ON;
               } else {
                    sprintf(G_reportbuff[0],
                              "Illegal selection as DC component filter.¥n");
                    goto ERROR_RECOVERY;
               }
               break;


          case 10: /* Reading output type of result */
               if ( strcmp(input_buff, "PL3") == 0 ) {
                    G_out_filetype = PL3;
               } else if ( strcmp( input_buff, "NORMAL" ) == 0 ) {
                    G_out_filetype = NORMAL;
               } else {
                    sprintf(G_reportbuff[0],
                              "Illegal selection as output file type.¥n");
                    goto ERROR_RECOVERY;
               }

          case 11: /* Reading output filename of result */
               if ( (fout_test = fopen(input_buff, "wt")) == NULL ) {
                    sprintf(G_reportbuff[0], "Unsuitable Output File Name.¥n");
                    goto ERROR_RECOVERY;
               }
               fclose(fout_test);
               strcpy(G_out_filename, input_buff);
               break;

     }
}

/* Data aveilability check */
G_fft_calc.averaging = Data_Aveilability_Check( useable_number,
```

```
                                                                    G_fft_calc.block_number,
                                                                    G_fft_calc.overrap,
                                                                    G_fft_calc.averaging );
    if ( G_fft_calc.averaging == 0 ) {
        sprintf(G_reportbuff[0], "There is no aveilable data for calculation.¥n");
        goto ERROR_RECOVERY;
    }

    fclose(fin_setupfile);
    return SUCCESS;


ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    fclose(fin_setupfile);
    return FAULT;
}


/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : long Binary_File_Check(FILE *fin, int byte)            */
/*  Prototype   : PRT_fft.h                                              */
/*  Discription : Check of binary file.                                  */
/*  Return Value: normal termination   -> number of data                 */
/*                abnormal termination -> -1                             */
/*  Variable    : *fin; (i  ) file pointer of data file.                 */
/*                byte; (i  ) byte of data.                              */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
long  Binary_File_Check(FILE *fin, int byte)
{
    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);
    if ( d1_dummy != d2_dummy ) {
        return -1;
    }

    data_number = file_size / byte;

    return data_number;
}


/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Std_Reading(FILE *fin, char *return_buff, int counter)  */
/*  Prototype   : PRT_fft.h                                              */
/*  Discription : Standard reading function of setup file.               */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_fft.h] */
/*                abnormal termination -> FAULT   [defined in DFN_fft.h] */
/*  Variable    : *fin        ; (i  ) file pointer of setup file.        */
/*                *return_buff; (  o) string buffer pointer for reading para- */
/*                                    meter.                           */
/*                counter     ; (i  ) logical ID of parameter.           */
/*¡¡Revision¡¡  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
int  Std_Reading(FILE *fin, char *ptr1, int counter)
{
```

```
    char  tmp_buff1[256];
    int         check;


    check = fscanf(fin, "%s %s", tmp_buff1, ptr1);
    sprintf( G_reportbuff[0], "%s %s¥n", tmp_buff1, ptr1 );
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    if ( (check != 2 ) || (strcmp( tmp_buff1, G_parameter_item[counter]) != 0) ) {
        return FAULT;
    }

    return SUCCESS;
}


/* ------+---------+---------+---------+---------+---------+---------+------ */
/* Format      : int Data_Aveilability_Check(int usuable_number,             */
/*                                           int ndata,                      */
/*                                           int overrap,                    */
/*                                           int average);                   */
/* Prototype   : PRT_fft.h                                                   */
/* Discription : Data aveilability check for fft calculation especialy       */
/*               averaging.                                                  */
/* Return Value: number of averaging                                         */
/* Variable    : usuable_number; (i  ) number of data which can use.         */
/*               ndata         ; (i  ) number of 1 block.                    */
/*               overrap       ; (i  ) number of overrap.                    */
/*               average       ; (i  ) specified number of averaging.        */
/*¡¡Revision¡¡  : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  Data_Aveilability_Check(int usuable_number, int ndata, int overrap, int average )
{
    int         effective_use, max_average;
    double      d_dummy;


    effective_use = ndata - overrap;

    d_dummy = (double)usuable_number / (double)effective_use;
    d_dummy = floor(d_dummy);

    max_average = (int)d_dummy;

    if ( average > max_average || max_average == 0 ) {
        return 0;
    }

    if ( average == -1 ) {
        return max_average;
    } else {
        return average;
    }
}
```

B-4: fft_util.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
//////////////////////////////////////////////////////////////////////////////
//                                                                          //
//  Power Spectrum Density Calcuration Program by Direct FFT Method.        //
//                                                                          //
//                                                                          //
```

```
/*//////////////////////////////////////////////////////////////////////////////
/*--------1---------2---------3---------4---------5---------6---------7--------- */

/* ------+---------+---------+---------+---------+---------+---------+---------- */
/* standard include defines.                                                     */
/* ------+---------+---------+---------+---------+---------+---------+---------- */

/* ------+---------+---------+---------+---------+---------+---------+---------- */
/* private include defines.                                                      */
/* ------+---------+---------+---------+---------+---------+---------+---------- */
#include "DFN_fft.h"
#include "VAR_fft.h"
#include "PRT_fft.h"

/* ------+---------+---------+---------+---------+---------+---------+---------- */
/*  Format      : void W_Writing(int type,                                       */
/*                               char **string,                                  */
/*                               int number                                      */
/*                               FILE *fout)                                     */
/*  Prototype   : PRT_fft.h                                                      */
/*  Description : Output strings to CRT & DISK.                                  */
/*  Return Value: none                                                           */
/*  Variable    : type   ; (i  ) Output type ( CRT, DISK, CRT_DISK ).            */
/*                *string; (i  ) Pointer array of output strings buffer.         */
/*                number;  (i  ) Number of output.                               */
/*                *fout  ; (i  ) File pointer of ourput file.                    */
/*。。Revision。。 : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+---------- */
void  W_Writing(int type, char (*string)[256], int number, FILE *fout)
{
     int        i;

     for ( i = 0; i < number; i++ ) {
          if ( type & CRT ) {
               printf( "%s", string[i] );
          }

          if ( (type & DISK) && ( fout != NULL ) ) {
               fprintf( fout, "%s", string[i] );
          }
     }
}

/* ------+---------+---------+---------+---------+---------+---------+---------- */
/*  Format      : void Now_Time_Set(char *day_buff, char *time_buff)              */
/*  Prototype   : PRT_fft.h                                                      */
/*  Description : Get now time information.                                      */
/*  Return Value: calendar time at now.                                          */
/*  Variable    : *day_buff;  (i  ) pointer of string for day.                   */
/*                *time_buff; (i  ) Pointer of string for time.                  */
/*。。Revision。。 : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+---------- */
time_t  Now_Time_Set(char *day_buff, char *time_buff)
{
    time_t      ct;
    struct tm   *now_time;

    ct = time( NULL );
    now_time = localtime( &ct );
```

```
    sprintf( day_buff,  "%04d/%02d/%02d",
                        now_time->tm_year+1900,
                        now_time->tm_mon+1,
                        now_time->tm_mday);

    sprintf( time_buff, "%02d:%02d:%02d",
                        now_time->tm_hour,
                        now_time->tm_min,
                        now_time->tm_sec );

     return ct;
]
```

B-5: DFN_fft.h

```
/* ------1--------2--------3--------4--------5--------6--------7--------8
//////////////////////////////////////////////////////////////////////////
//                                                                      //
//  Power Spectrum Density Calcuration Program by Direct FFT Method.    //
//                                                                      //
//                                                                      //
//////////////////////////////////////////////////////////////////////////
--------1--------2--------3--------4--------5--------6--------7------- */

/* ================================================================== */
/* prototype define.                                                  */
/* ================================================================== */
/* ------+--------+--------+--------+--------+--------+--------+------ */
/* fft_main.c                                                         */
/* ------+--------+--------+--------+--------+--------+--------+------ */
extern void     Print_Usage(void);
extern void     Title(void);
extern int      AutoMain(char **);

/* ------+--------+--------+--------+--------+--------+--------+------ */
/* fft_read.c                                                         */
/* ------+--------+--------+--------+--------+--------+--------+------ */
extern int      ReadFileCall(char *);
extern long     Binary_File_Check(FILE *, int);
extern int      Std_Reading(FILE *, char *, int);
extern int      Data_Aveilability_Check(int, int, int, int);

/* ------+--------+--------+--------+--------+--------+--------+------ */
/* fft_call.c                                                         */
/* ------+--------+--------+--------+--------+--------+--------+------ */
extern int      FFT_Call(void);
extern void     Mean_Delete(double *, int);
extern double   Read_Data(FILE *, int);
extern double   Window_Function(int, double *, int);
extern void     Bit_Reversal(double *, int);
extern void     Butterfly(double *, double *, int, double *, double *, int);

/* ------+--------+--------+--------+--------+--------+--------+------ */
/* armax1_call.c                                                      */
/* ------+--------+--------+--------+--------+--------+--------+------ */
extern void     Rpem_Routine(void);
extern int      Stable_Check(double *, int);

/* ------+--------+--------+--------+--------+--------+--------+------ */
/* armax1_pole.c                                                      */
```

```
/* ------+---------+---------+---------+---------+---------+---------- */
extern int      Pole_Calculation(int, double [], S_COMPLEX []);

/* ------+---------+---------+---------+---------+---------+---------- */
/* armax1_output.c                                                      */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void     Output_Result(FILE *, long);
extern void     Parameter_Writing(FILE *, int, int, double []);
extern void     Output_Pole(FILE *[]);
extern void     Output_PL(FILE *, long, int);

/* ------+---------+---------+---------+---------+---------+---------- */
/* armax1_util.c                                                        */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void     W_Writing(int, char (*)[256] , int, FILE *);
extern time_t   Now_Time_Set(char *, char *);

/* ------+---------+---------+---------+---------+---------+---------- */
/* armax1_math.c                                                        */
/* ------+---------+---------+---------+---------+---------+---------- */
extern double   Min_Calc(double, double);
extern int      Bairstow(double [], int, S_COMPLEX []);
extern void     Equation_2(double [], S_COMPLEX []);
extern void     Udl_to_P(int, double [][25], double [][25]);
```

B-6: PRT_fft.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                      _/
_/  Power Spectrum Density Calcuration Program by Direct FFT Method.    _/
_/                                                                      _/
_/                                                                      _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------ */

/* ====================================================================== */
/* extern define.                                                         */
/* ====================================================================== */
#ifdef FFT_MAIN
    #define EXT
#else
    #define EXT     extern
#endif

/* ====================================================================== */
/* varriable define.                                                      */
/* ====================================================================== */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* global.                                                                */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */

/* datafile parameter */
EXT S_DATAFILE              G_datafile;

/* FFT calculation parameter */
EXT S_FFT_CALC              G_fft_calc;

/* output file name and type */
EXT char                    G_out_filename[256];
EXT int                         G_out_filetype;
```

```
/* string array of reading parameter item */
#ifdef FFT_MAIN
char                    *G_parameter_item[] = {
                                "RECORD_LENGTH:::::::::::::::::::",
                                "DATA_FILENAME:::::::::::::::::::",
                                "NUMBER_OF_SKIPPING_DATA::::::::::",
                                "SAMPLING_FREQUENCY:::::::::::::::",
                                "1_BLOCK_NUMBER:::::::::::::::::::",
                                "OVERRAP_NUMBER:::::::::::::::::::",
                                "AVERAGING_NUMBER:::::::::::::::::",
                                "WINDOW_FUNCTION:::::::::::::::::::",
                                "SPECTRUM_TYPE:::::::::::::::::::::",
                                "DC_COMPONENT_FILTER:::::::::::::::",
                                "OUTPUT_TYPE:::::::::::::::::::::::",
                                "OUTPUT_FILENAME::::::::::::::::::" };
#else
extern char             *G_parameter_item[];
#endif

/* message buff array for W_Writing[in armax1_util.c] function */
EXT char                G_reportbuff[10][256];

/* file pointer for W_Writing[in armax1_util.c] function */
EXT FILE                *G_fout_report;

/* ------+---------+---------+---------+---------+---------+---------+------ */
#undef EXT
/* ------+---------+---------+---------+---------+---------+---------+------ */
```

B-7: VAR_fft.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Power Spectrum Density Calcuration Program by Direct FFT Method.        _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7--------- */

/* ========================================================================== */
/* standard prototype include.                                                */
/* ========================================================================== */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#include <time.h>


/* ========================================================================== */
/* macro define.                                                              */
/* ========================================================================== */

/* Routine check */
#define     SUCCESS         1
#define NOTHING         0
#define FAULT          -1

/* Switch */
#define ON              1
```

```
#define OFF             0

/* Window function type */
#define    HANNING          1
#define    HAMMING          2
#define BLACKMAN  3
#define AKAIKE          4

/* Spectrum type */
#define PSD             1
#define ESD             2

/* Output type */
#define PL3             1
#define NORMAL          2

/* Converged calculation factor */
#define EPS             1.0E-6
#define MAXLOOP         30000

/* Converge type */
#define CANNOTCALC      0
#define CONVERGE  1
#define NOCONVERGE      2

/* Display switch in W_Writing function */
#define CRT             0x0001
#define DISK            0x0002
#define CRT_DISK  0x0003

/* ================================================================== */
/* struct define.                                                     */
/* ================================================================== */
typedef struct {
     char  name[256];                  /* File name */
     int        byte;                       /* Data length */
     long  number;                       /* Number of the contained data */
     long  skip_number;                /* Skip number from the data head */
} S_DATAFILE;

typedef struct {
     double     sampling_frequency;          /* Sampling frequency of data */
     int        block_number;                /* block data size */
     int        max_block;                   /* max block */
     int        overrap;                     /* overrap number */
     int        averaging;                   /* averaging number */
     int        window;                           /* window function type */
     int        spectrum_type;               /* spectrum type [PSD or ESD] */
     int        dc_filter;                   /* dc component filter selection */
} S_FFT_CALC;

typedef struct  {
     double  real;                           /* real part of complex */
     double  image;                          /* imaginal part of complex */
} S_COMPLEX;

/* ================================================================== */
/* enum define.                                                       */
/* ================================================================== */
typedef enum     _tlog_login_pb_code {
     TLOG_LOGIN_EXEC=0,
     TLOG_LOGIN_MESSAGE,
     TLOG_LOGIN_QUIT
} TLOG_login_pb_code;
```

C Digital Filter Design プログラムリスト

C—1: digfil1_main.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/////////////////////////////////////////////////////////////////////////////
//                                                                           //
//  Digital Filter Design Program.                                           //
//                                                                           //
//                                                                           //
/////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------- */

/* ----+---------+---------+---------+---------+---------+---------+-------- */
/* MAIN FUNCTION SETTING.                                                    */
/* ----+---------+---------+---------+---------+---------+---------+-------- */
#define DIGFIL1_MAIN

/* ----+---------+---------+---------+---------+---------+---------+-------- */
/* standard include defines.                                                 */
/* ----+---------+---------+---------+---------+---------+---------+-------- */


/* ----+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                  */
/* ----+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_digfil1.h"
#include "VAR_digfil1.h"
#include "PRT_digfil1.h"


/* ----+---------+---------+---------+---------+---------+---------+-------- */
/* static variable defines.                                                  */
/* ----+---------+---------+---------+---------+---------+---------+-------- */
S_FILTER_DESIGN      filter_design;
S_DIGITAL_FILTER digital_filter;
S_TR_FUNCTION    Hs, Hz2;

/* ----+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : void main(int argc, char **argv)                           */
/*  prototype   : none                                                       */
/*  Function    : Main function for fft program.                             */
/*  Return Value: none                                                       */
/*  Variable    : argc  ; (i  ) Number of variable on the command line.      */
/*                **argv; (i  ) Pointer of strings on the command line.      */
/*。。Revision。。 : 2001/01/01                                               */
/* ----+---------+---------+---------+---------+---------+---------+-------- */
void  main(int argc, char **argv)
{

    int     check;


    if ( argc != 3 ) {
        Print_Usage();
        exit(0);
    }

    Title();

    if ( (G_fout_report = fopen(argv[2], "wt")) == NULL ) {
        printf( "Fatal Error: Cannot open the report file [%s]!! ¥n", argv[2] );
        exit(0);
```

```
    }

    check = AutoMain(argv);

    if ( check == SUCCESS ) {
        sprintf(G_reportbuff[0],
        "========================== NORMAL TERMINATION ==========================¥n¥n");
    } else {
        sprintf(G_reportbuff[0],
        "========================== ABNORMAL TERMINATION ==========================¥n¥n");
    }

    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    fclose(G_fout_report);
}



/* ------+---------+---------+---------+---------+---------+---------+--------- */
/* Format      : void Print_Usage(void)                                          */
/* prototype   : PRT_digfil1.h                                                   */
/* Function    : Display Usage                                                   */
/* Return Value: none                                                            */
/* Variable    : none                                                            */
/*。。Revision。。 : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void  Print_Usage(void) {

    printf( "Usage:¥n" );
    printf( "  digfil1.out  control_filename  report_filename¥n" );
    printf( "            control_filname: path of the control file.¥n" );
    printf( "            report_filnema : path of the report file.¥n¥n" );

}



/* ------+---------+---------+---------+---------+---------+---------+--------- */
/* Format      : void Title(void)                                                */
/* prototype   : PRT_digfil1.h                                                   */
/* Function    : Display Title                                                   */
/* Return Value: none                                                            */
/* Variable    : none                                                            */
/*。。Revision。。 : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void  Title(void) {

    printf("¥n¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("_/                                                                 _/¥n");
    printf("_/ DIGFIL1: DEGITAL FILTER DESIGN PROGRAM.                         _/¥n");
    printf("_/                                                                 _/¥n");
    printf("_/     Version 1.01 : 2001/01/01 : transplantation from PC version.  _/¥n");
    printf("_/                                                                 _/¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("¥n¥n");

}
```

```
/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : int AutoMain(void)                                           */
/*  prototype   : PRT_digfil1.h                                                */
/*  Function    : Execution Control                                            */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil1.h]   */
/*                abnormal termination -> FAULT   [defined in DFN_digfil1.h]   */
/*  Variable    : **argv; (i  ) Pointer of strings on the command line.        */
/*。。Revision。。 : 2001/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
int  AutoMain(char **argv)
{
    FILE        *fin_control;
    char        setup_filename[256];
    int         count, check;
    char        answer;


    /* Initialize */
    count = 0;

    /* Job control file open */
    if ( (fin_control = fopen(argv[1], "rt")) == NULL) {
        printf( "Fatal Error: Cannot open the control file [%s]!! ¥n", argv[1] )
;
        return FAULT;
    }


    /* Roop process */
    while ( !feof(fin_control) ) {
        count++;
        fscanf(fin_control, "%s", setup_filename);
        if ( feof(fin_control) ) {
            break;
        }

        sprintf( G_reportbuff[0], "¥n¥n" );
        sprintf( G_reportbuff[1],
            "--- Digital Filter Design ----------------------------------------------------------------------¥n");
       sprintf( G_reportbuff[2], "JOB No. %d¥n", count );
       sprintf( G_reportbuff[3], "Setup File Name = %s¥n¥n", setup_filename );
       W_Writing( CRT_DISK, G_reportbuff, 4, G_fout_report );

       if ( ReadFileCall(setup_filename, &filter_design) == FAULT) {
           continue;
       }

        switch ( filter_design.form ) {
            case IIR:
                check = IIR_Transfer_Function( &filter_design,
                                               &digital_filter,
                                               &Hs,
                                               &Hz2);
                if ( check == SUCCESS ) {
                    Output_Call( &digital_filter, &Hz2);
                }
                break;
            case FIR:
            sprintf( G_reportbuff[0], "FIR filter is not in service.¥n" );
            W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
                break;
        }
    }
    fclose(fin_control);
```

```
    return check;
}
```

C-2: digfil1_read.c

```
/* ----1--------2--------3--------4--------5--------6--------7--------8
/////////////////////////////////////////////////////////////////////////
/                                                                       /
/  Digital Filter Design Program.                                       /
/                                                                       /
/                                                                       /
/////////////////////////////////////////////////////////////////////////
--------1--------2--------3--------4--------5--------6--------7------- */


/* ------+--------+--------+--------+--------+--------+--------+------- */
/* standard include defines.                                           */
/* ------+--------+--------+--------+--------+--------+--------+------- */


/* ------+--------+--------+--------+--------+--------+--------+------- */
/* private include defines.                                            */
/* ------+--------+--------+--------+--------+--------+--------+------- */
#include "DFN_digfil1.h"
#include "VAR_digfil1.h"
#include "PRT_digfil1.h"


/* ------+--------+--------+--------+--------+--------+--------+------- */
/*  Format      : int ReadFileCall(char *filename, S_FILTER_DESIGN *s_filter) */
/*  Prototype   : PRT_digfil11.h                                       */
/*  Discription : Reading job file and setting parameter.              */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil1.h] */
/*                abnormal termination -> FAULT   [defined in DFN_digfil1.h] */
/*  Variable    : *filename; (i  ) pointer to filenmae buffer.         */
/*                *s_filter; ( o) pointer to struction contained the filter */
/*                           design.                                  */
/*。。Revision。。 : 2001/01/01                                        */
/* ------+--------+--------+--------+--------+--------+--------+------- */
int  ReadFileCall(char *filename, S_FILTER_DESIGN *s_filter)
{
    FILE        *fin_setupfile, *fout_test;
    S_FILTER_DESIGN  filter;
    char         input_buff[256];
    int          i;


    /* Setup file open */
    if ( (fin_setupfile = fopen(filename, "rt")) == NULL ) {
        sprintf(G_reportbuff[0], "Cannot open setup file [ %s ].¥n", filename);
        sprintf(G_reportbuff[1], "Abort this job.¥n", filename);
        W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);
        return FAULT;
    }


    /* Initialize */
    memset(&filter,       0x00, sizeof(S_FILTER_DESIGN));
    memset(G_out_filename,  0x00, 256);


