JNC TJ8400 2000-051

オンライン
臨界安全監視システムの
開発研究(Ⅳ)

(核燃料サイクル開発機構　委託研究成果報告書)

2000年2月

摂南大学 工学部 電気工学科
山田　　澄

JNC TJ8400 2000-051
2000年2月

オンライン臨界安全監視システムの開発研究（Ⅳ）

山田　澄*

要　旨

使用済み核燃料再処理工場等にける安全性の確保のために、安価で即応性の良い臨界安全監視システムの開発が望まれており、基礎研究による知見を基に具体的な臨界安全監視システムとして実現することを目的として開発研究が進められている。

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のため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も行ってきたが、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、今や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きており、そ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、今年度より、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。今年度は、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、本報告書として自己完結性を保つため、理論的な未臨界原子炉のＡＲＭＡモデル導出を示すとともに、逐次型ＡＲＭＡモデル同定アルゴリズムに関する要点を再録するとともに、付録としてプログラム使用マニュアルをまとめて添付した。

※ 本報告書は、山田澄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　野尻一郎

＊摂南大学 工学部 電気工学科

JNC TJ8400 200-051
February, 2000

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

Sumasu Yamada*

Abstract

Developing an inexpensive on-line criticality surveillance system is required for ensuring the safety of nuclear fuel reprocessing plants. Based on the series of researches for five years, R & D study on On-line Criticality Surveillance System has been carried out since 1996.

The concept of this Criticality Surveillance System is based on the Auto-Regressive Moving Average (ARMA) model identification algorithms to the time series of signal fluctuation of a neutron detector. We have proposed several new ideas of modification to the original design of the Criticality Surveillance System, and also reported some results of numerical analysis over the DCA experiments.

In those days, DOS/V personal computers with Microsoft Windows have came into wide use instead of those based on the MS-DOS, which have been popular in Japan. NEC, a major maker of MS-DOS computers, stopped the production of MS-DOS computers and changed their management policy toward production of DOS/V personal computers.

Our researches have been developed using MS-DOS computers. For the effective use of these important results, it became an urgent theme to transplant all programs developed on MS_DOS computers into computers with the OS, which is not easily affected by commercialism.

Since the design concept should be based on high reliability, electromagnetic disturbance-free and high expandability, and also computers have achieved remarkably high performance as well as low price in these days, these computers should be used not only as a simple signal processing unit but also a totally integrated signal analyzing system along with conventional signal analyzing software in stead of IC chips with analyzing soft wares. This configuration enables us to easily introduce newly developed techniques and to provide supplement information. Then, this approach can enhance the reliability of the Criticality Surveillance System without addition of any special devices, and also provide the flexibility of the system.

Under these circumstances, we decided to transplant all programs developed on the MS-DOS into those on the UNIX according to a yearly program. In this fiscal year, the recursive ARMA model identification program (ARMAX1), the main program of the criticality surveillance system, has been transplanted onto the UNIX.

For self-consistency, this report is incorporated with the operation manual of ARMAX1 as well as the basic theory of ARMA model of a subcritical reactor and the ARMA model identification algorithm.

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

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

目　次

図・表 目 次

1　緒　言

使用済み核燃料再処理工場等にける安全性の確保のために、安価で即応性の良い臨界安全監視システムの開発が望まれており、基礎研究による知見を基に具体的な臨界安全監視システムとして実現することを目的として開発研究が進められている。

本研究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しようとする観点から各種の検討を行ってきた。そのため、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システムの構成要素の各モジュールに関する検討を行うと共に、核燃料サイクル開発機構のＤＣＡ実験装置において行われた炉雑音実験データを解析するなど実用化に必要となる知見の蓄積も行ってきた。この間、計算機の環境が大きな変化を見せた。特にパーソナル計算機の進歩はめざましいものがあるが、わが国のプロセス制御系の計算機として支配的であったＭＳ-ＤＯＳ系の計算機が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、ＩＢＭ系のＤＯＳ／Ｖ系計算機がパーソナル計算機の中心となって、今や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きており、その研究成果を今後も有効に利用するためには、研究基盤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。臨界安全監視システムの様な特定の目的を持ったシステムでは、高速信号処理、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、一般に解析アルゴリズムはＲＯＭ化され、システムが固定化されてしまい、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。

この様な観点からも、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、今年度より、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。今年度は、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、報告書の一貫性を保つため、まず第二章には未臨界原子炉の物理パラメータとＡＲＭＡモデルのＡＲ、及びＭＡパラメータの理論的関係式を示し、第３章に逐次型 ARMA モデル同定アルゴリズムの基本原理を示し、第４章に今年度開発したＵＮＩＸ用の逐次型ＡＲＭＡＸモデル同定アルゴリズムのプログラムリストを示し、最後に第５章でその使用法を示す。

なお、ＭＳ－ＤＯＳ上で開発された同名のプログラムとの整合性を保つため、出力形式は全く変更されていない。

また、これまで長年にわたって蓄積されてきたバイナリーデータをＵＮＩＸ上でも有効に活用するため、ＭＳ-ＤＯＳ上のバイナリーデータをＵＮＩＸ上で利用できるようにするため、新たに変換プログラム、endian_conv.out が添付されている。

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。

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

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

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

$$D=Q+FE+EF^T \tag{2.2}$$

によって与えられる。ただし、

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

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

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

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、$\nu_0=\nu(1-\gamma\beta)$、$\nu_1=\nu\gamma\beta$、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

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

$$y(t)=Hx(t)+s(t) \tag{2.6.b}$$

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、

$$x(n+1)=\Phi x(n)+f(n+1) \tag{2.7.a}$$

$$x(n+1)=\Phi x(n)+f(n+1) \tag{2.7.b}$$

と書ける。ここで、

$$x(n)=x(n\Delta t) \tag{2.8}$$

$$y(n)=y(n\Delta t) \tag{2.9}$$

$$\Phi=\exp(F\Delta t) \tag{2.10}$$

であり、また$f(n)$、$v(n)$は

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

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

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

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

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

ただし、

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

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

によって与えられる。δmn 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

$$\begin{aligned}\Phi=\exp(F\Delta t)&=\begin{bmatrix}\Phi_{11} & \Phi_{12}\\ \Phi_{21} & \Phi_{22}\end{bmatrix}\\ &=\frac{\exp(\eta_+\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_- & \lambda\\ k\gamma\beta/l & -\lambda-\eta_-\end{bmatrix}-\frac{\exp(\eta_-\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_+ & \lambda\\ k\gamma\beta/l & -\lambda-\eta_+\end{bmatrix}\end{aligned} \tag{2.17}$$

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から得られるn時点での状態変数ベクトル$x(n)$の最適推定値

$y(n|n-1)$　：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

$$G(z) = \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.25}$$

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 a_1、a_2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 b_1、b_2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

$$y(n) = -a_1 y(n-1) - a_2(n-2) + e(n) + b_1 e(n-1) + b_2 e(n-2)$$

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

この章では、臨界安全監視システムにおいてオンライン未臨界度推定に用いられる中心的アルゴリズムである逐次型ＡＲＭＡ同定アルゴリズムについて述べる。

既に、これまでの報告書の中で繰り返し説明しているが、本報告書のまとまりと報告書としての一貫性を考慮し、再度この章を設けた。　ここでは、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。

$$A(z)y(t)=B(z)u(t)+C(z)e(t) \tag{3.2}$$

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

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

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

$$\theta^T=[a_1,a_2,\cdots,a_p,b_1,b_2,\cdots,b_r,c_1,c_2,\cdots,c_q] \