    /* Reading parameter in setup file */
```

```
for ( i = 0; i < 11; i++ ) {
    if ( Std_Reading(fin_setupfile, input_buff, i) == FAULT ) {
        sprintf(G_reportbuff[0], "Error: Format of setup file is unsuitable.¥n");
        goto ERROR_RECOVERY;
    }
    switch ( i ) {

        case 0: /* Reading filter form */
            if ( strcmp(input_buff, "IIR") == 0 ) {
                filter.form = IIR;
            } else if ( strcmp( input_buff, "FIR" ) == 0 ) {
                filter.form = FIR;
            } else {
                sprintf(G_reportbuff[0],
                        "Illegal selection as filter form.¥n");
                goto ERROR_RECOVERY;
            }
            break;

        case 1: /* Reading filter type */
            if ( strcmp(input_buff, "LPF") == 0 ) {
                filter.type = LPF;
            } else if ( strcmp( input_buff, "HPF" ) == 0 ) {
                filter.type = HPF;
            } else if ( strcmp( input_buff, "BPF" ) == 0 ) {
                filter.type = BPF;
            } else if ( strcmp( input_buff, "BRF" ) == 0 ) {
                filter.type = BRF;
            } else {
                sprintf(G_reportbuff[0],
                        "Illegal selection as filter type.¥n");
                goto ERROR_RECOVERY;
            }
            break;

        case 2: /* Reading anplitude characteristic */
            if ( strcmp(input_buff, "BUTTERWORTH") == 0 ) {
                filter.a_character = BUTTERWORTH;
            } else if ( strcmp( input_buff, "CHEBYSHEV" ) == 0 ) {
                filter.a_character = CHEBYSHEV;
            } else if ( strcmp( input_buff, "INVERSE_CHEBYSHEV" ) == 0 ) {
                filter.a_character = INVERSE_CHEBYSHEV;
            } else {
                sprintf(G_reportbuff[0],
                        "Illegal selection as anplitude characteristic.¥n");
                goto ERROR_RECOVERY;
            }
            break;

        case 3: /* Reading sampling frequency */
            filter.sampling_frequency = atof(input_buff);
            if ( filter.sampling_frequency <= 0.0 ) {
                sprintf(G_reportbuff[0],
                        "Illegal value as sampling frequency.¥n");
                goto ERROR_RECOVERY;
            }
            break;

        case 4: /* Reading 1st cutoff frequency */
            filter.cutoff_frequency_1 = atof(input_buff);
            if ( filter.cutoff_frequency_1 <= 0.0 ) {
                sprintf(G_reportbuff[0],
                        "Illegal value as 1st cutoff frequency.¥n");
                goto ERROR_RECOVERY;
```

```
        }
        break;

    case 5: /* Reading 1st stopping frequency */
        filter.stopping_frequency_1 = atof(input_buff);
        if ( filter.stopping_frequency_1 <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as 1st stopping frequency.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 6: /* Reading 2nd cutoff frequency */
        if ( filter.type == LPF || filter.type == HPF ) {
            filter.cutoff_frequency_2 = INEFFECTIVE;
            break;
        }
        filter.cutoff_frequency_2 = atof(input_buff);
        if ( filter.cutoff_frequency_2 <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as 2nd cutoff frequency.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 7: /* Reading 2nd stopping frequency */
        if ( filter.type == LPF || filter.type == HPF ) {
            filter.stopping_frequency_2 = INEFFECTIVE;
            break;
        }
        filter.stopping_frequency_2 = atof(input_buff);
        if ( filter.stopping_frequency_2 <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as 2nd stopping frequency.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 8: /* Reading pass ripple */
        filter.pass_ripple = atof(input_buff);
        if ( filter.pass_ripple <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as pass ripple.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 9: /* Reading rejection power */
        filter.rejection_power = atof(input_buff);
        if ( filter.rejection_power <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as rejection power.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 10: /* Reading output filename of result */
        if ( (fout_test = fopen(input_buff, "wt")) == NULL ) {
            sprintf(G_reportbuff[0], "Unsuitable Output File Name.¥n");
            goto ERROR_RECOVERY;
        }
        fclose(fout_test);
        strcpy(G_out_filename, input_buff);
        break;
```

```
        }
    }

    /* Data aveilability check */
    if ( Design_Aveilability_Check(&filter) > 0 ) {
        sprintf(G_reportbuff[0], "There are some contradictions in design.¥n");
        goto ERROR_RECOVERY;
    }

    *s_filter = filter;

    fclose(fin_setupfile);
    return SUCCESS;


ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    fclose(fin_setupfile);
    return FAULT;
}



/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : int Std_Reading(FILE *fin, char *return_buff, int counter)  */
/*  Prototype   : PRT_digfil1.h                                               */
/*  Discription : Standard reading function of setup file.                    */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil1.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_digfil1.h]  */
/*  Variable    : *fin         ; (i  ) file pointer of setup file.            */
/*                *return_buff; (  o) string buffer pointer for reading para- */
/*                                    meter.                                 */
/*                counter      ; (i  ) logical ID of parameter.              */
/*  ..Revision..  : 2001/01/01                                                */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  Std_Reading(FILE *fin, char *ptr1, int counter)
{

    char tmp_buff1[256];
    int      check;


    check = fscanf(fin, "%s %s", tmp_buff1, ptr1);
    sprintf( G_reportbuff[0], "%s %s¥n", tmp_buff1, ptr1 );
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    if ( (check != 2 ) || (stromp( tmp_buff1, G_parameter_item[counter]) != 0) ) {
        return FAULT;
    }

    return SUCCESS;
}



/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : int Design_Aveilability_Check(S_FILTER_DESIGN *filter)      */
/*  Prototype   : PRT_digfil1.h                                               */
/*  Discription : Filter design aveilability check.                           */
/*  Return Value: number of contradiction                                     */
/*  Variable    : *filter; (i  ) pointer of construction contained filter     */
/*                               design parameter.                            */
/*  ..Revision..  : 2001/01/01                                                */
```

```
/* ------+---------+---------+---------+---------+---------+---------+-------- */
int  Design_Aveilability_Check(S_FILTER_DESIGN *filter)
{

    int     i, count;
    int     warning[10];
    double  max_frequency;


    static char *err_string[] = {
                "[FATAL CONTRADICTION] inversion with pass ripple factor and rejection power factor.¥n"
                "[FATAL CONTRADICTION] 1st stopping frequency is larger than Nyquist frequency.¥n",
                "[FATAL CONTRADICTION] 1st cutoff frequency is larger than 1st stopping frequency.¥n",
                "[FATAL CONTRADICTION] 1st cutoff frequency is larger than Nyquist frequency.¥n",
                "[FATAL CONTRADICTION] 1st cutoff frequency is smaller than 1st stopping frequency.¥n",
                "[FATAL CONTRADICTION] 2nd cutoff frequency is larger than Nyquist frequency.¥n",
                "[FATAL CONTRADICTION] 1st cutoff frequency is larger than 2nd cutoff frequency.¥n",
                "[FATAL CONTRADICTION] 1st stopping frequency is larger than 2nd stopping frequency.¥n",
                "[FATAL CONTRADICTION] 2nd cuttoff frequency is larger than 2nd stopping frequency.¥n",
                "[FATAL CONTRADICTION] 2nd cutoff frequency is smaller than 2nd stopping frequency.¥n" };


    max_frequency = filter->sampling_frequency / 2.0;
    for (i = 0; i < 9; i++) {
        warning[i] = OFF;
    }

    if (filter->pass_ripple >= filter->rejection_power) {
        warning[0] = ON;
    }

    switch (filter->type) {
        case LPF :
            if (filter->stopping_frequency_1 > max_frequency) {
                warning[1] = ON;
            }
            if (filter->cutoff_frequency_1 >= filter->stopping_frequency_1) {
                warning[2] = ON;
            }
            break;
        case HPF :
            if (filter->cutoff_frequency_1 > max_frequency) {
                warning[3] = ON;
            }
            if (filter->cutoff_frequency_1 <= filter->stopping_frequency_1) {
                warning[4] = ON;
            }
            break;
        case BPF :
            if (filter->stopping_frequency_2 > max_frequency) {
                warning[5] = ON;
            }
            if (filter->cutoff_frequency_1 >= filter->cutoff_frequency_2) {
                warning[6] = ON;
            }
            if (filter->stopping_frequency_1 >= filter->stopping_frequency_2) {
                warning[7] = ON;
            }
            if (filter->cutoff_frequency_1 <= filter->stopping_frequency_1) {
                warning[4] = ON;
            }
            if (filter->cutoff_frequency_2 >= filter->stopping_frequency_2) {
                warning[8] = ON;
```

```
            }
            break;
        case BRF :
            if (filter->cutoff_frequency_2 > max_frequency) {
                warning[5] = ON;
            }
            if (filter->cutoff_frequency_1 >= filter->cutoff_frequency_2) {
                warning[6] = ON;
            }
            if (filter->stopping_frequency_1 >= filter->stopping_frequency_2) {
                warning[7] = ON;
            }
            if (filter->cutoff_frequency_1 >= filter->stopping_frequency_1) {
                warning[2] = ON;
            }
            if (filter->cutoff_frequency_2 <= filter->stopping_frequency_2) {
                warning[9] = ON;
            }
            break;
    }


    count = 0;
    for (i = 0; i <= 9; i++) {
        if (warning[i] == ON) {
            strcpy( G_reportbuff[count], err_string[i]);
            count++;
        }
    }
    if ( count > 0 ) {
        W_Writing( CRT_DISK, G_reportbuff, count, G_fout_report );
    }

    return count;
}
```

C-3 digfil1_IIR.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                              */
/*  Digital Filter Design Program.                                              */
/*                                                                              */
/*                                                                              */
/*                                                                              */
/*--------1---------2---------3---------4---------5---------6---------7-------- */

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* standard include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                     */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_digfil1.h"
#include "VAR_digfil1.h"
#include "PRT_digfil1.h"


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* static variable defines.                                                     */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
```

```
/* ------+--------+--------+--------+--------+--------+------ */
/* Format      : int IIR_Transfer_Function(S_FILTER_DESIGN *filter,          */
/*                                         S_DIGITAL_FILTER *digital,       */
/*                                         S_TR_FUNCTION *Hs,               */
/*                                         S_TR_FUNCTION *Hz2)              */
/* prototype   : none                                                       */
/* Function    : Main Calculation process for solving IIR Transfer Function. */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil1.h] */
/*               abnormal termination -> FAULT   [defined in DFN_digfil1.h] */
/* Variable    : *filter ; (i  ) Pointer of structure contained the filter  */
/*                               design.                                    */
/*               *digital; (  o) Pointer of structure contained the digital */
/*                               filter component.                          */
/*               *Hs     ; (  o) pointer of structure contained the transfer */
/*                               function.                                  */
/*               *Hz2    ; (  o) pointer of structure contained the transfer */
/*                               function.                                  */
/*。。Revision。。 : 2001/01/01                                              */
/* ------+--------+--------+--------+--------+--------+------ */
int  IIR_Transfer_Function(S_FILTER_DESIGN *filter,
                           S_DIGITAL_FILTER *digital,
                           S_TR_FUNCTION *Hs,
                           S_TR_FUNCTION *Hz2)
{

    S_PREWARP               prewarped_omega;
    S_COMPLEX               pole;
    double                  epsilon, epsilon2;
    int                     i, dimension;


    Prewarping(filter, &prewarped_omega);


    switch ( filter->type ) {
      ` case BUTTERWORTH :
            dimension = Butterworth_Dimension(filter, &prewarped_omega);
            sprintf( G_reportbuff[0], "¥nThe dimension will be %d order.¥n¥n", dimension );
            W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
            if (dimension > 30) {
                sprintf( G_reportbuff[0], "It is too high dimension. So, the process has been aborted.¥n" );
                W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
                return FAULT;
            }
            digital->dimension = dimension;
            epsilon2 = pow(10.0, (filter->pass_ripple / 10.0)) - 1.0;
            epsilon  = sqrt(epsilon2);
            Butterworth_Normal_LPF_Pole(digital->dimension, epsilon, &pole);
            switch (filter->type) {
                case LPF :
                    Butterworth_LPF(digital->dimension,
                                    epsilon,
                                    prewarped_omega.cutoff_1,
                                    &pole,
                                    Hs);
                    break;
                case HPF :
                    Butterworth_HPF(digital->dimension,
                                    epsilon,
                                    prewarped_omega.cutoff_1,
                                    &pole,
                                    Hs);
```

```
                break;
            case BPF :
            case BRF :
                sprintf( G_reportbuff[0], "BPF and BRF are not in service.¥n" );
                W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

                break;
        }
        Bilinear_Transformation(digital->dimension,
                                filter->sampling_frequency,
                                Hs,
                                Hz2);
        break;
    case CHEBYSHEV :
    case INVERSE_CHEBYSHEV :
        sprintf( G_reportbuff[0], "CHEBYSEV and INVERSE CHEBYSHEV are not in service.¥n" );
        W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
        break;
}


Direct_Method(digital, Hz2);


digital->sampling_frequency = filter->sampling_frequency;

digital->gain1 = Hz2->gain;
digital->gain2 = digital->gain1 * digital->n[0];

for (i = 0; i <= 30; i++) {
    digital->n2[i] = digital->n[i] / digital->n[0];
}

return SUCCESS;
}


/* ------+---------+---------+---------+---------+---------+------- */
/* Format      : void Prewarping(S_FILTER_DESIGN *filter,            */
/*                               S_PREWARP *prewarped_omega)         */
/* prototype   : PRT_digfil1.h                                       */
/* Function    : Prewarping calculation.                             */
/* Return Value: none                                                */
/* Variable    : *filter         ; (i  ) Pointer of structure contained the  */
/*                                       filter design.              */
/*               *prewarped_omega; (  o) Pointer of structure contained the  */
/*                                       prewarped omega factor.     */
/*..Revision..  : 2001/01/01                                        */
/* ------+---------+---------+---------+---------+---------+------- */
void Prewarping(S_FILTER_DESIGN *filter, S_PREWARP *prewarped_omega)
{
    double  dummy_d1, dummy_d2;


    dummy_d1 = 2*filter->sampling_frequency;
    dummy_d2 = M_PI/filter->sampling_frequency;

    prewarped_omega->cutoff_1 =
                dummy_d1 * tan(filter->cutoff_frequency_1 * dummy_d2);

    prewarped_omega->stopping_1 =
                dummy_d1 * tan(filter->stopping_frequency_1 * dummy_d2);
```

```
    prewarped_omega->cutoff_2 =
                dummy_d1 * tan(filter->cutoff_frequency_2 * dummy_d2);

    prewarped_omega->stopping_2 =
                dummy_d1 * tan(filter->stopping_frequency_2 * dummy_d2);

}



/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Butterworth_Dimension(S_FILTER_DESIGN *filter,     */
/*                                          S_PREWARP *prewarped_omega)  */
/*  prototype   : PRT_digfil1.h                                          */
/*  Function    : Introduce of the dimension of butterworth filter.      */
/*  Return Value: the dimension.                                         */
/*  Variable    : *filter         ; (i  ) Pointer of structure contained the */
/*                                        filter design.                 */
/*                *prewarped_omega; (i  ) Pointer of structure contained the */
/*                                        prewarped omega factor.        */
/*..Revision..  : 2001/01/01                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
int  Butterworth_Dimension(S_FILTER_DESIGN *filter, S_PREWARP *prewarped_omega)
{

    double   omega;
    int      dimension;


    switch (filter->type) {
        case LPF :
            omega = prewarped_omega->cutoff_1 /
                                        prewarped_omega->stopping_1;
            dimension = Butterworth_L_H_Dim(omega, filter->pass_ripple,
                                                   filter->rejection_power);
            break;
        case HPF :
            omega = prewarped_omega->stopping_1 /
                                        prewarped_omega->cutoff_1;
            dimension = Butterworth_L_H_Dim(omega, filter->pass_ripple,
                                                   filter->rejection_power);
            break;
        case BPF :
        case BRF :
            break;
    }

    return dimension;
}



/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Butterworth_L_H_Dim(double omega,                  */
/*                                        double pass,                   */
/*                                        double rejection)              */
/*  prototype   : PRT_digfil1.h                                          */
/*  Function    : Calculating the dimension of LPF or HPF characterized  */
/*                by butterworth.                                        */
/*  Return Value: the calcuated dimension.                               */
/*  Variable    : opmega    ; (i  ) omega factor. (see superior fanction.) */
/*                pass      ; (i  ) pass ripple.                       */
/*                rejection; (i  ) rejection power.                    */
/*..Revision..  : 2001/01/01                                           */
```

```
/* ------+--------+--------+--------+--------+--------+-------- */
int  Butterworth_L_H_Dim(double omega, double pass, double rejection)
{

    int     dimension;
    double  k, k1, bunshi, bunbo, dummy_d1;


    bunshi = pass / 10.0;
    bunbo  = rejection / 10.0;
    dummy_d1 = (pow(10.0, bunshi) - 1.0) / (pow(10.0, bunbo) - 1.0);
    k1 = log10(dummy_d1);
    k1 /= 2.0;
    k  = log10(omega);

    dummy_d1 = k1 / k;
    dummy_d1 = ceil(dummy_d1);

    dimension = (int)(dummy_d1);

    return dimension;
}
```

```
/* ------+--------+--------+--------+--------+--------+-------- */
/*  Format      : void Butterworth_Normal_LPF_Pold(int dimension,          */
/*                                                 double epsilon,        */
/*                                                 S_COMPLEX *pole)       */
/*  prototype   : PRT_digfil1.h                                            */
/*  Function    : Calculating the pole of butterworth normal LPF.          */
/*  Return Value: none.                                                    */
/*  Variable    : dimension; (i  ) filter dimension.                       */
/*                epsilon  ; (i  ) epsioin factor.                         */
/*                *pole    ; ( o) pointer of structure contained the pole. */
/*。。Revision。。 : 2001/01/01                                             */
/* ------+--------+--------+--------+--------+--------+-------- */
void Butterworth_Normal_LPF_Pole(int dimension, double epsilon, S_COMPLEX *pole)
{
    int     i, numerator, denominator;
    double  theta, special_epsilon, dummy_d1;


    dummy_d1 = -1.0 / (double)dimension;
    special_epsilon = pow(epsilon, dummy_d1);
    denominator = 2 * dimension;

    for (i = 1; i <= dimension; i++) {
        numerator = 2*i - 1;
        dummy_d1 = (double)numerator / (double)denominator;
        theta = dummy_d1 * M_PI;
        pole->real[i]  = -(sin(theta)) * special_epsilon;
        pole->imagi[i] = cos(theta) * special_epsilon;
    }

}
```

```
/* ------+--------+--------+--------+--------+--------+-------- */
/*  Format      : void Butterworth_LPF(int n,                              */
/*                                     double epsilon,                     */
/*                                     double wc,                          */
```

```
/*                                      S_COMPLEX *pole,                    */
/*                                      S_TR_FUNCTION *Hs)                  */
/*  prototype    : PRT_digfil1.h                                            */
/*  Function     : Calculating the transfer function of butterworth normal LPF.*/
/*  Return Value: none.                                                     */
/*  Variable     : n       ; (i  ) filter dimension.                        */
/*                 epsilon; (i  ) epsioln factor.                           */
/*                 wc      ; (i  ) 1st cutoff frequency after prewarping.   */
/*                 *pole   ; (i  ) pointer of structure contained the pole. */
/*                 *Hs     ; ( o) pointer of structure contained the transfer */
/*                           function.                                      */
/*。。Revision。。 : 2001/01/01                                               */
/* ------+---------+---------+---------+---------+---------+---------+------ */
void Butterworth_LPF(int n,
                     double epsilon,
                     double wc,
                     S_COMPLEX *pole,
                     S_TR_FUNCTION *Hs)
{

    int     i, nn;
    double  special_epsilon, special_epsilon2, wc2, dummy_d1;


    wc2 = SQR(wc);

    dummy_d1 = -1.0 / (double)n;
    special_epsilon = pow(epsilon, dummy_d1);
    special_epsilon2 = SQR(special_epsilon);

    switch (Even_Odd_Check(n)) {
        case EVEN :
            nn = n / 2;
            for (i = 1; i <= nn; i++) {
                Hs->d[i][0] = 1.0;
                Hs->d[i][1] = -(2 * wc * pole->real[i]);
                Hs->d[i][2] = wc2 * special_epsilon2;
                Hs->n[i][0] = 0.0;
                Hs->n[i][1] = 0.0;
                Hs->n[i][2] = wc2 * special_epsilon2;
            }
            break;
        case ODD :
            nn = (n - 1) / 2;
            for (i = 1; i <= nn; i++) {
                Hs->d[i][0] = 1.0;
                Hs->d[i][1] = -(2 * wc * pole->real[i]);
                Hs->d[i][2] = wc2 * special_epsilon2;
                Hs->n[i][0] = 0.0;
                Hs->n[i][1] = 0.0;
                Hs->n[i][2] = wc2 * special_epsilon2;
            }
            Hs->d[nn+1][0] = 0.0;
            Hs->d[nn+1][1] = 1.0;
            Hs->d[nn+1][2] = wc * special_epsilon;
            Hs->n[nn+1][0] = 0.0;
            Hs->n[nn+1][1] = 0.0;
            Hs->n[nn+1][2] = wc * special_epsilon;
            break;
    }

}
```

```
/* ------+-------+-------+-------+-------+-------+------- */
/*  Format      : void Butterworth_HPF(int n,                */
/*                                     double epsilon,       */
/*                                     double wc,            */
/*                                     S_COMPLEX *pole,      */
/*                                     S_TR_FUNCTION *Hs)    */
/*  prototype   : PRT_digfilt.h                              */
/*  Function    : Calculating the transfer function of butterworth normal HPF.*/
/*  Return Value: none.                                      */
/*  Variable    : n       ; (i  ) filter dimension.          */
/*                epsilon; (i  ) epsioln factor.             */
/*                wc      ; (i  ) 1st cutoff frequency after prewarping.*/
/*                *pole   ; (i  ) pointer of structure contained the pole.*/
/*                *Hs     ; ( o) pointer of structure contained the transfer*/
/*                                function.                  */
/*  ..Revision..  : 2001/01/01                               */
/* ------+-------+-------+-------+-------+-------+------- */
void Butterworth_HPF(int n,
                     double epsilon,
                     double wc,
                     S_COMPLEX *pole,
                     S_TR_FUNCTION *Hs)
{

    int     i, nn;
    double  special_epsilon, special_epsilon2, wc2, dummy_d1;


    wc2 = SQR(wc);

    dummy_d1 = 1.0 / (double)n;
    special_epsilon = pow(epsilon, dummy_d1);
    special_epsilon2 = SQR(special_epsilon);

    switch (Even_Odd_Check(n)) {
        case EVEN :
            nn = n / 2;
            for (i = 1; i <= nn; i++) {
                Hs->d[i][0] = 1.0;
                Hs->d[i][1] = -(2 * wc * pole->real[i] * special_epsilon2);
                Hs->d[i][2] = wc2 * special_epsilon2;
                Hs->n[i][0] = 1.0;
                Hs->n[i][1] = 0.0;
                Hs->n[i][2] = 0.0;
            }
            break;
        case ODD :
            nn = (n - 1) / 2;
            for (i = 1; i <= nn; i++) {
                Hs->d[i][0] = 1.0;
                Hs->d[i][1] = -(2 * wc * pole->real[i] * special_epsilon2);
                Hs->d[i][2] = wc2 * special_epsilon2;
                Hs->n[i][0] = 1.0;
                Hs->n[i][1] = 0.0;
                Hs->n[i][2] = 0.0;
            }
            Hs->d[nn+1][0] = 0.0;
            Hs->d[nn+1][1] = 1.0;
            Hs->d[nn+1][2] = wc * special_epsilon;
            Hs->n[nn+1][0] = 0.0;
            Hs->n[nn+1][1] = 1.0;
            Hs->n[nn+1][2] = 0.0;
            break;
```

```
    }

}


/* ------+--------+--------+--------+--------+--------+------ */
/*  Format      : void Bilinear_Transformation(int n,          */
/*                                             double fs,      */
/*                                             S_TR_FUNCTION *Hs)      */
/*                                             S_TR_FUNCTION *Hz2)     */
/*  prototype   : PRT_digfil1.h                                */
/*  Function    : Executing the bilinear transformation.       */
/*  Return Value: none.                                        */
/*  Variable    : n   ; (i  ) filter dimension.                */
/*                fs  ; (i  ) sampling frequency.              */
/*                *Hs ; (i  ) pointer of structure contained the transfer  */
/*                            function.                        */
/*                *Hz2; ( o) pointer of structure contained the bilineared */
/*                            transfer function.               */
/*。。Revision。。 : 2001/01/01                                 */
/* ------+--------+--------+--------+--------+--------+------ */
void Bilinear_Transformation(int n,
                             double fs,
                             S_TR_FUNCTION *Hs,
                             S_TR_FUNCTION *Hz2)
{
    int     i, nn, check_dimension;
    double  fs2, gain, total_gain, dummy_d1;


    fs *= 2.0;
    fs2 = SQR(fs);
    check_dimension = Even_Odd_Check(n);

    switch (check_dimension) {
        case EVEN :
            nn = n / 2;
            break;
        case ODD :
            nn = (n - 1) / 2;
            break;
    }


    total_gain = 1.0;

    for (i = 1; i <= nn; i++) {
        gain = 1.0 / (fs2*Hs->d[i][0] + fs*Hs->d[i][1] + Hs->d[i][2]);
        total_gain *= gain;
        Hz2->d[i][0] = 1.0;
        Hz2->d[i][1] = 2.0 * gain * (Hs->d[i][2] - fs2*Hs->d[i][0]);
        Hz2->d[i][2] = gain *
                       (fs2*Hs->d[i][0] - fs*Hs->d[i][1] + Hs->d[i][2]);
        dummy_d1 = fs2 * Hs->n[i][0];
        Hz2->n[i][0] = dummy_d1 + fs*Hs->n[i][1] + Hs->n[i][2];
        Hz2->n[i][1] = 2 * (Hs->n[i][2] - dummy_d1);
        Hz2->n[i][2] = dummy_d1 - fs2*Hs->n[i][1] + Hs->n[i][2];
    }


    if (check_dimension == ODD) {
        gain = 1.0 / (fs*Hs->d[i][1] + Hs->d[nn+1][2]);
        total_gain *= gain;
```

```
        Hz2->d[nn+1][0] = 1.0;
        Hz2->d[nn+1][1] = gain * (Hs->d[nn+1][2] - fs*Hs->d[nn+1][1]);
        Hz2->d[nn+1][2] = 0.0;
        dummy_d1 = fs * Hs->n[nn+1][1];
        Hz2->n[nn+1][0] = dummy_d1 + Hs->n[nn+1][2];
        Hz2->n[nn+1][1] = Hs->n[nn+1][2] - dummy_d1;
        Hz2->n[nn+1][2] = 0.0;
    }

    Hz2->gain = total_gain;

}


/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : void Direct_Method(S_DIGITAL_FILTER *digital,          */
/*                                  S_TR_FUNCTION *Hz2 )                */
/* prototype   : PRT_digfil1.h                                          */
/* Function    : Executing the direct method.                           */
/* Return Value: none.                                                  */
/* Variable    : *digital: (i/o) pointer of structure contained digital */
/*                               filter component.                      */
/*               *Hz2    : (i  ) pointer of structure contained the     */
/*                               bilineared transfer function.          */
/*。。Revision。。 : 2001/01/01                                          */
/* ------+---------+---------+---------+---------+---------+---------- */
void Direct_Method(S_DIGITAL_FILTER *digital, S_TR_FUNCTION *Hz2)
{
    int     n, nn, check_dimension;


    n = digital->dimension;
    check_dimension = Even_Odd_Check(n);

    switch (check_dimension) {
        case EVEN :
            nn = n / 2;
            break;
        case ODD :
            nn = (n + 1) / 2;
            break;
    }

    Expansion(nn, Hz2->d, digital->d);

    Expansion(nn, Hz2->n, digital->n);

}


/* ------+---------+---------+---------+---------+---------+---------- */
/* Format      : void Expansion(int nn, double Hz[][3], double tr_func[])*/
/* prototype   : PRT_digfil1.h                                          */
/* Function    : Expansion process for direct method.                   */
/* Return Value: none.                                                  */
/* Variable    : nn       ; (i  ) filter dimension.                     */
/*               Hz[][3]  ; (i  ) pointer of two dimensional array contained */
/*                                tarnsfer function component.          */
/*               tr_func[]; (  o) pointer of array contained the last   */
```

```
/*                                  transfer function.                          */
/*..Revision..  : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
void Expansion(int nn, double Hz[][3], double tr_func[])
{

    double  left[31], right[3];
    int     i, j, k, left_order, right_order;



    for (i = 0; i <= 30; i++)
        left[i] = 1.0;


    left_order  = 0;
    right_order = 2;


    for (i = 1; i <= nn; i++) {

        for (j = 0; j <= 30; j++)
            tr_func[j] = 0.0;

        for (j = 0; j <= 2; j++)
            right[j] = Hz[i][j];

        for (j = 0; j <= left_order; j++)
            for (k = 0; k <= right_order; k++)
                tr_func[j+k] += (left[j] * right[k]);

        left_order += right_order;
        for (j = 0; j <= left_order; j++)
            left[j] = tr_func[j];
    }
}
```

```
/* ------+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : int Even_Odd_Check(int n)                                      */
/*  prototype   : PRT_digfil1.h                                                  */
/*  Function    : Return parity the given value.                                 */
/*  Return Value: n is evne  -> EVEN[defined in DFN_digfil1.h]                   */
/*                n is odd   -> ODD [defined in DFN_digfil1.h]                   */
/*  Variable    : n; (i  ) evaluation value.                                     */
/*..Revision..  : 2001/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
int  Even_Odd_Check(int n)
{
    if ((n % 2) == 0)
        return EVEN;
    else
        return ODD;
}
```

C-4 digfil1_output.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
/_                                                                            _/
/_  Digital Filter Design Program.                                            _/
/_                                                                            _/
```

```
/*                                                                    */
/*--------1---------2---------3---------4---------5---------6---------7--------*/

/*--------+---------+---------+---------+---------+---------+---------+--------*/
/* standard include defines.                                          */
/*--------+---------+---------+---------+---------+---------+---------+--------*/


/*--------+---------+---------+---------+---------+---------+---------+--------*/
/* private include defines.                                           */
/*--------+---------+---------+---------+---------+---------+---------+--------*/
#include "DFN_digfil1.h"
#include "VAR_digfil1.h"
#include "PRT_digfil1.h"


/*--------+---------+---------+---------+---------+---------+---------+--------*/
/* static variable defines.                                           */
/*--------+---------+---------+---------+---------+---------+---------+--------*/

/*--------+---------+---------+---------+---------+---------+---------+--------*/
/*  Format      : void main(int argc, char **argv)                    */
/*  prototype   : none                                                */
/*  Function    : Main function for fft program.                      */
/*  Return Value: none                                                */
/*  Variable    : argc  ; (i  ) Number of variable on the command line.*/
/*                **argv; (i  ) Pointer of strings on the command line.*/
/*..Revision..  : 2001/01/01                                          */
/*--------+---------+---------+---------+---------+---------+---------+--------*/
void Output_Call(S_DIGITAL_FILTER *digital, S_TR_FUNCTION *Hz2)
{

    FILE*fout;

    Output_1(digital);
    sprintf(G_reportbuff[0],"¥n¥n");
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    Output_2(digital);
    sprintf(G_reportbuff[0],"¥n¥n");
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    Output_3(digital->dimension, Hz2);
    fout = fopen(G_out_filename, "wt");
    File_Output(fout, digital);
    fclose(fout);
}

/*--------+---------+---------+---------+---------+---------+---------+--------*/
/*  Format      : void Output_1(S_DIGITAL_FILTER *digital)             */
/*  prototype   : PTR_digfil1.h                                       */
/*  Function    : Result output by formation type 1.                  */
/*  Return Value: none                                                */
/*  Variable    : *digital; (i  ) Pointer to structure contained digital*/
/*                                component.                          */
/*..Revision..  : 2001/01/01                                          */
/*--------+---------+---------+---------+---------+---------+---------+--------*/
void Output_1(S_DIGITAL_FILTER *digital)
{
    int     i;

    sprintf(G_reportbuff[0], "------ Direct construction (non round part are not unnormalized) ------¥n¥n" );
```

```
    sprintf(G_reportbuff[1],"                        -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[2],"                 b0 + b1z + b2z + b3z + ... + bNz¥n");
    sprintf(G_reportbuff[3],"  H(z) = Gain ------------------------------------¥n");
    sprintf(G_reportbuff[4],"                        -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[5],"                 1  + a1z + a2z + a3z + ... + bNz¥n¥n");
    sprintf(G_reportbuff[6],"  Gain  = %21.16e¥n", digital->gain1);
    sprintf(G_reportbuff[7],"  N     = %d¥n", digital->dimension);
    W_Writing( CRT_DISK, G_reportbuff, 8, G_fout_report );

    sprintf(G_reportbuff[0], "                                                  b[%2d] = % 21.16e¥n",
                        i, digital->n[i]);
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    for (i = 1; i <= digital->dimension; i++) {
        sprintf(G_reportbuff[0], "  a[%2d] = % 21.16e            b[%2d] = % 21.16e¥n",
                            i, digital->d[i], i, digital->n[i]);
        W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
    }
}

/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : void Output_2(S_DIGITAL_FILTER *digital)                       */
/*  prototype   : PTR_digfil1.h                                                  */
/*  Function    : Result output by formation type 2.                            */
/*  Return Value: none                                                           */
/*  Variable    : *digital; (i  ) Pointer to structure contained digital         */
/*                                component.                                     */
/*。。Revision。。 : 2001/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void Output_2(S_DIGITAL_FILTER *digital)
{
    int     i;


    sprintf(G_reportbuff[0], "------ Direct construction (non round part are not normalized) ------¥n¥n" );
    sprintf(G_reportbuff[1],"                        -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[2],"                 1  + b1z + b2z + b3z + ... + bNz¥n");
    sprintf(G_reportbuff[3],"  H(z) = Gain ------------------------------------¥n");
    sprintf(G_reportbuff[4],"                        -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[5],"                 1  + a1z + a2z + a3z + ... + bNz¥n¥n");
    sprintf(G_reportbuff[6],"  Gain  = %21.16e¥n", digital->gain2);
    sprintf(G_reportbuff[7],"  N     = %d¥n", digital->dimension);
    W_Writing( CRT_DISK, G_reportbuff, 8, G_fout_report );

    for (i = 1; i <= digital->dimension; i++) {
        sprintf(G_reportbuff[0], "  a[%2d] = % 21.16e            b[%2d] = % 21.16e¥n",
                            i, digital->d[i], i, digital->n2[i]);
        W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
    }

}

/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : void Output_3(int dimension, S_TR_FUNCTION *digital)           */
/*  prototype   : PTR_digfil1.h                                                  */
/*  Function    : Result output by formation type 3.                            */
/*  Return Value: none                                                           */
/*  Variable    : dimension:(i  ) Filter dimension.                              */
/*                *digital :(i  ) Pointer to structure contained digital         */
/*                                component.                                     */
/*。。Revision。。 : 2001/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void Output_3(int dimension, S_TR_FUNCTION *Hz2)
{
```

```
    int     i, j, check_dimension, nn;

    check_dimension = Even_Odd_Check(dimension);

    switch (check_dimension) {
        case EVEN :
            nn = dimension / 2;
            break;
        case ODD :
            nn = (dimension + 1) / 2;
            break;
    }

    sprintf(G_reportbuff[0], "---------- Chain construction ----------¥n¥n" );
    sprintf(G_reportbuff[1],"                                  -1      -2¥n");
    sprintf(G_reportbuff[2],"                L    bk,0 + bk,1z + bk,2z ¥n");
    sprintf(G_reportbuff[3],"  H(z) = Gain  !!  ----------------------¥n");
    sprintf(G_reportbuff[4],"               k=1                -1      -2¥n");
    sprintf(G_reportbuff[5],"                        1    + ak,1z + ak,2z¥n¥n");
    sprintf(G_reportbuff[6],"  Gain      = %21.16e¥n", Hz2->gain);
    sprintf(G_reportbuff[7],"  L         = %d¥n", nn);
    W_Writing( CRT_DISK, G_reportbuff, 8, G_fout_report );

    for (i = 1; i <= nn; i++) {
        for (j = 0; j <= 2; j++) {
            sprintf(G_reportbuff[0],"  a[%2d][%2d] = % 21.16e     b[%2d][%2d] = % 21.16e¥n",
                                i, j, Hz2->d[i][j], i, j, Hz2->n[i][j]);
            W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
        }
    }

}

/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : void File_Output(FILE *fout, S_DIGITAL_FILTER *digital)  */
/*  prototype   : PTR_digfil1.h                                           */
/*  Function    : Result output by PO formation.                          */
/*  Return Value: none                                                    */
/*  Variable    : *fout   ; (i  ) Pointer to FILE structure.              */
/*                *digital; (i  ) Pointer to structure contained digital  */
/*                                component.                              */
/*..Revision..  : 2001/01/01                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
void File_Output(FILE *fout, S_DIGITAL_FILTER *digital)
{
    int     i;

    fprintf(fout, "%d¥n", digital->dimension);
    for (i = 1; i <= digital->dimension; i++) {
        fprintf(fout, "% 21.16e¥n", digital->d[i]);
    }

    fprintf(fout, "%d¥n", digital->dimension);
    for (i = 1; i <= digital->dimension; i++) {
        fprintf(fout, "% 21.16e¥n", digital->n2[i]);
    }

    fprintf(fout, "% 21.16e¥n", digital->sampling_frequency);
    fprintf(fout, "% 21.16e¥n", digital->gain2);

}
```

C-5 digfil1_util.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
//////////////////////////////////////////////////////////////////////////////
//                                                                          //
//  Digital Filter Design Program.                                          //
//                                                                          //
//                                                                          //
//////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7--------- */


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+-------- */


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_digfil1.h"
#include "VAR_digfil1.h"
#include "PRT_digfil1.h"


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : void W_Writing(int type,                                    */
/*                               char **string,                               */
/*                               int number                                   */
/*                               FILE *fout)                                  */
/*  Prototype   : PRT_digfil1.h                                               */
/*  Description : Output strings to CRT & DISK.                               */
/*  Return Value: none                                                        */
/*  Variable    : type   ; (i  ) Output type ( CRT, DISK, CRT_DISK ).         */
/*                *string; (i  ) Pointer array of output strings buffer.      */
/*                number;  (i  ) Number of output.                            */
/*                *fout  ; (i  ) File pointer of ourput file.                 */
/*。。Revision。。 : 2001/01/01                                                */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
void  W_Writing(int type, char (*string)[256], int number, FILE *fout)
{
    int     i;

    for ( i = 0; i < number; i++ ) {
        if ( type & CRT ) {
            printf( "%s", string[i] );
        }

        if ( (type & DISK) && ( fout != NULL ) ) {
            fprintf( fout, "%s", string[i] );
        }
    }
}


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : void Now_Time_Set(char *day_buff, char *time_buff)           */
/*  Prototype   : PRT_digfil1.h                                               */
/*  Description : Get now time information.                                   */
/*  Return Value: calendar time at now.                                       */
/*  Variable    : *day_buff;  (i  ) pointer of string for day.                */
/*                *time_buff; (i  ) Pointer of string for time.               */
/*。。Revision。。 : 2001/01/01                                                */
```

```
/* ──────┼─────┼─────┼─────┼─────┼─────┼─────┼───── */
time_t  Now_Time_Set(char *day_buff, char *time_buff)
{
    time_t      ct;
    struct tm   *now_time;

    ct = time( NULL );
    now_time = localtime( &ct );

    sprintf( day_buff,  "%04d/%02d/%02d",
                        now_time->tm_year+1900,
                        now_time->tm_mon+1,
                        now_time->tm_mday);

    sprintf( time_buff, "%02d:%02d:%02d",
                        now_time->tm_hour,
                        now_time->tm_min,
                        now_time->tm_sec );

    return ct;
}
```

C-6： DFN_digfil1.h

```
/* ─────1──────2──────3──────4──────5──────6──────7──────8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Digital Filter Design Program.                                          _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
──────1──────2──────3──────4──────5──────6──────7───── */

/* ══════════════════════════════════════════════════════ */
/* standard prototype include.                            */
/* ══════════════════════════════════════════════════════ */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#include <time.h>


/* ══════════════════════════════════════════════════════ */
/* macro define.                                          */
/* ══════════════════════════════════════════════════════ */

/* Routine check */
#define  SUCCESS        1
#define NOTHING         0
#define FAULT      -1

/* Switch */
#define ON              1
#define OFF             0

/* filter form */
#define  IIR            1
#define  FIR            2

/* filter type */
#define LPF             1
```

```
#define HPF             2
#define BPF             3
#define BRF             4

/* amplitude characteristic */
#define BUTTERWORTH             1
#define CHEBYSHEV         2
#define INVERSE_CHEBYSHEV  3

/* EVEN vs ODD */
#define  EVEN             0
#define ODD                   1

/* Output type */
#define PL3       1
#define NORMAL        2

/* Converged calculation factor */
#define EPS             1.0E-6
#define MAXLOOP         30000

/* Ohters */
#define  INEFFECTIVE   0.0

/* Display switch in W_Writing function */
#define CRT             0x0001
#define DISK       0x0002
#define CRT_DISK   0x0003

/* SQR function macro */
#define SQR(x)        ((x)*(x))

/* ========================================================== */
/* struct define.                                             */
/* ========================================================== */
typedef struct {
    int     form;                   /* filter form */
    int     type;                   /* filter type */
    int     a_character;          /* amplitude characteristic */
    double  sampling_frequency;     /* sampling frequency of data */
    double  cutoff_frequency_1;     /* 1st cutoff frequency */
    double  cutoff_frequency_2;     /* 2nd cutoff frequency */
    double  stopping_frequency_1; /* 1st stopping frequency */
    double  stopping_frequency_2; /* 2nd stooping frequency */
    double  pass_ripple;          /* pass ripple */
    double  rejection_power;      /* rejection power */
} S_FILTER_DESIGN;

typedef struct {
    double  cutoff_1;               /* the 1st cutoff frequency after prewarping */
    double  stopping_1;             /* the 1st stopping frequency after prewarping */
    double  cutoff_2;               /* the 2nd cutoff frequency after prewarping */
    double  stopping_2;             /* the 2nd stopping frequency after prewarping */
} S_PREWARP;

typedef struct  {
    double  real[31];             /* real part of complex */
    double  imagi[31];              /* imaginal part of complex */
} S_COMPLEX;

typedef struct  {
    int     dimension;              /* digital filter dimension */
    double  sampling_frequency;     /* sampling frequency */
    double  d[31];                  /* round part of digital filter */
```

```
    double   n[31];                 /* non round part of digital filter */
    double   n2[31];                /* non round part of digital filter */
    double   gain1;                 /* gain (unnormalization) */
    double   gain2;                 /* gain (normalization) */
} S_DIGITAL_FILTER;

typedef struct  {
    double   gain;                  /* gain */
    double   d[16][3];              /* round part of digital filter */
    double   n[16][3];              /* non round part of digital filter */
} S_TR_FUNCTION;

/* ================================================================ */
/* enum define.                                                     */
/* ================================================================ */
typedef enum _tlog_login_pb_code {
    TLOG_LOGIN_EXEC=0,
    TLOG_LOGIN_MESSAGE,
    TLOG_LOGIN_QUIT
} TLOG_login_pb_code;
```

C-7 : PRT_digfil1.h

```
/* ------1--------2--------3--------4--------5--------6--------7--------8
/                                                                       /
/  Digital Filter Design Program.                                       /
/                                                                       /
/                                                                       /
/                                                                       /
--------1--------2--------3--------4--------5--------6--------7------- */


/* ================================================================ */
/* prototype define.                                                */
/* ================================================================ */
/* ------+--------+--------+--------+--------+--------+--------+---- */
/* digfil1_main.c                                                   */
/* ------+--------+--------+--------+--------+--------+--------+---- */
extern void     Print_Usage(void);
extern void     Title(void);
extern int      AutoMain(char **);


/* ------+--------+--------+--------+--------+--------+--------+---- */
/* digfil1_read.c                                                   */
/* ------+--------+--------+--------+--------+--------+--------+---- */
extern int      ReadFileCall(char *, S_FILTER_DESIGN *);
extern int      Std_Reading(FILE *, char *, int);
extern int      Design_Aveilability_Check(S_FILTER_DESIGN *);

/* ------+--------+--------+--------+--------+--------+--------+---- */
/* digfil_IIR.c                                                     */
/* ------+--------+--------+--------+--------+--------+--------+---- */
extern int      IIT_Transfer_Function(S_FILTER_DESIGN *,
                                      S_DIGITAL_FILTER *,
                                      S_TR_FUNCTION *,
                                      S_TR_FUNCTION *);
extern void     Prewarping(S_FILTER_DESIGN *, S_PREWARP *);
extern int      Butterworth_Dimension(S_FILTER_DESIGN *, S_PREWARP *);
extern int      Butterworth_L_H_Dim(double, double, double);
extern void     Butterworth_Normal_LPF_Pole(int, double, S_COMPLEX *);
```

```
extern void     Butterworth_LPF(int, double, double, S_COMPLEX *, S_TR_FUNCTION *);
extern void     Butterworth_HPF(int, double, double, S_COMPLEX *, S_TR_FUNCTION *);
extern void     Bilinear_Transformation(int, double, S_TR_FUNCTION *, S_TR_FUNCTION *);
extern void     Direct_Method(S_DIGITAL_FILTER *, S_TR_FUNCTION *);
extern void     Expansion(int, double [][3], double []);
extern int      Even_Odd_Check(int);


/* ------+---------+---------+---------+---------+---------+---------+------ */
/* digfil1_output.c                                                          */
/* ------+---------+---------+---------+---------+---------+---------+------ */
extern void     Output_Call(S_DIGITAL_FILTER *, S_TR_FUNCTION *);
extern void     Output_1(S_DIGITAL_FILTER *);
extern void     Output_2(S_DIGITAL_FILTER *);
extern void     Output_3(int, S_TR_FUNCTION *);
extern void     File_Output(FILE *, S_DIGITAL_FILTER *);


/* ------+---------+---------+---------+---------+---------+---------+------ */
/* digfil1_util.c                                                            */
/* ------+---------+---------+---------+---------+---------+---------+------ */
extern void  W_Writing(int, char (*)[256] ,int, FILE *);
extern time_tNow_Time_Set(char *, char *);
```

C-8: VAR_digfil1.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
//////////////////////////////////////////////////////////////////////////////
//                                                                          //
//  Digital Filter Design Program.                                          //
//                                                                          //
//                                                                          //
//////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------ */

/* ========================================================================= */
/* extern define.                                                            */
/* ========================================================================= */
#ifdef DIGFIL1_MAIN
    #define EXT
#else
    #define EXT     extern
#endif

/* ========================================================================= */
/* varriable define.                                                         */
/* ========================================================================= */
/* +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* global.                                                                   */
/* +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */

/* output file name and type */
EXT char            G_out_filename[256];

/* string array of reading parameter item */
#ifdef DIGFIL1_MAIN
char                *G_parameter_item[] = {
                                    "FILTER_FORM:::::::::::::::::::::",
                                    "FILTER_TYPE:::::::::::::::::::::",
                                    "AMPLITUDE_CHARACTERISTIC:::::::::",
                                    "SAMPLING_FREQUENCY:::::::::::::::",
                                    "1ST_CUTOFF_FREQUENCY:::::::::::::",
```

```
                                        "1ST_STOPPING_FREQUENCY:::::::::::",
                                        "2ND_CUTOFF_FREQUENCY:::::::::::::",
                                        "2ND_STOPPING_FREQUENCY:::::::::::",
                                        "PASS_RIPPLE:::::::::::::::::::::::",
                                        "REJECTION_POWER:::::::::::::::::::",
                                        "DIGFIL2_FORM_OUTPUT_FILENAME::::" };
#else
extern char             *G_parameter_item[];
#endif

/* message buff array for W_Writing[in armax1_util.c] function */
EXT char            G_reportbuff[10][256];

/* file pointer for W_Writing[in armax1_util.c] function */
EXT FILE            *G_fout_report;

/* ------+---------+---------+---------+---------+---------+---------+------ */
#undef EXT
/* ------+---------+---------+---------+---------+---------+---------+------ */
```

D Digital Filtering Tool プログラムリスト

D-1: digfil2_main.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed    _/
_/  by DIGFIL1.                                                               _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* MAIN FUNCTION SETTING.                                                      */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#define  DIGFIL2_MAIN

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_digfil2.h"
#include "VAR_digfil2.h"
#include "PRT_digfil2.h"

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* static variable defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+------- */
S_PROCESSING     processing;
S_FILTER         filter;


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : void main(int argc, char **argv)                             */
/*  prototype   : none                                                         */
/*  Function    : Main function for fft program.                               */
/*  Return Value: none                                                         */
/*  Variable    : argc  ; (i  ) Number of variable on the command line.        */
/*                **argv; (i  ) Pointer of strings on the command line.        */
/*。。Revision。。 : 2001/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  main(int argc, char **argv)
{

    int      check;


    if ( argc != 3 ) {
        Print_Usage();
        exit(0);
    }

    Title();

    if ( (G_fout_report = fopen(argv[2], "wt")) == NULL ) {
        printf( "Fatal Error: Cannot open the report file [%s]!! ¥n", argv[2] );
```

```
        exit(0);
    }

    check = AutoMain(argv);

    if ( check == SUCCESS ) {
        sprintf(G_reportbuff[0],
        "========================== NORMAL TERMINATION ==========================¥n¥n");
    } else {
        sprintf(G_reportbuff[0],
        "========================= ABNORMAL TERMINATION =========================¥n¥n");
    }

    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    fclose(G_fout_report);
}


/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : void Print_Usage(void)                                      */
/*  prototype   : PRT_digfil2.h                                               */
/*  Function    : Display Usage                                               */
/*  Return Value: none                                                        */
/*  Variable    : none                                                        */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------ */
void  Print_Usage(void) {

    printf( "Usage:¥n" );
    printf( "  digfil2.out  control_filename  report_filename¥n" );
    printf( "              control_filname: path of job file.¥n" );
    printf( "              report_filnema : path of report file.¥n¥n" );

}

/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : void Title(void)                                            */
/*  prototype   : PRT_fft.h                                                   */
/*  Function    : Display Title                                               */
/*  Return Value: none                                                        */
/*  Variable    : none                                                        */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------ */
void  Title(void) {

    printf("¥n¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("_/                                                               _/¥n");
    printf("_/ DIGFIL2: DIGITAL FILTERING TOOL TO BINALY DATA FILE WITH THE  _/¥n");
    printf("_/          PARAMETER OUTPUTED BY DIGFIL1                        _/¥n");
    printf("_/                                                               _/¥n");
    printf("_/     Version 1.01 : 2001/01/01 : transplantation from PC version.  _/¥n");
    printf("_/                                                               _/¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("¥n¥n");

}


/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : int AutoMain(void)                                          */
/*  prototype   : PRT_digfil2.h                                               */
/*  Function    : Execution Control                                           */
```

```
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/*  Variable    : **argv; (i  ) Pointer of strings on the command line.       */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  AutoMain(char **argv)
{
    FILE        *fin_control;
    char        setup_filename[256];
    int         count, check;
    char        answer;


    /* Initialize */
    count = 0;

    /* Job control file open */
    if ( (fin_control = fopen(argv[1], "rt")) == NULL) {
        printf( "Fatal Error: Cannot open the control file [%s]!! ¥n", argv[1] )
;
        return FAULT;
    }


    /* Roop process */
    while ( !feof(fin_control) ) {
         count++;
         fscanf(fin_control, "%s", setup_filename);
         if ( feof(fin_control) ) {
             break;
         }

         sprintf( G_reportbuff[0], "¥n¥n" );
         sprintf( G_reportbuff[1],
             "-- DIGFIL2 Execution ----------------------------------------------------------------------¥n");
        sprintf( G_reportbuff[2], "JOB No. %d¥n", count );
        sprintf( G_reportbuff[3], "Setup File Name = %s¥n¥n", setup_filename );
        W_Writing( CRT_DISK, G_reportbuff, 4, G_fout_report );

        if ( ReadFileCall(setup_filename, &processing, &filter) == FAULT) {
            continue;
        }
        check = FilteringCall(&processing, &filter);
    }
    fclose(fin_control);

    return check;

}
```

D-2: digfil2_read.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed    _/
_/  by DIGFIL1.                                                               _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
--------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------ */
/* standard include defines.                                                  */
```

```
/* ------+---------+---------+---------+---------+---------+---------+------ */


/* ------+---------+---------+---------+---------+---------+---------+------ */
/* private include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+------ */
#include "DFN_digfil2.h"
#include "VAR_digfil2.h"
#include "PRT_digfil2.h"


/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : int ReadFileCall(char *filename,                             */
/*                                 S_PROCESSING *processing,                   */
/*                                 S_FILTER *filter)                           */
/*  Prototype   : PRT_digfil2.h                                                */
/*  Discription : Reading job file and setting parameter.                      */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]   */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h]   */
/*  Variable    : *filename  ; (i  ) pointer to filenmae buffer.               */
/*                *processing; (  o) pointer to structure contained processing*/
/*                                   components.                              */
/*                *filter    ; (  o) pointer to structure contained filter    */
/*                                   design.                                  */
/*。。Revision。。 : 2000/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  ReadFileCall(char *filename, S_PROCESSING *processing, S_FILTER *filter)
{
    FILE     *fin_setupfile, *fin_filter, *fin_data, *fout_test;
    char     input_buff[256];
    int      i, j, check;
    int          usual_number;


    /* Setup file open */
    if ( (fin_setupfile = fopen(filename, "rt")) == NULL ) {
        sprintf(G_reportbuff[0], "Cannot open setup file [ %s ].¥n", filename);
        sprintf(G_reportbuff[1], "Abort this job.¥n", filename);
        W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);
        return FAULT;
    }


    /* Initialize */
    memset(processing, 0x00, sizeof(S_PROCESSING));
    memset(filter,     0x00, sizeof(S_FILTER));


    /* Reading parameter in setup file */
    for ( i = 0; i < 7; i++ ) {
        if ( Std_Reading(fin_setupfile, input_buff, i) == FAULT ) {
            sprintf(G_reportbuff[0], "Error: Format of setup file is unsuitable.¥n");
            goto ERROR_RECOVERY;
        }
        switch ( i ) {

            case 0: /* Filter filename */
                if ( (fin_filter = fopen(input_buff, "rt")) == NULL ) {
                    sprintf(G_reportbuff[0], "Cannot open the filter file.¥n" );
                    goto ERROR_RECOVERY;
                 } else {
                    strcpy(processing->filter_name, input_buff);
                    check = Filter_File_Reading(fin_filter, filter);
                    fclose(fin_filter);
```

```
            if ( check == FAULT ) {
                sprintf(G_reportbuff[0],
                    "Filter file [ %s ] is not aveirable file.¥n", input_buff);
                goto ERROR_RECOVERY;
            }
        }
        break;

    case 1: /* Reading record length of binary time series */
        processing->input_byte = atoi(input_buff);
        if ( processing->input_byte != 2 && processing->input_byte != 8 ) {
            sprintf(G_reportbuff[0],
                        "Illegal value as record length of the input time series data.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 2: /* Input filename */
        if ( (fin_data = fopen(input_buff, "rb")) == NULL ) {
            sprintf(G_reportbuff[0],
                "Cannot open the time series data file.¥n" );
            goto ERROR_RECOVERY;
         } else {
            strcpy(processing->input_name, input_buff);
            processing->total_number = Binary_File_Check(fin_data, processing->input_byte);
            fclose(fin_data);
            if ( processing->total_number == -1 ) {
                sprintf(G_reportbuff[0],
                    "Probably, [ %s ] is not binary data file with %d byte¥n",
                    input_buff, processing->input_byte);
                goto ERROR_RECOVERY;
            }
        }
        usual_number = processing->total_number;
        break;

    case 3: /* Skip number */
        processing->skip_number = atoi(input_buff);
        if ( processing->skip_number < 0 || processing->skip_number > usual_number ) {
            sprintf(G_reportbuff[0], "Illegal value as skip number.¥n");
            goto ERROR_RECOVERY;
        }
        usual_number -= processing->skip_number;
        break;

    case 4: /* Processing number */
        if ( strcmp(input_buff, "MAX") == 0 ) {
            processing->processing_number = usual_number;
        } else {
            processing->processing_number = atoi(input_buff);
            if ( processing->processing_number <= 0  ||
                            processing->processing_number > usual_number) {
                sprintf(G_reportbuff[0], "Illegal value as processing number.¥n");
                goto ERROR_RECOVERY;
            }
        }
        break;

    case 5: /* Output record length of binary time series */
        processing->output_byte = atoi(input_buff);
        if ( processing->output_byte != 2 && processing->output_byte != 8 ) {
            sprintf(G_reportbuff[0],
                        "Illegal value as record length of the output time series data.¥n");
            goto ERROR_RECOVERY;
```

```
                }
                break;

            case 6: /* output filename  */
                if ( (fout_test = fopen(input_buff, "wb")) == NULL ) {
                    sprintf(G_reportbuff[0], "Unsuitable Output File Name.¥n");
                    goto ERROR_RECOVERY;
                }
                fclose(fout_test);
                strcpy(processing->output_name, input_buff);
                break;

        }
    }

    sprintf(G_reportbuff[0], "¥nProcessing Number = %d¥n", processing->processing_number);
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    /* Output filter characteristic */
    Output_2(filter);


    fclose(fin_setupfile);
    return SUCCESS;


ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    fclose(fin_setupfile);
    return FAULT;
}

/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : long Binary_File_Check(FILE *fin, int byte)                  */
/*  Prototype   : PRT_digfil2.h                                                */
/*  Discription : Check of binary file.                                        */
/*  Return Value: normal termination   -> number of data                       */
/*                abnormal termination -> -1                                   */
/*  Variable    : *fin; (i  ) file pointer of data file.                       */
/*                byte; (i  ) byte of data.                                    */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------- */
long  Binary_File_Check(FILE *fin, int byte)
{
    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);
    if ( d1_dummy != d2_dummy ) {
        return -1;
    }

    data_number = file_size / byte;

    return data_number;
}

/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : int Filter_File_Reading(FILE *fin, S_FILTER *filter)         */
```

```
/*  Prototype   : PRT_digfil2.h                                              */
/*  Discription : Reading filter parameter and check the components.         */
/*                averaging.                                                 */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h] */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h] */
/*  Variable    : *fin   ; (i  ) Pointer to FILE structure.                  */
/*                *filter; (  o) Pointer to the structure contained filter   */
/*                               components.                                 */
/*。。Revision。。 : 2001/01/01                                               */
/* ------+--------+--------+--------+--------+--------+--------+---------- */
int  Filter_File_Reading(FILE *fin, S_FILTER *filter)
{

    char input_buff[256];
    int      i;


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->denominator_order = atoi(input_buff);
    if (filter->denominator_order < 0 || filter->denominator_order > 100 ||
                                *input_buff < '0' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as rounde part order.¥n");
        goto ERROR_RECOVERY;
    }

    for (i = 1; i <= filter->denominator_order; i++) {
        if ( Std_Reading2(fin, input_buff) == FAULT ) {
            goto ERROR_RECOVERY;
        }
        filter->denominator[i] = atof(input_buff);
        if (*input_buff < '-' || *input_buff > '9') {
            sprintf(G_reportbuff[0], "Illegal value as rounde part parameter.¥n");
            goto ERROR_RECOVERY;
        }
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->numerator_order = atoi(input_buff);
    if (filter->numerator_order < 0 || filter->numerator_order > 100 ||
                                *input_buff < '0' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as non rounde part order.¥n");
        goto ERROR_RECOVERY;
    }

    for (i = 1; i <= filter->numerator_order; i++) {
        if ( Std_Reading2(fin, input_buff) == FAULT ) {
            goto ERROR_RECOVERY;
        }
        filter->numerator[i] = atof(input_buff);
        if (*input_buff < '-' || *input_buff > '9') {
            sprintf(G_reportbuff[0], "Illegal value as non rounde part parameter.¥n");
            goto ERROR_RECOVERY;
        }
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
```

```
    filter->sampling_frequency = atof(input_buff);
    if (filter->sampling_frequency <= 0 || filter->sampling_frequency > 1.7E308 ||
                            *input_buff < '-' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as sampling frequency.¥n");
        goto ERROR_RECOVERY;
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->gain = atof(input_buff);
    if (filter->gain <= 0 || filter->gain > 1.7E308 ||
                                        *input_buff < '-' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as filter gain.¥n");
        goto ERROR_RECOVERY;
    }

    return SUCCESS;

ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    return FAULT;
}



/* ------+--------+--------+--------+--------+--------+--------+------- */
/* Format      : int Std_Reading(FILE *fin, char *return_buff, int counter)  */
/* Prototype   : PRT_digfil2.h                                               */
/* Discription : Standard reading function to setup file.                    */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*               abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/* Variable    : *fin        ; (i  ) file pointer to setup file.             */
/*               *return_buff; (  o) string buffer pointer for reading para- */
/*                                   meter.                                  */
/*               counter     ; (i  ) logical ID of parameter.                */
/*。。Revision。。 : 2001/01/01                                               */
/* ------+--------+--------+--------+--------+--------+--------+------- */
int  Std_Reading(FILE *fin, char *ptr1, int counter)
{

    char tmp_buff1[256];
    int      check;


    check = fscanf(fin, "%s %s", tmp_buff1, ptr1);
    sprintf( G_reportbuff[0], "%s %s¥n", tmp_buff1, ptr1 );
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    if ( (check != 2 ) || (strcmp( tmp_buff1, G_parameter_item[counter]) != 0) ) {
        return FAULT;
    }

    return SUCCESS;
}

/* ------+--------+--------+--------+--------+--------+--------+------- */
/* Format      : int Std_Reading2(FILE *fin, char *return_buff)              */
/* Prototype   : PRT_digfil2.h                                               */
/* Discription : Standard reading function to filter file.                   */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*               abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/* Variable    : *fin        ; (i  ) file pointer to filter file.            */
```

```
/*                *return_buff; (  o) string buffer pointer for reading para- */
/*                                    meter.                                  */
/*。。Revision。。 : 2001/01/01                                                */
/* ------+---------+---------+---------+---------+---------+---------+------- */
int  Std_Reading2(FILE *fin, char *ptr1)
{

    char tmp_buff1[256];
    int      check;


    check = fscanf(fin, "%s", ptr1);

    if ( check != 1 ) {
        sprintf(G_reportbuff[0], "Error: Format of filter file is unsuitable.¥n");
        return FAULT;
    }

    return SUCCESS;
}
```

D-3: digfil2_output.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed  _/
_/  by DIGFIL1.                                                             _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_digfil2.h"
#include "VAR_digfil2.h"
#include "PRT_digfil2.h"


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* static variable defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */


* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format       : void Output_2(S_DIGITAL_FILTER *digital)                   */
/*  prototype    : PTR_digfil2.h                                              */
/*  Function     : Result output by formation type 2.                         */
/*  Return Value: none                                                        */
/*  Variable     : *filter; (i  ) Pointer to structure contained digital      */
/*                                component.                                  */
/*。。Revision。。 : 2001/01/01                                                */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void Output_2(S_FILTER *filter)
{
    int      i;
    int      max_dimension;
```

```
    max_dimension = max(filter->denominator_order, filter->numerator_order);

    sprintf(G_reportbuff[0],"¥n");
    sprintf(G_reportbuff[1],"                          -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[2],"                  1  + b1z + b2z + b3z + ... + bNz¥n");
    sprintf(G_reportbuff[3],"  H(z) = Gain ------------------------------------¥n");
    sprintf(G_reportbuff[4],"                          -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[5],"                  1  + a1z + a2z + a3z + ... + bNz¥n¥n");
    sprintf(G_reportbuff[6],"  Gain  = %21.16e¥n", filter->gain);
    sprintf(G_reportbuff[7],"  N     = %d¥n", max_dimension);
    sprintf(G_reportbuff[8],"  Fs    = %21.16e¥n", filter->sampling_frequency );
    W_Writing( CRT_DISK, G_reportbuff, 9, G_fout_report );

    for (i = 1; i <= max_dimension; i++) {
        sprintf(G_reportbuff[0], "  a[%2d] = % 21.16e             b[%2d] = % 21.16e¥n",
                                 i, filter->denominator[i], i, filter->numerator[i]);
        W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
    }

}
```

D-4: digfil2_make.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed    _/
_/  by DIGFIL1.                                                               _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_digfil2.h"
#include "VAR_digfil2.h"
#include "PRT_digfil2.h"


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : int  FilteringCall(S_PROCESSING *processing,                */
/*                                   S_FILTER *filter)                        */
/*  Prototype   : PRT_digfil2.h                                               */
/*  Discription : Execution process of filtering.                             */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/*  Variable    : *processing; (i  ) pointer to structure contained processing*/
/*                                   components.                              */
/*                *filter    ; (i  ) pointer to structure contained filter    */
/*                                   design.                                  */
/*  Revision    : 2001/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
int FilteringCall(S_PROCESSING *processing, S_FILTER *filter)
```

```
{
    short           s_value, s_output;
    int             i;
    time_t          start_time, end_time;
    char        day_buff[32], time_buff[32];
    double          previous_input[100], previous_output[100];
    double          dummy_d1, current_data, d_output;
    double          require_time;
    FILE        *fin, *fout;


    /* Initialization */
    for (i = 1; i <= 100; i++) {
        previous_input[i] = previous_output[i] = 0.0;
    }

    /* Start time setting */
    start_time = Now_Time_Set(day_buff, time_buff);
    sprintf(G_reportbuff[0], "¥nStart of Calculation = %s %s¥n", day_buff, time_buff);
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    /* Input file open and binaly data skipping */
    fin = fopen(processing->input_name, "rb");
    fseek(fin,. processing->skip_number, SEEK_SET);

    /* Output file open */
    fout = fopen(processing->output_name, "wb");

    /* Main execution */
    switch ( processing->input_byte ) {
        case 2 :
            switch ( processing->output_byte) {
                case 2 :
                    for ( i = 1; i <= processing->processing_number; i++) {
                        fread(&s_value, 2, 1, fin);
                        current_data = (double)s_value;
                        d_output = System_Filter(filter,
                                                 previous_input,
                                                 previous_output,
                                                 current_data);
                        s_output = (int)d_output;
                        fwrite(&s_output, 2, 1, fout);
                    }
                    break;
                case 8 :
                    for ( i = 1; i <= processing->processing_number; i++) {
                        fread(&s_value, 2, 1, fin);
                        current_data = (double)s_value;
                        d_output = System_Filter(filter,
                                                 previous_input,
                                                 previous_output,
                                                 current_data);
                        fwrite(&d_output, 8, 1, fout);
                    }
                    break;
            }
            break;
        case 8 :
            for (i = 1; i <= processing->processing_number; i++) {
                fread(&current_data, 8, 1, fin);
                d_output = System_Filter(filter,
                                         previous_input,
```

```
                                        previous_output,
                                        current_data);
                fwrite(&d_output, 8, 1, fout);
            }
            break;
    }

    /* End time setting */
    end_time = Now_Time_Set(day_buff, time_buff);
    require_time = difftime(end_time, start_time);
    sprintf(G_reportbuff[0], "End of Execution     = %s %s¥n", day_buff, time_buff);
    sprintf(G_reportbuff[1], "Requirement Time     = %f [sec]¥n", require_time);
    W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);


    fclose(fin);
    fclose(fout);

    return SUCCESS;

}


/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : double System_Filter(S_FILTER *filter,                 */
/*                                     double *previous_input,           */
/*                                     double *previous_output,          */
/*                                     double current_data)              */
/*  Prototype   : PRT_digfil2.h                                          */
/*  Discription : Execution process of filtering.(sub routine)           */
/*  Return Value: calcurated value.                                      */
/*  Variable    : *filter         ; (  o) pointer to structure contained */
/*                                        filter design.                 */
/*                *previous_input ; (i/o) pointer to array of previsou input */
/*                                        data.                          */
/*                *previous_output; (i/o) pointer to array of previsou output */
/*                                        data.                          */
/*                current_data    ; (i  ) current reading data.          */
/*。。Revision。。 : 2001/01/01                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
double System_Filter(S_FILTER *filter,
                     double *previous_input,
                     double *previous_output,
                     double current_data)
{
    int     i;
    double  y;

    y = current_data;

    for (i = 1; i <= filter->numerator_order; i++)
        y += filter->numerator[i] * previous_input[i];

    y *= filter->gain;

    for (i = 1; i <= filter->denominator_order; i++)
        y -= filter->denominator[i] * previous_output[i];

    for (i = filter->denominator_order; i >= 2; i--)
        previous_output[i] = previous_output[i-1];

    for (i = filter->numerator_order; i >= 2; i--)
        previous_input[i] = previous_input[i-1];
```

```
    previous_output[1] = y;
    previous_input[1]  = current_data;

    return y;
}

/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : void Monitor(void)                                           */
/*  Prototype   : PRT_digfil2.h                                                */
/*  Discription : Monitoring.                                                  */
/*  Return Value: none.                                                        */
/*  Variable    : none.                                                        */
/*。。Revision。。 : 2001/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  Monitor(void)
{
    sprintf(G_reportbuff[0], "o");
    W_Writing(CRT, G_reportbuff, 1, G_fout_report);
    sleep(1);
}
```

D-5: digfil2_util.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed    _/
_/  by DIGFIL1.                                                               _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_digfil2.h"
#include "VAR_digfil2.h"
#include "PRT_digfil2.h"


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : void W_Writing(int type,                                    */
/*                               char **string,                               */
/*                               int number                                   */
/*                               FILE *fout)                                  */
/*  Prototype   : PRT_digfil2.h                                                */
/*  Description : Output strings to CRT & DISK.                                */
/*  Return Value: none                                                         */
/*  Variable    : type   ; (i  ) Output type ( CRT, DISK, CRT_DISK ).          */
/*                *string; (i  ) Pointer array of output strings buffer.       */
/*                number;  (i  ) Number of output.                             */
/*                *fout  ; (i  ) File pointer of ourput file.                  */
/*。。Revision。。 : 2001/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  W_Writing(int type, char (*string)[256], int number, FILE *fout)
{
```

```
    int     i;

    for ( i = 0; i < number; i++ ) {
        if ( type & CRT ) {
            printf( "%s", string[i] );
        }

        if ( (type & DISK) && ( fout != NULL ) ) {
            fprintf( fout, "%s", string[i] );
        }
    }
}
```

```
/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : void Now_Time_Set(char *day_buff, char *time_buff)          */
/*  Prototype   : PRT_digfil2.h                                              */
/*  Description : Get now time information.                                  */
/*  Return Value: calendar time at now.                                      */
/*  Variable    : *day_buff;  (i  ) pointer of string for day.               */
/*                *time_buff; (i  ) Pointer of string for time.              */
/*。。Revision。。 : 2001/01/01                                               */
/* ------+---------+---------+---------+---------+---------+---------+------ */
time_t  Now_Time_Set(char *day_buff, char *time_buff)
{
    time_t      ct;
    struct tm   *now_time;

    ct = time( NULL );
    now_time = localtime( &ct );

    sprintf( day_buff,  "%04d/%02d/%02d",
                        now_time->tm_year+1900,
                        now_time->tm_mon+1,
                        now_time->tm_mday);

    sprintf( time_buff, "%02d:%02d:%02d",
                        now_time->tm_hour,
                        now_time->tm_min,
                        now_time->tm_sec );

    return ct;
}
```

D-6: DFN_digfil2.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed  _/
_/  by DIGFIL1.                                                            _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------ */

/* ========================================================================== */
/* prototype define.                                                          */
/* ========================================================================== */
/* ------+---------+---------+---------+---------+---------+---------+------ */
/* digfil2_main.c                                                             */
/* ------+---------+---------+---------+---------+---------+---------+------ */
```

```
extern void       Print_Usage(void);
extern void       Title(void);
extern int        AutoMain(char **);

/* ------+--------+--------+--------+--------+--------+--------+------- */
/* digfil2_read.c                                                       */
/* ------+--------+--------+--------+--------+--------+--------+------- */
extern int        ReadFileCall(char *, S_PROCESSING *, S_FILTER *);
extern long       Binary_File_Check(FILE *, int);
extern int        Filter_File_Reading(FILE *, S_FILTER *);
extern int        Std_Reading(FILE *, char *, int);
extern int        Std_Reading2(FILE *, char *);

/* ------+--------+--------+--------+--------+--------+--------+------- */
/* digfil2_make.c                                                       */
/* ------+--------+--------+--------+--------+--------+--------+------- */
extern int        FilteringCall(S_PROCESSING *, S_FILTER *);
extern doubleSystem_Filter(S_FILTER *, double *, double *, double);
extern void       Monitor(void);

/* ------+--------+--------+--------+--------+--------+--------+------- */
/* digfil2_util.c                                                       */
/* ------+--------+--------+--------+--------+--------+--------+------- */
extern void  W_Writing(int, char (*)[256] ,int, FILE *);
extern time_tNow_Time_Set(char *, char *);
```

D-7: VAR_digfil2.h

```
/* ------1---------2---------3---------4---------5---------6---------7--------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Digital Filtering Tool to Binaly Data File with the Parameter outputed  _/
_/  by DIGFIL1.                                                            _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
--------1---------2---------3---------4---------5---------6---------7------- */

/* ==================================================================== */
/* extern define.                                                       */
/* ==================================================================== */
#ifdef DIGFIL2_MAIN
    #define EXT
#else
    #define EXT       extern
#endif

/* ==================================================================== */
/* varriable define.                                                    */
/* ==================================================================== */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* global.                                                              */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */

/* output file name and type */
EXT char              G_out_filename[256];

/* string array of reading parameter item */
#ifdef DIGFIL2_MAIN
```

```
char                    *G_parameter_item[] = {
                                        "FILTER_FILE_BY_DIGFIL1:::::::::::",
                                        "INPUT_RECORD_RENGTH(2/8)::::::::::",
                                        "BINARY_DATA_FILE:::::::::::::::::",
                                        "SKIP_DATA:::::::::::::::::::::::",
                                        "NUMBER_OF_DATA:::::::::::::::::::",
                                        "OUTPUT_RECORD_LENGTH(2/8)::::::::",
                                        "OUTPUT_DATA_FILE:::::::::::::::::" };
#else
extern char                 *G_parameter_item[];
#endif

/* message buff array for W_Writing[in armax1_util.c] function */
EXT char                G_reportbuff[10][256];

/* file pointer for W_Writing[in armax1_util.c] function */
EXT FILE                *G_fout_report;

/* ------+---------+---------+---------+---------+---------+---------+------ */
#undef EXT
/* ------+---------+---------+---------+---------+---------+---------+------ */
```

E PO3 プログラムリスト

E-1 po3_main.c

```
* ------1---------2---------3---------4---------5---------6---------7--------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  Frequency Characteristic Calculation Proguram P03.                      _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7-------- */

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* MAIN FUNCTION SETTING.                                                      */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#define  P03_MAIN

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* standard include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+-------- */



/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_po3.h"
#include "VAR_po3.h"
#include "PRT_po3.h"

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* static variable defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
S_PROCESSING    processing;
S_FILTER        filter;
S_OUTPUT        output;

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : void main(int argc, char **argv)                             */
/*  prototype   : none                                                         */
/*  Function    : Main function for fft program.                               */
/*  Return Value: none                                                         */
```

```
/* Variable    : argc  ; (i  ) Number of variable on the command line.       */
/*               **argv; (i  ) Pointer of strings on the command line.       */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+--------+--------+--------+--------+--------+--------+-------- */
void  main(int argc, char **argv)
{

    int      check;


    if ( argc != 3 ) {
        Print_Usage();
        exit(0);
    }

    Title();

    if ( (G_fout_report = fopen(argv[2], "wt")) == NULL ) {
        printf( "Fatal Error: Cannot open the report file [%s]!! ¥n", argv[2] );
        exit(0);
    }

    check = AutoMain(argv);

    if ( check == SUCCESS ) {
        sprintf(G_reportbuff[0],
        "========================== NORMAL TERMINATION ==========================¥n¥n");
    } else {
        sprintf(G_reportbuff[0],
        "========================= ABNORMAL TERMINATION =========================¥n¥n");
    }

    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);

    fclose(G_fout_report);
}



/* ------+--------+--------+--------+--------+--------+--------+------- */
```

```
/*  Format      : void Print_Usage(void)                                    */
/*  prototype   : PRT_po3.h                                                 */
/*  Function    : Display Usage                                             */
/*  Return Value: none                                                      */
/*  Variable    : none                                                      */
/*。。Revision。。 : 2001/01/01                                              */
/* ------+---------+---------+---------+---------+---------+------------- */
void  Print_Usage(void) {

    printf( "Usage:¥n" );
    printf( "  po3.out  control_filename  report_filename¥n" );
    printf( "             control_filname: path of job file.¥n" );
    printf( "             report_filnema : path of report file.¥n¥n" );

}

/* ------+---------+---------+---------+---------+---------+------------- */
/*  Format      : void Title(void)                                          */
/*  prototype   : PRT_po3.h                                                 */
/*  Function    : Display Title                                             */
/*  Return Value: none                                                      */
/*  Variable    : none                                                      */
/*。。Revision。。 : 2001/01/01                                              */
/* ------+---------+---------+---------+---------+---------+------------- */
void  Title(void) {

    printf("¥n¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("_/                                                                                _/¥n");
    printf("_/ P03: FREQUENCY CHRACTERISTIC CALCULATION PROGRAM WITH THE PARAMETER            _/¥n");
    printf("_/      OUTPUTED BY DIGFIL1                                                       _/¥n");
    printf("_/                                                                                _/¥n");
    printf("_/     Version 1.01 : 2001/01/01 : transplantation from PC version.               _/¥n");
    printf("_/                                                                                _/¥n");
    printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥n");
    printf("¥n¥n");

}
```

```
/* ------+--------+--------+--------+--------+--------+--------+------ */
/*  Format      : int AutoMain(void)                                     */
/*  prototype   : PRT_po3.h                                              */
/*  Function    : Execution Control                                      */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/*  Variable    : **argv; (i  ) Pointer of strings on the command line.  */
/*..Revision..  : 2001/01/01                                             */
/* ------+--------+--------+--------+--------+--------+--------+------ */
int  AutoMain(char **argv)
{
    FILE        *fin_control, *fout_result;
    char        setup_filename[256];
    int         count, check;


    /* Initialize */
    count = 0;

    /* Job control file open */
    if ( (fin_control = fopen(argv[1], "rt")) == NULL) [
        printf( "Fatal Error: Cannot open the control file [%s]!! ¥n", argv[1] )
;
        return FAULT;
    ]


    /* Roop process */
    while ( !feof(fin_control) ) [
        count++;
        fscanf(fin_control, "%s", setup_filename);
        if ( feof(fin_control) ) [
            break;
        ]

        sprintf( G_reportbuff[0], "¥n¥n" );
        sprintf( G_reportbuff[1],
            "-- P03 Execution ----------------------------------------------------------------¥n");
```

```
        sprintf( G_reportbuff[2], "JOB No. %d¥n", count );
        sprintf( G_reportbuff[3], "Setup File Name = %s¥n¥n", setup_filename );
        W_Writing( CRT_DISK, G_reportbuff, 4, G_fout_report );

        if ( ReadFileCall(setup_filename, &processing, &filter) == FAULT) {
            continue;
        }

        /* memory assignment */
        filter.nyquist_frequency = filter.sampling_frequency / 2.0;
        processing.decade = (int)(log10(filter.nyquist_frequency / processing.minimum_frequency )) + 1;
        output.sampling_point = processing.mesh * processing.decade + 1;
        output.frequency      = (double *)calloc((output.sampling_point+1), sizeof(double));
        output.psd            = (double *)calloc((output.sampling_point+1), sizeof(double));
        if ( output.frequency == NULL || output.psd == NULL ) {
        sprintf(G_reportbuff[0], "Fatal conditions with Out of memory¥n" );
        W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
            continue;
        }
        Psd_Output(&processing, &output);
       Psd_Calculation(&processing, &filter, &output);
        Psd_Output(&processing, &output);
        free(output.frequency);
        free(output.psd);
    }
    fclose(fin_control);

    return SUCCESS;

}
```

E-2 po3_read.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/                                                                              /
/  Frequency Characteristic Calculation Proguram P03.                          /
/                                                                              /
/                                                                              /
/                                                                              /
---------1---------2---------3---------4---------5---------6---------7------- */

/* ---------+---------+---------+---------+---------+---------+---------+------ */
/* standard include defines.                                                   */
```

```
/* ------+---------+---------+---------+---------+---------+---------+--------- */


/* ------+---------+---------+---------+---------+---------+---------+--------- */
/* private include defines.                                                      */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
#include "DFN_po3.h"
#include "VAR_po3.h"
#include "PRT_po3.h"


/* ------+---------+---------+---------+---------+---------+---------+--------- */
/* Format      : int ReadFileCall(char *filename,                                */
/*                                S_PROCESSING *processing,                      */
/*                                S_FILTER *filter)                              */
/* Prototype   : PRT_po3.h                                                       */
/* Discription : Reading job file and setting parameter.                         */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_po3.h]          */
/*               abnormal termination -> FAULT   [defined in DFN_po3.h]          */
/* Variable    : *filename  ; (i  ) pointer to filenmae buffer.                  */
/*               *processing; ( o) pointer to structure contained processing*/
/*                              components.                                      */
/*               *filter    ; ( o) pointer to structure contained filter       */
/*                              design.                                          */
/*。。Revision。。 : 2000/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
int  ReadFileCall(char *filename, S_PROCESSING *processing, S_FILTER *filter)
{
    FILE    *fin_setupfile, *fin_filter, *fout_test;
    char    input_buff[256];
    int     i, j, check;
    double      d_dummy;


    /* Setup file open */
    if ( (fin_setupfile = fopen(filename, "rt")) == NULL ) {
        sprintf(G_reportbuff[0], "Cannot open setup file [ %s ].¥n", filename);
        sprintf(G_reportbuff[1], "Abort this job.¥n", filename);
        W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);
        return FAULT;
    }


    /* Initialize */
    memset(processing, 0x00, sizeof(S_PROCESSING));
    memset(filter,     0x00, sizeof(S_FILTER));


    /* Reading parameter in setup file */
    for ( i = 0; i < 5; i++ ) {
        if ( Std_Reading(fin_setupfile, input_buff, i) == FAULT ) {
            sprintf(G_reportbuff[0], "Error: Format of setup file is unsuitable.¥n");
            goto ERROR_RECOVERY;
        }
        switch ( i ) {

            case 0: /* Filter filename */
                if ( (fin_filter = fopen(input_buff, "rt")) == NULL ) {
                    sprintf(G_reportbuff[0], "Cannot open the filter file.¥n" );
                    goto ERROR_RECOVERY;
                } else {
                    strcpy(processing->filter_name, input_buff);
                    check = Filter_File_Reading(fin_filter, filter);
                    fclose(fin_filter);
```

```
                if ( check == FAULT ) {
                    sprintf(G_reportbuff[0],
                        "Filter file [ %s ] is not aveirable file.¥n", input_buff);
                    goto ERROR_RECOVERY;
                }
            }
            break;

        case 1: /* minimum frequency */
            if ( strcmp(input_buff, "AUTO") == 0 ) {
                d_dummy = floor(log10(filter->sampling_frequency)) - 5.0;
                processing->minimum_frequency = pow(10, d_dummy);
            } else {
             processing->minimum_frequency = atof(input_buff);
             if ( processing->minimum_frequency <= 0.0 ) {
                sprintf(G_reportbuff[0],
                        "Illegal value as minimum frequency.¥n");
                goto ERROR_RECOVERY;
             }
             }
            break;

        case 2: /* mesh per decade */
            processing->mesh = atoi(input_buff);
            if ( processing->mesh <= 0 ) {
               sprintf(G_reportbuff[0], "Illegal value as mesh per decade.¥n");
               goto ERROR_RECOVERY;
            }
            break;

         case 3: /* output filename  */
             if ( (fout_test = fopen(input_buff, "wt")) == NULL ) {
                 sprintf(G_reportbuff[0], "Unsuitable Output File Name.¥n");
                 goto ERROR_RECOVERY;
             }
             fclose(fout_test);
             strcpy(processing->output_name, input_buff);
             break;

        case 4: /* output type */
            if ( strcmp(input_buff, "PSD") == 0 ) {
               processing->output_type = PSD;
            } else if ( strcmp( input_buff, "DESIBEL" ) == 0 ) {
               processing->output_type = DESIBEL;
            } else {
               sprintf(G_reportbuff[0],
                       "Illegal selection as output type.¥n");
               goto ERROR_RECOVERY;
            }
            break;

    }
}

/* Output filter characteristic */
Output_2(filter);

fclose(fin_setupfile);
return SUCCESS;


ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    fclose(fin_setupfile);
```

```
    return FAULT;
}



/* ------+--------+--------+--------+--------+--------+-------- */
/* Format      : int Filter_File_Reading(FILE *fin, S_FILTER *filter)      */
/* Prototype   : PRT_po3.h                                                 */
/* Discription : Reading filter parameter and check the components.        */
/*               averaging.                                                */
/* Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h] */
/*               abnormal termination -> FAULT   [defined in DFN_digfil2.h] */
/* Variable    : *fin   ; (i  ) Pointer to FILE structure.                 */
/*               *filter; ( o) Pointer to the structure contained filter   */
/*                             components.                                 */
/*。。Revision。。 : 2001/01/01                                             */
/* ------+--------+--------+--------+--------+--------+-------- */
int  Filter_File_Reading(FILE *fin, S_FILTER *filter)
{

    char input_buff[256];
    int     i;


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->denominator_order = atoi(input_buff);
    if (filter->denominator_order < 0 || filter->denominator_order > 100 ||
                                *input_buff < '0' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as rounde part order.¥n");
        goto ERROR_RECOVERY;
    }

    for (i = 1; i <= filter->denominator_order; i++) {
        if ( Std_Reading2(fin, input_buff) == FAULT ) {
            goto ERROR_RECOVERY;
        }
        filter->denominator[i] = atof(input_buff);
        if (*input_buff < '-' || *input_buff > '9') {
            sprintf(G_reportbuff[0], "Illegal value as rounde part parameter.¥n");
            goto ERROR_RECOVERY;
        }
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->numerator_order = atoi(input_buff);
    if (filter->numerator_order < 0 || filter->numerator_order > 100 ||
                                *input_buff < '0' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as non rounde part order.¥n");
        goto ERROR_RECOVERY;
    }

    for (i = 1; i <= filter->numerator_order; i++) {
        if ( Std_Reading2(fin, input_buff) == FAULT ) {
            goto ERROR_RECOVERY;
        }
        filter->numerator[i] = atof(input_buff);
        if (*input_buff < '-' || *input_buff > '9') {
```

```
            sprintf(G_reportbuff[0], "Illegal value as non rounde part parameter.¥n");
            goto ERROR_RECOVERY;
        }
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->sampling_frequency = atof(input_buff);
    if (filter->sampling_frequency <= 0 || filter->sampling_frequency > 1.7E308 ||
                            *input_buff < '-' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as sampling frequency.¥n");
        goto ERROR_RECOVERY;
    }


    if ( Std_Reading2(fin, input_buff) == FAULT ) {
        goto ERROR_RECOVERY;
    }
    filter->gain = atof(input_buff);
    if (filter->gain <= 0 || filter->gain > 1.7E308 ||
                                        *input_buff < '-' || *input_buff > '9') {
        sprintf(G_reportbuff[0], "Illegal value as filter gain.¥n");
        goto ERROR_RECOVERY;
    }

    return SUCCESS;

ERROR_RECOVERY:
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    return FAULT;
}



/* ------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Std_Reading(FILE *fin, char *return_buff, int counter)  */
/*  Prototype   : PRT_po3.h                                                   */
/*  Discription : Standard reading function to setup file.                    */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h]  */
/*  Variable    : *fin        ; (i  ) file pointer to setup file.             */
/*                *return_buff; (  o) string buffer pointer for reading para- */
/*                                    meter.                                 */
/*                counter     ; (i  ) logical ID of parameter.                */
/*。。Revision。。 : 2001/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------- */
int  Std_Reading(FILE *fin, char *ptr1, int counter)
{

    char tmp_buff1[256];
    int      check;


    check = fscanf(fin, "%s %s", tmp_buff1, ptr1);
    sprintf( G_reportbuff[0], "%s %s¥n", tmp_buff1, ptr1 );
    W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );

    if ( (check != 2 ) || (strcmp( tmp_buff1, G_parameter_item[counter]) != 0) ) {
        return FAULT;
    }

    return SUCCESS;
```

```
}


/* ------+--------+--------+--------+--------+--------+-------- */
/*  Format      : int Std_Reading2(FILE *fin, char *return_buff)  */
/*  Prototype   : PRT_po3.h                                       */
/*  Discription : Standard reading function to filter file.       */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_digfil2.h] */
/*                abnormal termination -> FAULT   [defined in DFN_digfil2.h] */
/*  Variable    : *fin        : (i  ) file pointer to filter file.  */
/*                *return_buff: (  o) string buffer pointer for reading para- */
/*                                    meter.                     */
/*  ｡｡Revision｡｡  : 2001/01/01                                  */
/* ------+--------+--------+--------+--------+--------+-------- */
int  Std_Reading2(FILE *fin, char *ptr1)
{

    char tmp_buff1[256];
    int      check;


    check = fscanf(fin, "%s", ptr1);

    if ( check != 1 ) {
        sprintf(G_reportbuff[0], "Error: Format of filter file is unsuitable.¥n");
        return FAULT;
    }

    return SUCCESS;
}
```

E-3 po3_output.c

```
/* ------1--------2--------3--------4--------5--------6--------7--------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                    _/
_/  Frequency Characteristic Calculation Proguram P03.               _/
_/                                                                    _/
_/                                                                    _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
--------1--------2--------3--------4--------5--------6--------7-------- */

/* ------+--------+--------+--------+--------+--------+-------- */
/* standard include defines.                                     */
/* ------+--------+--------+--------+--------+--------+-------- */


/* ------+--------+--------+--------+--------+--------+-------- */
/* private include defines.                                      */
/* ------+--------+--------+--------+--------+--------+-------- */
#include "DFN_po3.h"
#include "VAR_po3.h"
#include "PRT_po3.h"


/* ------+--------+--------+--------+--------+--------+-------- */
/* static variable defines.                                      */
/* ------+--------+--------+--------+--------+--------+-------- */
```

```
/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : void Output_2(S_FILTER *filter)                               */
/*  prototype   : PTR_po3.h                                                     */
/*  Function    : Result output by formation type 2.                           */
/*  Return Value: none                                                          */
/*  Variable    : *filter; (i  ) Pointer to structure contained digital         */
/*                               component.                                     */
/*。。Revision。。 : 2001/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------ */
void Output_2(S_FILTER *filter)
{
    int     i;
    int     max_dimension;


    max_dimension = max(filter->denominator_order, filter->numerator_order);

    sprintf(G_reportbuff[0],"¥n" );
    sprintf(G_reportbuff[1],"                          -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[2],"                  1  + b1z + b2z + b3z + ... + bNz¥n");
    sprintf(G_reportbuff[3],"  H(z) = Gain ------------------------------------¥n");
    sprintf(G_reportbuff[4],"                          -1    -2    -3          -N¥n");
    sprintf(G_reportbuff[5],"                  1  + a1z + a2z + a3z + ... + bNz¥n¥n");
    sprintf(G_reportbuff[6],"  Gain  = %21.16e¥n", filter->gain);
    sprintf(G_reportbuff[7],"  N     = %d¥n", max_dimension);
    sprintf(G_reportbuff[8],"  Fs    = %21.16e¥n", filter->sampling_frequency );
    W_Writing( CRT_DISK, G_reportbuff, 9, G_fout_report );

    for (i = 1; i <= max_dimension; i++) {
        sprintf(G_reportbuff[0], "  a[%2d] = % 21.16e            b[%2d] = % 21.16e¥n",
                                 i, filter->denominator[i], i, filter->numerator[i]);
        W_Writing( CRT_DISK, G_reportbuff, 1, G_fout_report );
    }

}



/* ------+---------+---------+---------+---------+---------+---------+------ */
/*  Format      : void Psd_Output(S_PROCESSING *processing, S_OUTPUT *output) */
/*  prototype   : PTR_po3.h                                                     */
/*  Function    : Result output.                                                */
/*  Return Value: none                                                          */
/*  Variable    : *processing; (i  ) pointer to structure contained processing*/
/*                               components.                                    */
/*                *output     ; (i  ) Pointer to structure contained result.   */
/*。。Revision。。 : 2001/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------ */
void Psd_Output(S_PROCESSING *processing, S_OUTPUT *output)
{
    FILE*fout;
    int     i;

    fout = fopen(processing->output_name, "wt");

    fprintf(fout, "%d¥n", output->sampling_point);

    for (i = 0; i < output->sampling_point; i++) {
        fprintf(fout, "%e    %e¥n",output->frequency[i], output->psd[i]);
    }
    fclose(fout);
}
```

E-4 po3_psd.c

```
/* ----1-------2-------3-------4-------5-------6-------7-------8
//////////////////////////////////////////////////////////////////
//                                                              //
//  Frequency Characteristic Calculation Proguram P03.          //
//                                                              //
//                                                              //
//////////////////////////////////////////////////////////////////
--------1-------2-------3-------4-------5-------6-------7------- */

/* ------+-------+-------+-------+-------+-------+-------+------ */
/* standard include defines.                                    */
/* ------+-------+-------+-------+-------+-------+-------+------ */


/* ------+-------+-------+-------+-------+-------+-------+------ */
/* private include defines.                                     */
/* ------+-------+-------+-------+-------+-------+-------+------ */
#include "DFN_po3.h"
#include "VAR_po3.h"
#include "PRT_po3.h"


/* ------+-------+-------+-------+-------+-------+-------+------ */
/*  Format      : void Psd_Calculation(S_PROCESSING *processing,  */
/*                                     S_FILTER *filter,        */
/*                                     S_OUTPUT *output)        */
/*  Prototype   : PRT_po3.h                                      */
/*  Discription : Main routine of calcurating PSD.               */
/*  Return Value: none.                                          */
/*  Variable    : *processing: (i  ) pointer to structure contained processing*/
/*                                   components.                */
/*                *filter    : (i  ) pointer to structure contained filter    */
/*                                   design.                    */
/*                *output    : ( o) pointer to structure contained result.   */
/*  Revision    : 2001/01/01                                     */
/* ------+-------+-------+-------+-------+-------+-------+------ */
void Psd_Calculation(S_PROCESSING *processing,
                     S_FILTER *filter,
                     S_OUTPUT *output)
{
    int     i, j;
    double  sampling_time_T, denominator_real, denominator_imagi, numerator_real,
            numerator_imagi, omega;


    /* preparation */
    sampling_time_T      = 1 / filter->sampling_frequency;
    output->frequency[0]  = processing->minimum_frequency;
    for (i = 1; i <= output->sampling_point; i++) {
        output->frequency[i] = pow(10.0, (double)i/(double)processing->mesh) *
                                                        processing->minimum_frequency;
        if (output->frequency[i] >= filter->nyquist_frequency) {
            output->frequency[i] = filter->nyquist_frequency;
            output->sampling_point = i;
            break;
        }
    }
```

```
    /* calculation */
    for (i = 0; i <= output->sampling_point; i++) {
        omega = 2 * M_PI * output->frequency[i] * sampling_time_T;
        denominator_real  = 1.0;
        denominator_imagi = 0.0;
        for (j = 1; j <= filter->denominator_order; j++) {
            denominator_real  += filter->denominator[j] * cos(j * omega);
            denominator_imagi -= filter->denominator[j] * sin(j * omega);
        }
        numerator_real  = 1.0;
        numerator_imagi = 0.0;
        for (j = 1; j <= filter->numerator_order; j++) {
            numerator_real  += filter->numerator[j] * cos(j * omega);
            numerator_imagi -= filter->numerator[j] * sin(j * omega);
        }
        output->psd[i] = (SQR(numerator_real) + SQR(numerator_imagi)) /
                         (SQR(denominator_real) + SQR(denominator_imagi));

        output->psd[i] *= SQR(filter->gain);

        if ( processing->output_type == DESIBEL ) {
            output->psd[i] = sqrt(output->psd[i]);
        }


        if ( processing->output_type == DESIBEL ) {
            output->psd[i] = 20.0 * log10(output->psd[i]);
        }
    }

}
```

E-5 po3_util.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/                                                                              /
/  Frequency Characteristic Calculation Proguram P03.                          /
/                                                                              /
/                                                                              /
------1---------2---------3---------4---------5---------6---------7--------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_po3.h"
#include "VAR_po3.h"
#include "PRT_po3.h"



/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Format      : void W_Writing(int type,                                    */
/*                               char **string,                               */
/*                               int number                                   */
```

```
/*                              FILE *fout)                                  */
/*  Prototype   : PRT_po3.h                                                  */
/*  Description : Output strings to CRT & DISK.                              */
/*  Return Value: none                                                       */
/*  Variable    : type   ; (i  ) Output type ( CRT, DISK, CRT_DISK ).        */
/*                *string; (i  ) Pointer array of output strings buffer.     */
/*                number;  (i  ) Number of output.                           */
/*                *fout  ; (i  ) File pointer of ourput file.                */
/*  Revision    : 2001/01/01                                                 */
/* ------+--------+--------+--------+--------+--------+--------+----------- */
void  W_Writing(int type, char (*string)[256], int number, FILE *fout)
{
    int     i;

    for ( i = 0; i < number; i++ ) {
        if ( type & CRT ) {
            printf( "%s", string[i] );
        }

        if ( (type & DISK) && ( fout != NULL ) ) {
            fprintf( fout, "%s", string[i] );
        }
    }
}



/* ------+--------+--------+--------+--------+--------+--------+----------- */
/*  Format      : void Now_Time_Set(char *day_buff, char *time_buff)         */
/*  Prototype   : PRT_po3.h                                                  */
/*  Description : Get now time information.                                  */
/*  Return Value: calendar time at now.                                      */
/*  Variable    : *day_buff;  (i  ) pointer of string for day.               */
/*                *time_buff; (i  ) Pointer of string for time.              */
/*  Revision    : 2001/01/01                                                 */
/* ------+--------+--------+--------+--------+--------+--------+----------- */
time_t  Now_Time_Set(char *day_buff, char *time_buff)
{
    time_t      ct;
    struct tm   *now_time;

    ct = time( NULL );
    now_time = localtime( &ct );

    sprintf( day_buff,  "%04d/%02d/%02d",
                        now_time->tm_year+1900,
                        now_time->tm_mon+1,
                        now_time->tm_mday);

    sprintf( time_buff, "%02d:%02d:%02d",
                        now_time->tm_hour,
                        now_time->tm_min,
                        now_time->tm_sec );

    return ct;
}
```

E-6 PRT_po3.h

```
/* -------1---------2---------3---------4---------5---------6---------7--------8
/////////////////////////////////////////////////////////////////////////////
//                                                                         //
//  Frequency Characteristic Calculation Proguram P03.                     //
//                                                                         //
```

```
/*                                                                      */
/*                                                                      */
/*--------1---------2---------3---------4---------5---------6---------7--------- */

/* ==================================================================== */
/* prototype define.                                                    */
/* ==================================================================== */
/* ------+---------+---------+---------+---------+---------+---------- */
/* po3_main.c                                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void     Print_Usage(void);
extern void     Title(void);
extern int      AutoMain(char **);

/* ------+---------+---------+---------+---------+---------+---------- */
/* po3_read.c                                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
extern int      ReadFileCall(char *, S_PROCESSING *, S_FILTER *);
extern int      Filter_File_Reading(FILE *, S_FILTER *);
extern int      Std_Reading(FILE *, char *, int);
extern int      Std_Reading2(FILE *, char *);

/* ------+---------+---------+---------+---------+---------+---------- */
/* po3_psd.c                                                            */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void     Psd_Calculation(S_PROCESSING *, S_FILTER *, S_OUTPUT *);

/* ------+---------+---------+---------+---------+---------+---------- */
/* po3_output.c                                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void     Output_2(S_FILTER *);
extern void     Psd_Output(S_PROCESSING *, S_OUTPUT *);

/* ------+---------+---------+---------+---------+---------+---------- */
/* po3_util.c                                                           */
/* ------+---------+---------+---------+---------+---------+---------- */
extern void W_Writing(int, char (*)[256] ,int, FILE *);
extern time_tNow_Time_Set(char *, char *);
```

E-7 DFN_po3.h

```
/*--------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                      */
/*  Frequency Characteristic Calculation Proguram P03.                  */
/*                                                                      */
/*                                                                      */
/*--------1---------2---------3---------4---------5---------6---------7--------- */

/* ==================================================================== */
/* standard prototype include.                                          */
/* ==================================================================== */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
```

```
#include <math.h>
#include <time.h>
#include <sys/ddi.h>


/* ========================================================================== */
/* macro define.                                                              */
/* ========================================================================== */

/* Routine check */
#define  SUCCESS       1
#define NOTHING        0
#define FAULT     -1

/* Switch */
#define ON             1
#define OFF            0

/* Display switch in W_Writing function */
#define CRT            0x0001
#define DISK       0x0002
#define CRT_DISK   0x0003

/* Output type */
#define PSD            1
#define DESIBEL        2

/* SQR function macro */
#define SQR(x)         ((x)*(x))

/* ========================================================================== */
/* struct define.                                                             */
/* ========================================================================== */
typedef struct {
    int      denominator_order;        /* denominator order */
    double   denominator[101];     /* array of denominator value */
    int      numerator_order;      /* numerator order */
    double   numerator[101];           /* array of nemerator value */
    double   sampling_frequency;       /* sampling frequency */
    double   gain;                     /* filter gain */
    double   nyquist_frequency;        /* nyquist frequency */
} S_FILTER;

typedef struct {
    charfilter_name[256];     /* filter file name */
    double   minimum_frequency;        /* minimum frequency */
    int      decade;                   /* number of decade  */
    int      mesh;                     /* mesh per decade */
    charoutput_name[256];     /* input data file name */
    int      output_type;          /* output type */
} S_PROCESSING;

typedef struct {
    int      sampling_point;           /* number of sampling */
    double   *frequency;               /* output frequency */
    double   *psd;                     /* output psd */
} S_OUTPUT;


/* ========================================================================== */
/* enum define.                                                               */
/* ========================================================================== */
```

E-7 VAR_po3.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
//////////////////////////////////////////////////////////////////////////////
//                                                                          //
//  Frequency Characteristic Calculation Proguram P03.                      //
//                                                                          //
//                                                                          //
//////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------- */

/* ======================================================================== */
/* extern define.                                                           */
/* ======================================================================== */
#ifdef P03_MAIN
    #define EXT
#else
    #define EXT       extern
#endif

/* ======================================================================== */
/* varriable define.                                                        */
/* ======================================================================== */
/* ------------------------------ */
/* global.                        */
/* ------------------------------ */

/* output file name and type */
EXT char              G_out_filename[256];

/* string array of reading parameter item */
#ifdef P03_MAIN
char     *G_parameter_item[] = {
                               "FILTER_FILE_BY_DIGFIL1:::::::::::",
                              "MINIMUM_FREQUENCY:::::::::::::::::",
                              "MESH_PER_DECADE:::::::::::::::::::",
                               "OUTPUT_DATA_FILE:::::::::::::::::",
                              "OUTPUT_TYPE(PSD/DESIBEL)::::::::::" };
#else
extern char               *G_parameter_item[];
#endif

/* message buff array for W_Writing[in armax1_util.c] function */
EXT char              G_reportbuff[10][256];

/* file pointer for W_Writing[in armax1_util.c] function */
EXT FILE              *G_fout_report;

/* ------+---------+---------+---------+---------+---------+---------+------ */
#undef EXT
/* ------+---------+---------+---------+---------+---------+---------+------ */
```

F　FFT計算プログラムリスト

F-1： nucsys_main.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]           _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* MAIN FUNCTION SETTING.                                                     */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#define NUCSYS_MAIN

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_nucsys.h"
#include "VAR_nucsys.h"
#include "PRT_nucsys.h"


/* ---------------------------------------------------------------------- */
/* グローバル変数                                                         */
/* ---------------------------------------------------------------------- */
double      wmat[10][10];       /* マトリックス作業変数 */
char     message_buff[128];    /* 終了メッセージ格納バッファ */


/* ---------------------------------------------------------------------- */
/* 関数名        main                                                     */
/*                                                                        */
/* 形式          int main(int argc, char *argv[]);                        */
/*                                                                        */
/* 引数          argc   : コマンドラインからの引数の数                    */
/*               *argv[]: コマンドラインの引数の文字列を格納しているバ    */
/*                        ッファのポインタの配列                          */
/*                                                                        */
/* プロトタイプ  none                                                     */
/*                                                                        */
/* 戻り値        いかなる場合にも, SUCCESS(=1)を返します.                 */
/*                                                                        */
/* 解説          メインルーチン.                                          */
/* ---------------------------------------------------------------------- */
int main(int argc, char *argv[])
{
    FILE*fcont, *frepo;
    char working_buff[128];
    int      report;

    printf("  ARMA PARAMETER OF REACTOR SYSTEM CALCULATION   Ver. 1.0  ");

    if (argc < 2) {
        Usage_Info();
        printf("Try again, please.¥n");
        return SUCCESS;
    }

    printf("Control File : %s¥r¥n", argv[1]);
    if ( (fcont = fopen(argv[1], "rt")) == NULL ) {
        printf("Cannot open the control file.¥n");
        printf("%s", message_buff);
        return SUCCESS;
    }
    if ( (frepo = fopen(argv[2], "wt")) == NULL ) {
        printf("Cannot open the report file.¥n");
        return SUCCESS;
    }
```

```
    report = Main_Process(fcont, frepo);

    switch (report) {
        case SUCCESS:
            sprintf(message_buff, "Normal Termination");
            break;
        case FAULT:
            break;
    }
    printf("%s", message_buff);
    fprintf(frepo, "::::: %s :::::", message_buff);

    fclose(fcont);
    fclose(frepo);

    return SUCCESS;
}

/* ------------------------------------------------------------------ */
/* 関数名        Main_Process                                         */
/*                                                                    */
/* 形式          int Main_Process(FILE *fcont, FILE *fout);           */
/*                                                                    */
/* 引数          *fcont: コントロールファイルのファイル構造体を指すポイ*/
/*                       ンタ                                         */
/*               *fout : 出力ファイルのファイル構造体を指すポインタ    */
/*                                                                    */
/* プロトタイプ  PRT_nuosys.h                                         */
/*                                                                    */
/* 戻り値        正常にプロセスを終了した場合には, SUCCESS(=1)を返し,  */
/*               異常が生じた場合には, FAULT(=-1)を返します.          */
/*                                                                    */
/* 解説          コントロールファイルを解読しプロセスを逐次実行します. */
/* ------------------------------------------------------------------ */
int Main_Process(FILE *fcont, FILE *frepo)
{

    static PHY_CONST physical_const;
    static char         *const_field[]  =  {"LIFE_TIME:::",  "BETA:::::::::",  "RAMBDA:::::::",  "INTENSITY_S::",
"NEUTRON_NU:::", "SUMPLING_F:::"};
    static char         *unit_field[]  = {"[s]", "", "[1/s]", "[1/s]", "", "[Hz]"};

    FILE            *fout;
    char            input_buff1[32], input_buff2[32], input_buff3[32];
    int                 i, counter, scanfactor;
    double              *ptr, subcri, r;


    ptr = &physical_const.l;
    for (i = 0; i < 6; i++) {
        if (fscanf(fcont, "%s %s", input_buff1, input_buff2) != 2) {
            Common_Message(ILLEGAL_FORMAT_ERROR);
            return FAULT;
        }
        if ( strcmp(input_buff1, const_field[i]) != 0) {
            Common_Message(ILLEGAL_FORMAT_ERROR);
            return FAULT;
        }
        *ptr = atof(input_buff2);
        printf("%s %20.15e %s¥n", const_field[i], *ptr, unit_field[i]);
        fprintf(frepo, "%s %20.15e %s¥n", const_field[i], *ptr, unit_field[i]);
        ptr++;
    }
    fprintf(frepo, "¥n¥n");


    counter = 1;
    scanfactor = 1;
    while (ON) {
        scanfactor = fscanf(fcont, "%s %s %s", input_buff1, input_buff2, input_buff3);
        if (scanfactor != 3) {
            if (scanfactor == EOF)
                return SUCCESS;
            Common_Message(ILLEGAL_FORMAT_ERROR);
            return FAULT;
        }

        subcri = atof(input_buff1);
```

```
        r = atof(input_buff2);

        printf("[No.%2d] Subcriticality   = %20.15e [$]¥n", counter, subcri);
        printf("        Obsavation noise = %20.15e¥n", r);
        printf("        Output file name = %s¥n", input_buff3);
        fprintf(frepo, "[No.%2d] Subcriticality   = %20.15e [$]¥n", counter, subcri);
        fprintf(frepo, "        Obsavation noise = %20.15e¥n", r);
        fprintf(frepo, "        Output file name = %s¥n", input_buff3);


        if (subcri <= 0.0) {
            sprintf(message_buff, "Subcriticality must be zero or less than zero.¥n");
            return FAULT;
        }
        if (r < 0.0) {
            sprintf(message_buff, "Obsavation noise must be less than zero.¥n");
            return FAULT;
        }

        if ( (fout = fopen(input_buff3, "wt")) == NULL ) {
            sprintf(message_buff, "Cannot open the output file.¥n");
            return FAULT;
        }

        physical_const.subcri = subcri;
        physical_const.r      = r;

        Param_Calculation(frepo, fout, &physical_const);

        fprintf(frepo, "¥n¥n");
        fclose(fout);

        counter++;
    }
}



/* ------------------------------------------------------------------ */
/* 関数名        Common_Message                                       */
/*                                                                    */
/* 形式          void Common_Message(int item)                        */
/*                                                                    */
/* 引数          item: 共通エラー番号                                 */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          共通エラー番号で指定されたエラーメッセージをグローバル*/
/*               変数 message_buff に格納します.                      */
/* ------------------------------------------------------------------ */
void  Common_Message(int item)
{
    static char      *m_buff[] = {"Format of control file is illegal.",
                                  "Illegal matrix name has used.  Assigned names are from A to J."};


    sprintf(message_buff, "%s¥n", m_buff[item]);
}


/* ------------------------------------------------------------------ */
/* 関数名        Centering                                            */
/*                                                                    */
/* 形式          void Centering(char *source, char *destination);     */
/*                                                                    */
/* 引数          *source      : コピー元の文字列を格納したバッファのポイ*/
/*                              ンタ                                  */
/*               *destination: コピー先の文字列を格納したバッファのポイ*/
/*                              ンタ                                  */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */

/*                                                                    */
/* 解説          文字列を８０文字中にセンタリングします.             */
/* ------------------------------------------------------------------ */
void Centering(char *source, char *destination)
{
    char*ptr1;
```

```
    int     i, add_space, length, fine;


    ptr1 = destination;

    length = strlen(source);
    add_space = (80 - length) / 2;
    for (i = 0; i < add_space; i++, ptr1++)
        *ptr1 = ' ';
    for (i = 0; i < length; i++, source++, ptr1++)
        *ptr1 = *source;
    for (i = 0; i < add_space; i++, ptr1++)
        *ptr1 = ' ';
    *ptr1 = '¥0';
    fine = 80 - strlen(destination);
    for (i = 0; i < fine; i++, ptr1++)
        *ptr1 = ' ';
    *ptr1 = '¥0';

}

/* ------------------------------------------------------------------- */
/* 関数名        Usage_Info                                            */
/*                                                                     */
/* 形式          void Usage_Info(void)                                 */
/*                                                                     */
/* 引数          none                                                  */
/*                                                                     */
/* プロトタイプ  PRT_nucsys.h                                          */
/*                                                                     */
/* 戻り値        none                                                  */
/*                                                                     */
/* 解説          コマンドラインオプションの記述が不正な場合，利用方法等*/
/*               を簡潔に明記したメッセージルーチンです．              */
/* ------------------------------------------------------------------- */
void  Usage_Info(void)
{
    static char  *help_message[] = {
        "Usage :",
        "    NUCSYS [CONTROL FILE NAME] [REPORT FILE NAME]",
        "",
        "",
        "Function :",
        "    This program generates the ARMA parameter of the suboritical reactor system.",
        "",
        ""};

    int i;


    for (i = 0; i < 8; i++)
        printf("%s¥n", help_message[i]);

}
```

F-2： nucsys_mat.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                           _/
_/  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]          _/
_/                                                                           _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* standard include defines.                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------- */


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_nucsys.h"
#include "VAR_nucsys.h"
#include "PRT_nucsys.h"
```

```
/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Mul                                           */
/*                                                                    */
/* 形式          void Matrix_Mul(double ans[][4],                     */
/*                               double mat1[][4],                    */
/*                               double mat2[][4],                    */
/*                               int column,                          */
/*                               int row                              */
/*                               int common);                         */
/*                                                                    */
/* 引数          ans[][4] : 計算した結果を格納するマトリックス配列    */
/*               mat1[][4]: 乗算時の左側マトリックス配列              */
/*               mat2[][4]: 乗算時の右側マトリックス配列              */
/*               column   : 生成マトリックスの行数                    */
/*               row      : 生成マトリックスの列数                    */
/*               common   : 被演算マトリックスの共通サイズ           */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          mat1 で指定されたマトリックスと mat2 で指定されたマトリッ*/
/*               クスを mat1 * mat2 の順序で掛け算し, ans で指定されたマ*/
/*               トリックスに格納します.                              */
/* ------------------------------------------------------------------ */
void  Matrix_Mul(double ans[][4], double mat1[][4], double mat2[][4], int column, int row, int common)
{

    extern double    wmat[][10];

    int      i, j, k;
    double   sum;


    for (i = 0; i < column; i++)
        for (j = 0; j < row; j++) {
            sum = 0.0;
            for (k = 0; k < common; k++)
                sum += mat1[i][k] * mat2[k][j];
            wmat[i][j] = sum;
        }

    for (i = 0; i < column; i++)
        for (j = 0; j < row; j++)
            ans[i][j] = wmat[i][j];

}


/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Add                                           */
/*                                                                    */
/* 形式          void Matrix_Add(double ans[][4],                     */
/*                               double mat1[][4],                    */
/*                               double mat2[][4],                    */
/*                               int column,                          */
/*                               int row);                            */
/*                                                                    */
/* 引数          ans[][4] : 計算した結果を格納するマトリックス配列    */
/*               mat1[][4]: 加算時の左側マトリックス配列              */
/*               mat2[][4]: 加算時の右側マトリックス配列              */
/*               column   : 生成マトリックスの行数                    */
/*               row      : 生成マトリックスの列数                    */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          mat1 で指定されたマトリックスと mat2 で指定されたマトリッ*/
/*               クスを mat1 + mat2 の順序で加算し, ans で指定されたマト*/
/*               リックスに格納します.                                */
/* ------------------------------------------------------------------ */
void  Matrix_Add(double ans[][4], double mat1[][4], double mat2[][4], int column, int row)
{

    int      i, j;


    for (i = 0; i < column; i++)
        for (j = 0; j < row; j++)
            ans[i][j] = mat1[i][j] + mat2[i][j];
```

```
}
/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Sub                                           */
/*                                                                    */
/* 形式          void Matrix_Sub(double ans[][4],                     */
/*                               double mat1[][4],                    */
/*                               double mat2[][4],                    */
/*                               int column,                          */
/*                               int row);                            */
/*                                                                    */
/* 引数          ans[][4] : 計算した結果を格納するマトリックス配列    */
/*               mat1[][4]: 減算時の左側マトリックス配列              */
/*               mat2[][4]: 減算時の右側マトリックス配列              */
/*               column   : 生成マトリックスの行数                    */
/*               row      : 生成マトリックスの列数                    */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          mat1で指定されたマトリックスとmat2で指定されたマトリッ*/
/*               クスを mat1 - mat2 の順序で加算し, ansで指定されたマト*/
/*               リックスに格納します.                                */
/* ------------------------------------------------------------------ */
void  Matrix_Sub(double ans[][4], double mat1[][4], double mat2[][4], int column, int row)
{

    int     i, j;


    for (i = 0; i < column; i++)
        for (j = 0; j < row; j++)
            ans[i][j] = mat1[i][j] - mat2[i][j];


}


/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Tra                                           */
/*                                                                    */
/* 形式          void Matrix_Tra(double ans[][4],                     */
/*                               double mat1[][4],                    */
/*                               int column,                          */
/*                               int row);                            */
/*                                                                    */
/* 引数          ans[][4] : 計算した結果を格納するマトリックス配列    */
/*               mat1[][4]: 被演算マトリックス配列                    */
/*               column   : 被演算マトリックスの行数                  */
/*               row      : 被演算マトリックスの列数                  */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          mat1で指定されたマトリックスの転置行列をansで指定され*/
/*               たマトリックスに格納します.                          */
/* ------------------------------------------------------------------ */
void  Matrix_Tra(double ans[][4], double mat1[][4],int column, int row)
{

    extern double    wmat[][10];

    int     i, j;


    for (i = 0; i < row; i++)
        for (j = 0; j < column; j++)
            wmat[i][j] = mat1[j][i];

    for (i = 0; i < row; i++)
        for (j = 0; j < column; j++)
            ans[i][j] = wmat[i][j];

}


/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Uni                                           */
/*                                                                    */
/* 形式          void Matrix_Uni(double ans[][4],                     */
/*                               double factor,                       */
```

```
/*                              int n);                                 */
/*                                                                      */
/* 引数          ans[][4]: 計算した結果を格納するマトリックス配列       */
/*               factor  : 単位行列に施す係数                           */
/*               n       : 生成次数                                     */
/*                                                                      */
/* プロトタイプ  PRT_nucsys.h                                           */
/*                                                                      */
/* 戻り値        none                                                   */
/*                                                                      */
/* 解説          n で指定された次数の単位行列を factor で指定された値を掛 */
/*               けて ans で指定されたマトリックスに格納します.         */
/* -------------------------------------------------------------------- */
void  Matrix_Uni(double ans[][4], double factor, int n)
{

    int     i, j;


    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            ans[i][j] = 0.0;
        ans[i][i] = factor;
    }

}


/* -------------------------------------------------------------------- */
/* 関数名        Matrix_Zer                                             */
/*                                                                      */
/* 形式          void Matrix_Zer(double ans[][4],                       */
/*                               int column,                            */
/*                               int row);                              */
/*                                                                      */
/* 引数          ans[][4]: 計算した結果を格納するマトリックス配列       */
/*               column  : 被演算マトリックスの行数                     */
/*               row     : 被演算マトリックスの列数                     */
/*                                                                      */
/* プロトタイプ  PRT_nucsys.h                                           */
/*                                                                      */
/* 戻り値        none                                                   */
/*                                                                      */
/* 解説          ans で指定されたマトリックスを零行列にします.         */
/* -------------------------------------------------------------------- */
void  Matrix_Zer(double ans[][4], int column, int row)
{
    int     i, j;


    for (i = 0; i < column; i++)
        for (j = 0; j < row; j++)
            ans[i][j] = 0.0;

}


/* -------------------------------------------------------------------- */
/* 関数名        Matrix_Inv                                             */
/*                                                                      */
/* 形式          int  Matrix_Inv(double ans[][4],                       */
/*                               double mat1[][4],                      */
/*                               int n);                                */
/*                                                                      */
/* 引数          ans[][4]: 計算した結果を格納するマトリックス配列       */
/*               factor  : 被演算マトリックス配列                       */
/*               n       : 被演算マトリックス次数                       */
/*                                                                      */
/* プロトタイプ  PRT_nucsys.h                                           */
/*                                                                      */
/* 戻り値        計算が正常終了した場合及び, 中途放棄する場合のいずれに*/
/*               おいても SUCCESS(=1) を返します.                       */
/*                                                                      */
/* 解説          n 次の正方マトリックス mat1 の逆行列を ans で指定されたマ */
/*               トリックスに格納します.                                */
/* -------------------------------------------------------------------- */
int  Matrix_Inv(double ans[][4], double mat1[][4], int n)
{
    extern double    wmat[][10];
```

```
    int         i, j, k, noseq[10], ip, dummy_i;
    double      p, w, dummy_d;


    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            wmat[i][j] = mat1[i][j];


    if (n == 1) {
        if (wmat[0][0] != 0.0)
            mat1[0][0] = 1.0 / wmat[0][0];
        return SUCCESS;
    }


    for (i = 0; i < n; i++)
        noseq[i] = i;


    for (k = 0; k < n; k++) {
        p = 0.0;
        for (i = k; i < n; i++)
            if (p < fabs(wmat[i][0])) {
                p = fabs(wmat[i][0]);
                ip = i;
            }
        if (p < 1.0e-12) {
            printf("Operating pivot is very little, therefor terminate the calculation.");
            return SUCCESS;
        }
        dummy_i = noseq[ip];
        noseq[ip] = noseq[k];
        noseq[k] = dummy_i;

        for (j = 0; j < n; j++) {
            dummy_d = wmat[ip][j];
            wmat[ip][j] = wmat[k][j];
            wmat[k][j] = dummy_d;
        }

        w = wmat[k][0];
        for (j = 1; j < n; j++)
            wmat[k][j-1] = wmat[k][j] / w;
        wmat[k][n-1] = 1.0 / w;

        for (i = 0; i < n; i++) {
            if (i != k) {
                w = wmat[i][0];
                for (j = 1; j < n; j++)
                    wmat[i][j-1] = wmat[i][j] - w*wmat[k][j-1];
                wmat[i][n-1] = -w*wmat[k][n-1];
            }
        }
    }

    for (k = 0; k < n; k++) {
        for (j = k; j < n; j++) {
            if (noseq[j] == k) {
                noseq[j] = noseq[k];
                for (i = 0; i < n; i++) {
                    w = wmat[i][j];
                    wmat[i][j] = wmat[i][k];
                    wmat[i][k] = w;
                }
            }
        }
    }


    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            ans[i][j] = wmat[i][j];

    return SUCCESS;

}
```

F-3： nucsys_eig.c

F-4： nucsys_cal.c

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                              */
/*  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]             */
/*                                                                              */
/*                                                                              */
/* ------1---------2---------3---------4---------5---------6---------7--------- */

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* standard include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */

/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                     */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_nucsys.h"
#include "VAR_nucsys.h"
#include "PRT_nucsys.h"


/* ------------------------------------------------------------------------- */
/* 関数名        Param_Calculation                                           */
/*                                                                           */
/* 形式          void Param_Calculation(FILE *frepo,                         */
/*                                      FILE *fout,                          */
/*                                      PHY_CONST *physical_const);          */
/*                                                                           */
/* 引数          *frepo          : 計算過程表示ファイルの情報を格納した */
/*                                 ファイル構造体を指すポインタ              */
/*                 *fout           : 出力ファイルの情報を格納したファイル */
/*                                 構造体を指すポインタ                      */
/*                 *physical_const: 原子炉及に関する物理情報及び計算情報 */
/*                                 を格納した構造体を指すポインタ            */
/*                                                                           */
/* プロトタイプ  PRT_nucsys.h                                                */
/*                                                                           */
/* 戻り値        none                                                        */
/*                                                                           */
/* 解説          与えられた原子炉定数及び計算情報からＡＲＭＡモデルパラ*/
/*               メータを導出します．計算の過程は，*frepo の示すファイル*/
/*               に逐次出力します．                                          */
/* ------------------------------------------------------------------------- */
int  Param_Calculation(FILE *frepo, FILE *fout, PHY_CONST *physical_const)
{

    int         i, j;
    double      a[4][4], q[4][4], se[4][4], d[4][4], ex1[4][4], ex2[4][4];
    double      phi[4][4], ex1t[4][4], ex2t[4][4], w1[4][4], w2[4][4];
    double      w3[4][4], w4[4][4], w[4][4], phi_t[4][4], h[4][4], h_t[4][4];
    double      phi_inv[4][4], r[4][4], r_inv[4][4], hh1[4][4], hh2[4][4];
    double      hh3[4][4], hh[4][4], p1[4][4], p2[4][4], p[4][4], rp[4][4];
    double      rp1[4][4], rp2[4][4], rp_dummy[4][4];
    double      eigen_vector[4][4], eigen_value[4];
    double      ar[2], ma[2], re[2], im[2];
    double      keff, st, coeff_q, eig_b, eta1, eta2, ex_eta1, ex_eta2;
    double      factor1, factor2, factor3, factor4, dummy_d;
    char    string_buff[128];


    /* Inialization */
    keff = 1.0 / (1.0 + physical_const->beta*physical_const->subcri);
    st = 1.0 / physical_const->sf;
    Matrix_Zer(h, 2, 2);
    Matrix_Zer(h_t, 2, 2);
    Matrix_Zer(r, 2, 2);
    h[0][0] = 1.0;
    h[0][1] = 0.0;
    h_t[0][0] = 1.0;
    h_t[1][0] = 0.0;
    r[0][0] = physical_const->r;
    r_inv[0][0] = 1.0 / physical_const->r;
    printf("        k-effective      = %20.15e¥n¥n", keff);
    fprintf(frepo, "        k-effective      = %20.15e¥n¥n", keff);
```

```
/* generation of the state matrix */
a[0][0] = ((1.0 - physical_const->beta)*keff - 1.0) / physical_const->l;
a[0][1] = physical_const->rambda;
a[1][0] = physical_const->beta * keff / physical_const->l;
a[1][1] = -physical_const->rambda;
sprintf(string_buff, "State matrix A");
Matrix_Print(frepo, string_buff, a);


/* generation of the binary noise equivalnat source */
coeff_q = (physical_const->s * keff) / (physical_const->nu * (1.0 - keff));
dummy_d = physical_const->nu*(physical_const->nu - 1.0);
q[0][0] = coeff_q * dummy_d * (1.0 - physical_const->beta) * (1.0 - physical_const->beta);
q[0][1] = coeff_q * dummy_d * physical_const->beta * (1.0 - physical_const->beta);
q[1][0] = q[0][1];
q[1][1] = coeff_q * dummy_d * physical_const->beta * physical_const->beta;
sprintf(string_buff, "Random force term -> binary noise equivalant source Q");
Matrix_Print(frepo, string_buff, q);


/* generation of the single noise equivalant source */
se[0][0] = 2.0 * (1.0 - (1.0 - physical_const->beta)*keff) * physical_const->s / (1.0 - keff);
se[0][1] = -2.0 * physical_const->s * physical_const->beta * keff / (1.0 - keff);
se[1][0] = se[0][1];
se[1][1] = -se[0][1];
sprintf(string_buff, "Random force term -> single noise equivalant source ΛE+EΛ");
Matrix_Print(frepo, string_buff, se);


/* generation of the diffusion constant matrix */
d[0][0] = q[0][0] + se[0][0];
d[0][1] = q[0][1] + se[0][1];
d[1][0] = q[1][0] + se[1][0];
d[1][1] = q[1][1] + se[1][1];
sprintf(string_buff, "Diffusion constant matrix 2D");
Matrix_Print(frepo, string_buff, d);


/* calculation of Φ by sysIvester's eigen value expansion method */
eig_b = (physical_const->l*physical_const->rambda - (1.0 - physical_const->beta)*keff + 1.0) / physical_const->l;
dummy_d = eig_b * eig_b + 4.0*(keff-1.0)*physical_const->rambda/physical_const->l;
dummy_d = sqrt(dummy_d);
eta1 = (-eig_b + dummy_d)*0.5;
eta2 = (-eig_b - dummy_d)*0.5;
for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++)
        ex1[i][j] = ex2[i][j] = a[i][j];
    ex1[i][i] -= eta1;
    ex2[i][i] -= eta2;
}
ex_eta1 = exp(eta1*st);
ex_eta2 = exp(eta2*st);
dummy_d = 1.0 / (eta1 - eta2);
for (i = 0; i < 2; i++)
    for (j = 0; j < 2; j++)
        phi[i][j] = dummy_d * (ex2[i][j]*ex_eta1 - ex1[i][j]*ex_eta2);
sprintf(string_buff, "System regression matrix Φ");
Matrix_Print(frepo, string_buff, phi);


/* calculation of variance of the randam force */
for (i = 0; i < 2; i++)
    for (j = 0; j < 2; j++) {
        ex1t[i][j] = ex1[j][i];
        ex2t[i][j] = ex2[j][i];
    }
Matrix_Mul(w1, ex2, d, 2, 2, 2);
Matrix_Mul(w1, w1, ex2t, 2, 2, 2);
Matrix_Mul(w2, ex2, d, 2, 2, 2);
Matrix_Mul(w2, w2, ex1t, 2, 2, 2);
Matrix_Mul(w3, ex1, d, 2, 2, 2);
Matrix_Mul(w3, w3, ex2t, 2, 2, 2);
Matrix_Mul(w4, ex1, d, 2, 2, 2);
Matrix_Mul(w4, w4, ex1t, 2, 2, 2);
dummy_d = 1.0 / ((eta1 - eta2)*(eta1 - eta2));
```

```
    factor1 = dummy_d * (ex_eta1 * ex_eta1 - 1.0) / (2.0 * eta1);
    factor2 = dummy_d * (-(ex_eta1*ex_eta2) + 1.0) / (eta1 + eta2);
    factor3 = factor2;
    factor4 = dummy_d * (ex_eta2 * ex_eta2 - 1.0) / (2.0 * eta2);
    for (i = 0; i < 2; i++)
        for (j = 0; j < 2; j++) {
            w1[i][j] *= factor1;
            w2[i][j] *= factor2;
            w3[i][j] *= factor3;
            w4[i][j] *= factor4;
            w[i][j] = w1[i][j] + w2[i][j] + w3[i][j] + w4[i][j];
        }
    sprintf(string_buff, "Variance matrix of random force w");
    Matrix_Print(frepo, string_buff, w);


    /* Calculation of eigen value and eigen vector of intermediate matrix hh */
    Matrix_Zer(hh1, 2, 2);
    Matrix_Zer(hh2, 2, 2);
    Matrix_Zer(hh3, 2, 2);

    Matrix_Tra(phi_t, phi, 2, 2);
    Matrix_Inv(phi_inv, phi, 2);

    Matrix_Mul(hh1, h_t, r_inv, 2, 1, 1);
    Matrix_Mul(hh1, hh1, h, 2, 2, 1);
    Matrix_Mul(hh1, hh1, phi_inv, 2, 2, 2);
    Matrix_Mul(hh1, hh1, w, 2, 2, 2);
    Matrix_Add(hh1, hh1, phi_t, 2, 2);

    Matrix_Mul(hh2, h_t, r_inv, 2, 1, 1);
    Matrix_Mul(hh2, hh2, h, 2, 2, 1);
    Matrix_Mul(hh2, hh2, phi_inv, 2, 2, 2);

    Matrix_Mul(hh3, phi_inv, w, 2, 2, 2);

    for (i = 0; i < 2; i++)
        for (j = 0; j < 2; j++) {
            hh[i][j]     =  hh1[i][j];
            hh[i][j+2]   = -hh2[i][j];
            hh[i+2][j]   = -hh3[i][j];
            hh[i+2][j+2] =  phi_inv[i][j];
        }
    printf("Conduction matrix for seeking the solution of the Ricatti equation:¥n");
    fprintf(frepo, "Conduction matrix for seeking the solution of the Ricatti equation:¥n");
    for (i = 0; i < 4; i++) {
        printf("    % e    % e    % e    % e¥n", hh[i][0], hh[i][1], hh[i][2], hh[i][3]);
        fprintf(frepo, "    % e    % e    % e    % e¥n", hh[i][0], hh[i][1], hh[i][2], hh[i][3]);
    }
    printf("¥n");
    fprintf(frepo, "¥n");

    printf("Calculation of the eigen value & eigen vector of the conduction matrix:¥n");
    fprintf(frepo, "Calculation of the eigen value & eigen vector of the conduction matrix:¥n");
    Matrix_Eigen(frepo, hh, eigen_vector, eigen_value, 4);
    printf("¥n");
    fprintf(frepo, "¥n");
    printf("Summary of eigen value of the conduction matrix:¥n");
    fprintf(frepo, "Summary of eigen value of the conduction matrix:¥n");
    for (i = 0; i < 4; i++) {
        dummy_d = 1.0 / eigen_value[i];
        printf("    λ%d   % 20.15e    ---> inverse % 20.15e¥n", i, eigen_value[i], dummy_d);
        fprintf(frepo, "    λ%d   % 20.15e    ---> inverse % 20.15e¥n", i, eigen_value[i], dummy_d);
    }
    printf("¥n");
    fprintf(frepo, "¥n");


    printf("Eigen vector of the conduction matrix:¥n");
    fprintf(frepo, "Eigen vector of the conduction matrix:¥n");
    for (i = 0; i < 4; i++) {
        printf("    % e    % e    % e    % e¥n", eigen_vector[i][0], eigen_vector[i][1], eigen_vector[i][2],
eigen_vector[i][3]);
        fprintf(frepo, "    % e    % e    % e    % e¥n", eigen_vector[i][0], eigen_vector[i][1], eigen_vector[i][2],
eigen_vector[i][3]);
    }
    printf("¥n");
    fprintf(frepo, "¥n");


    /* Calculation of the P Matrix as the solution of the Ricatti eauation */
```

```
for (i = 0; i < 2; i++)
    for ( j = 0; j < 2; j++) {
        p2[i][j] = eigen_vector[i][j+2];
        p1[i][j] = eigen_vector[i+2][j+2];
    }

sprintf(string_buff, "Matrix constructed by partial eigen vector η");
Matrix_Print(frepo, string_buff, p1);
sprintf(string_buff, "Matrix constructed by partial eigen vector ξ");
Matrix_Print(frepo, string_buff, p2);

Matrix_Inv(p2, p2, 2);
Matrix_Mul(p, p1, p2, 2, 2, 2);
sprintf(string_buff, "P Matrix as the solution of the Ricatti equation");
Matrix_Print(frepo, string_buff, p);


/* Verification of Riccati equation */
Matrix_Mul(rp1, phi, p, 2, 2, 2);
Matrix_Mul(rp1, rp1, phi_t, 2, 2, 2);

Matrix_Mul(rp2, phi, p, 2, 2, 2);
Matrix_Mul(rp2, rp2, h_t, 2, 1, 2);
Matrix_Mul(rp_dummy, h, p, 1, 2, 2);
Matrix_Mul(rp_dummy, rp_dummy, h_t, 1, 1, 2);
Matrix_Add(rp_dummy, rp_dummy, r, 1, 1);
rp_dummy[0][0] = 1.0 / rp_dummy[0][0];
Matrix_Mul(rp2, rp2, h, 2, 2, 1);
Matrix_Mul(rp2, rp2, p, 2, 2, 2);
Matrix_Mul(rp2, rp2, phi_t, 2, 2, 2);
for (i = 0; i < 2; i++)
    for (j = 0; j < 2; j++)
        rp2[i][j] *= rp_dummy[0][0];

Matrix_Sub(rp, rp1, rp2, 2, 2);
Matrix_Add(rp, rp, w, 2, 2);

sprintf(string_buff, "Verificaltion: right side matrix of Riccati equation");
Matrix_Print(frepo, string_buff, rp);

Matrix_Sub(rp, p, rp, 2, 2);
printf("Error of P-Matrix¥n");
printf("    % 18.15f         % 18.15f¥n", rp[0][0], rp[0][1]);
printf("    % 18.15f         % 18.15f¥n¥n", rp[1][0], rp[1][1]);
fprintf(frepo, "Error of P-Matrix¥n");
fprintf(frepo, "    % 18.15f         % 18.15f¥n", rp[0][0], rp[0][1]);
fprintf(frepo, "    % 18.15f         % 18.15f¥n¥n", rp[1][0], rp[1][1]);


/* AR parameter & system pole */
ar[0] = -(phi[0][0] + phi[1][1]);
ar[1] = phi[0][0]*phi[1][1] - phi[0][1]*phi[1][0];
Equation2(1.0, ar[0], ar[1], re, im);
printf("AR parameter¥n");
fprintf(frepo, "AR parameter¥n");
printf("    AR1 = % 20.15e     AR2 = % 20.15e¥n", ar[0], ar[1]);
fprintf(frepo, "    AR1 = % 20.15e     AR2 = % 20.15e¥n", ar[0], ar[1]);
printf("System pole¥n");
fprintf(frepo, "System pole¥n");
for (i = 0; i < 2; i++) {
    printf("    z%d  = % 20.15e  +  % 20.15ej¥n", (i+1), re[i], im[i]);
    fprintf(frepo, "    z%d  = % 20.15e  +  % 20.15ej¥n", (i+1), re[i], im[i]);
}
printf("¥n");
fprintf(frepo, "¥n");


/* MA parameter & system zero */
dummy_d = p[0][0] + physical_const->r;
ma[0] = (p[0][1]*phi[0][1] - p[0][0]*phi[1][1] + physical_const->r*ar[0]) / dummy_d;
ma[1] = (phi[0][0]*phi[1][1] - phi[0][1]*phi[1][0])*physical_const->r / dummy_d;
Equation2(1.0, ma[0], ma[1], re, im);
printf("MA parameter¥n");
fprintf(frepo, "MA parameter¥n");
printf("    MA1 = % 20.15e     MA2 = % 20.15e¥n", ma[0], ma[1]);
fprintf(frepo, "    MA1 = % 20.15e     MA2 = % 20.15e¥n", ma[0], ma[1]);
printf("System zero¥n");
fprintf(frepo, "System zero¥n");
for (i = 0; i < 2; i++) {
    printf("    z%d  = % 20.15e  +  % 20.15ej¥n", (i+1), re[i], im[i]);
    fprintf(frepo, "    z%d  = % 20.15e  +  % 20.15ej¥n", (i+1), re[i], im[i]);
}
printf("¥n");
```

```
    fprintf(frepo, "¥n");

    /* Output to PO file */
    dummy_d = sqrt(p[0][0] + physical_const->r);
    fprintf(fout, "2¥n");
    for (i = 0; i < 2; i++)
        fprintf(fout, "% 20.15e¥n", ar[i]);
    fprintf(fout, "2¥n");
    for (i = 0; i < 2; i++)
        fprintf(fout, "% 20.15e¥n", ma[i]);
    fprintf(fout, "% 20.15e¥n", physical_const->sf);
    fprintf(fout, "% 20.15e¥n", dummy_d);

    return SUCCESS;
}

/* ------------------------------------------------------------------ */
/* 関数名        Matrix_Print                                         */
/*                                                                    */
/* 形式          void Matrix_Print(FILE *frepo,                       */
/*                                 char *str_ptr,                     */
/*                                 double matrix[][4]);               */
/*                                                                    */
/* 引数          *frepo     : 計算過程表示ファイルの情報を格納したファ */
/*                            イル構造体を指すポインタ                */
/*               *str_ptr   : 過程の内容を格納したバッファを指すポイン */
/*                            タ                                      */
/*               matrix[][4]: 表示すべき行列を格納した配列            */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          ディスプレイ及び*frepo で示されたファイルに，計算過程の*/
/*               内容とマトリックスの要素を表示します．               */
/* ------------------------------------------------------------------ */
void  Matrix_Print(FILE *frepo, char *str_ptr, double matrix[][4])
{
    printf("%s:¥n", str_ptr);
    printf("    % 20.15e     % 20.15e¥n", matrix[0][0], matrix[0][1]);
    printf("    % 20.15e     % 20.15e¥n¥n", matrix[1][0], matrix[1][1]);
    fprintf(frepo, "%s:¥n", str_ptr);
    fprintf(frepo, "    % 20.15e     % 20.15e¥n", matrix[0][0], matrix[0][1]);
    fprintf(frepo, "    % 20.15e     % 20.15e¥n¥n", matrix[1][0], matrix[1][1]);
}

/* ------------------------------------------------------------------ */
/* 関数名        Equation2                                            */
/*                                                                    */
/* 形式          void Equation2(double a,                             */
/*                              double b,                             */
/*                              double c,                             */
/*                              double re[],                          */
/*                              double im[]);                         */
/*                                                                    */
/* 引数          a   : 二次方程式の二次の係数                         */
/*               b   : 二次方程式の一次の係数                         */
/*               c   : 二次方程式の定数項                             */
/*               re[]: 解の実部を格納する配列                         */
/*               im[]: 解の虚部を格納する配列                         */
/*                                                                    */
/* プロトタイプ  PRT_nucsys.h                                         */
/*                                                                    */
/* 戻り値        none                                                 */
/*                                                                    */
/* 解説          二次方程式を解の公式により解き，re[]，im[]に解を代入し*/
/*               ます．                                               */
/* ------------------------------------------------------------------ */
void  Equation2(double a, double b, double c, double re[], double im[])
{
    double      dummy_d;

    dummy_d = b*b - 4*a*c;
```

```
        if (dummy_d < 0) {
            re[0] = re[1] = -b / (2*a);
            dummy_d = sqrt(-dummy_d);
            im[0] = dummy_d;
            im[1] = -dummy_d;
        } else {
            re[0] = (-b + sqrt(dummy_d)) / (2*a);
            re[1] = (-b - sqrt(dummy_d)) / (2*a);
            im[0] = im[1] = 0.0;
        }
}
```

F-5： DFN_nucsys.h

```
/* ────1────────2────────3────────4────────5────────6────────7────────8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]             _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
────1────────2────────3────────4────────5────────6────────7──────── */

/* ================================================================ */
/* standard prototype include.                                      */
/* ================================================================ */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#include <time.h>


/* ================================================================ */
/* macro define.                                                    */
/* ================================================================ */
/* スイッチ用諸定義 */
#define ON          1
#define OFF         0
#define SUCCESS     1
#define FAULT       0
#define NOTHING    -1


/* 固有値・固有ベクトル演算処理結果定義 */
#define RESULT1     0       /* 単根収束 */
#define RESULT2     1       /* 異符号根 */
#define RESULT3     2       /* 共役複素根 */
#define RESULT4     3       /* 固有値0 */
#define RESULT5     4       /* 実根が非収束 */
#define RESULT6     5       /* 複素根が非収束 */


/* 共通エラーメッセージ番号定義 */
#define ILLEGAL_FORMAT_ERROR  0

/* ================================================================ */
/* struct define.                                                   */
/* ================================================================ */

typedef struct {
    double  l;                      /* 中性子寿命 */
    double  beta;                   /* 遅発中性子割合 */
    double  rambda;                 /* 先行核崩壊定数 */
    double  s;                      /* 中性子源強度 */
    double  nu;                     /* 中性子発生平均個数 */
    double  sf;                     /* サンプリング周波数 */
    double  subcri;                 /* 未臨界度 */
    double  r;
} PHY_CONST;                        /* 原子炉物理定数格納構造体 */


typedef struct {
    int     L;                      /* 固有値収束精度 */
    long    LIM;                    /* 反復回数の最大値 */
    int     IMAG;                   /* 複素固有値のテスト間隔 */
    int     M;                      /* 求める固有値の個数 */
} EIGENSET;                         /* 固有値算出パラメータ構造体 */
```

```
/* ========================================================================== */
/* enum define.                                                               */
/* ========================================================================== */
```

F-6： PRT_nucsys.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                            */
/*  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]           */
/*                                                                            */
/*                                                                            */
--------1---------2---------3---------4---------5---------6---------7------- */

/* ========================================================================== */
/* prototype define.                                                          */
/* ========================================================================== */
/* ------+---------+---------+---------+---------+---------+---------+------- */
/* nucsys_main.c                                                              */
/* ------+---------+---------+---------+---------+---------+---------+------- */
extern int      Main_Process(FILE *fcont, FILE *frepo);
extern void     Common_Message(int item);
extern void     Centering(char *source, char *destination);
extern void     Usage_Info(void);

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* nucsys_cal.c                                                               */
/* ------+---------+---------+---------+---------+---------+---------+------- */
extern int      Param_Calculation(FILE *frepo, FILE *fout, PHY_CONST *physical_const);
extern void     Matrix_Print(FILE *frepo, char *str_ptr, double matrix[][4]);
extern void     Equation2(double a, double b, double c, double re[], double im[]);

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* nucsys_mat.c                                                               */
/* ------+---------+---------+---------+---------+---------+---------+------- */
extern void     Matrix_Mul(double ans[][4], double mat1[][4], double mat2[][4], int column, int row, int
common);
extern void     Matrix_Add(double ans[][4], double mat1[][4], double mat2[][4], int column, int row);
extern void     Matrix_Sub(double ans[][4], double mat1[][4], double mat2[][4], int column, int row);
extern void     Matrix_Tra(double ans[][4], double mat1[][4], int column, int row);
extern void     Matrix_Uni(double ans[][4], double factor, int n);
extern void     Matrix_Zer(double ans[][4], int column, int row);
extern int      Matrix_Inv(double ans[][4], double mat1[][4], int n);

/* ------+---------+---------+---------+---------+---------+---------+------- */
/* nucsys_eig.c                                                               */
/* ------+---------+---------+---------+---------+---------+---------+------- */
extern int      Matrix_Eigen(FILE *frepo,  double mat1[][4], double eigen_vector[][4], double eigen_value
[], int n);
extern int      Eigen_Sub(double wmat[][10], int n, EIGENSET *eigenset, double *val1, double *val2, double e_vec1[], double
e_vec2[], double *eps, long *number, int IT);
extern void     Eigen_Prediction(double wmat[][10], int n, double u0[], double u1[], double *b1, int IT);
extern double   Eigen_Vectornorm(int n, double a[], double b[]);
extern int      Eigen_Complex(EIGENSET *eigenset, int n, double delta, int *imag1, double u0[], double u1[], double u2[], double
b1, double b2, double *sasa, double *pp, double *wlw1);
extern double   pow10(double value);
extern double   dmax(double value1, double value2);
```

F-7： VAR_nucsys.h

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                            */
/*  未臨界原子炉 ARMA モデル生成プログラム [リカッチ方程式直接解法]           */
/*                                                                            */
/*                                                                            */
--------1---------2---------3---------4---------5---------6---------7------- */

/* ========================================================================== */
/* extern define.                                                             */
/* ========================================================================== */
#ifdef NUCSYS_MAIN
```

```
    #define EXT
#else
    #define EXT       extern
#endif

/* ========================================================================== */
/* varriable define.                                                          */
/* ========================================================================== */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* global.                                                                    */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
```

G　UNIXプログラムの使用法

FFT，DIGFIL1，DIGFIL2，PO3，NUCSYS，の起動法はすべて同じ形式になっており、NUCSYSを例に説明する。

```
nucsys.out  control_file  report_file
```

を実行すれば、control_fileに書かれた条件で計算が行われ、指定のファイルに結果を書き出す。
実行の過程はCRT上及びreport_fileで確認できる。

control_file　の内容：

```
test1
test2
```

test1の内容：

```
FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
MINIMUM_FREQUENCY:::::::::::::::::: AUTO
MESH_PER_DECADE::::::::::::::::::: 100
OUTPUT_DATA_FILE::::::::::::::::::: t1.txt
OUTPUT_TYPE(PSD/DESIBEL)::::::::::: PSD
```

test2の内容：

```
FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
MINIMUM_FREQUENCY:::::::::::::::::: AUTO
MESH_PER_DECADE::::::::::::::::::: 100
OUTPUT_DATA_FILE::::::::::::::::::: t2.txt
OUTPUT_TYPE(PSD/DESIBEL)::::::::::: DESIBEL
```

lpf1.txtの内容：

```
9
-5.7969897498483780e+00
 1.5336905888046468e+01
-2.4196527771450533e+01
 2.5009388335576475e+01
-1.7521711999881791e+01
 8.3058885232390214e+00
-2.5650802822519383e+00
 4.6773151609422703e-01
-3.8329191427588602e-02
9
 9.0000000000000000e+00
 3.6000000000000000e+01
 8.4000000000000000e+01
```

1.2600000000000000e+02
1.2600000000000000e+02
8.3999999999999986e+01
3.5999999999999993e+01
9.0000000000000000e+00
1.0000000000000000e+00
8.8888887999999997e+02
2.4907579999182440e-06

t1.txt(report file)の内容

── P03 Execution ──────────────────────────────────────

JOB No. 1
Setup File Name = test1

FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
MINIMUM_FREQUENCY:::::::::::::::: AUTO
MESH_PER_DECADE:::::::::::::::::: 100
OUTPUT_DATA_FILE::::::::::::::::: t1.txt
OUTPUT_TYPE(PSD/DESIBEL)::::::::: PSD

$$H(z) = Gain \frac{1 + b1z^{-1} + b2z^{-2} + b3z^{-3} + \ldots + bNz^{-N}}{1 + a1z^{-1} + a2z^{-2} + a3z^{-3} + \ldots + bNz^{-N}}$$

Gain = 2.4907579999182440e-06
N = 9
Fs = 8.8888887999999997e+02

a[1] =	-5.7969897498483780e+00	b[1] =	9.0000000000000000e+00
a[2] =	1.5336905888046468e+01	b[2] =	3.6000000000000000e+01
a[3] =	-2.4196527771450533e+01	b[3] =	8.4000000000000000e+01
a[4] =	2.5009388335576475e+01	b[4] =	1.2600000000000000e+02
a[5] =	-1.7521711999881791e+01	b[5] =	1.2600000000000000e+02
a[6] =	8.3058885232390214e+00	b[6] =	8.3999999999999986e+01
a[7] =	-2.5650802822519383e+00	b[7] =	3.5999999999999993e+01
a[8] =	4.6773151609422703e-01	b[8] =	9.0000000000000000e+00
a[9] =	-3.8329191427588602e-02	b[9] =	1.0000000000000000e+00

── P03 Execution ──────────────────────────────────────

JOB No. 2
Setup File Name = test2

FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
MINIMUM_FREQUENCY:::::::::::::::: AUTO
MESH_PER_DECADE:::::::::::::::::: 100

```
OUTPUT_DATA_FILE:::::::::::::::::: t2.txt
OUTPUT_TYPE(PSD/DESIBEL)::::::::: DESIBEL

                      -1    -2    -3          -N
                1  + b1z + b2z + b3z + ... + bNz
  H(z) = Gain ------------------------------------
                      -1    -2    -3          -N
                1  + a1z + a2z + a3z + ... + bNz

  Gain  = 2.4907579999182440e-06
  N     = 9
  Fs    = 8.8888887999999997e+02
  a[ 1] = -5.7969897498483780e+00          b[ 1] =  9.0000000000000000e+00
  a[ 2] =  1.5336905888046468e+01          b[ 2] =  3.6000000000000000e+01
  a[ 3] = -2.4196527771450533e+01          b[ 3] =  8.4000000000000000e+01
  a[ 4] =  2.5009388335576475e+01          b[ 4] =  1.2600000000000000e+02
  a[ 5] = -1.7521711999881791e+01          b[ 5] =  1.2600000000000000e+02
  a[ 6] =  8.3058885232390214e+00          b[ 6] =  8.3999999999999986e+01
  a[ 7] = -2.5650802822519383e+00          b[ 7] =  3.5999999999999993e+01
  a[ 8] =  4.6773151609422703e-01          b[ 8] =  9.0000000000000000e+00
  a[ 9] = -3.8329191427588602e-02          b[ 9] =  1.0000000000000000e+00
========================== NORMAL TERMINATION ==========================
```

以下に、各プログラムの control_file を示す。

Fft control_file

```
RECORD_LENGTH:::::::::::::::::::: 2
DATA_FILENAME:::::::::::::::::::: /home/asuka/DIGFIL/SRC/digfil2/filt.dat
NUMBER_OF_SKIPPING_DATA:::::::::: 0
SAMPLING_FREQUENCY:::::::::::::::: 888.888888
1_BLOCK_NUMBER:::::::::::::::::::: 1024
OVERRAP_NUMBER:::::::::::::::::::: 0
AVERAGING_NUMBER:::::::::::::::::: MAX
WINDOW_FUNCTION::::::::::::::::::: HANNING
SPECTRUM_TYPE:::::::::::::::::::: PSD
DC_COMPONENT_FILTER:::::::::::::: OFF
OUTPUT_TYPE:::::::::::::::::::::: PL3
OUTPUT_FILENAME::::::::::::::::::: fft.txt
```

DigFil1 control_file

```
FILTER_FORM:::::::::::::::::::::: IIR
FILTER_TYPE:::::::::::::::::::::: LPF
AMPLITUDE_CHARACTERISTIC::::::::: BUTTERWORTH
SAMPLING_FREQUENCY:::::::::::::::: 888.88888
```

```
1ST_CUTOFF_FREQUENCY:::::::::::::: 100.0
1ST_STOPPING_FREQUENCY:::::::::::: 200.0
2ND_CUTOFF_FREQUENCY:::::::::::::: INEFFECTIVE
2ND_STOPPING_FREQUENCY:::::::::::: INEFFECTIVE
PASS_RIPPLE:::::::::::::::::::::::: 20.0
REJECTION_POWER:::::::::::::::::::: 80.0
DIGFIL2_FORM_OUTPUT_FILENAME::::: lpf1.txt
```

DigFil2 control_file

```
FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
INPUT_RECORD_RENGTH(2/8)::::::::::: 2
BINARY_DATA_FILE::::::::::::::::::: /home/asuka/DATA/test
SKIP_DATA:::::::::::::::::::::::::: 0
NUMBER_OF_DATA:::::::::::::::::::: MAX
OUTPUT_RECORD_LENGTH(2/8)::::::::: 2
OUTPUT_DATA_FILE::::::::::::::::::: ./filt.dat
```

P03 control_file

```
FILTER_FILE_BY_DIGFIL1:::::::::::: lpf1.txt
MINIMUM_FREQUENCY:::::::::::::::::: AUTO
MESH_PER_DECADE::::::::::::::::::: 100
OUTPUT_DATA_FILE::::::::::::::::::: t1.txt
OUTPUT_TYPE(PSD/DESIBEL)::::::::::: PSD
```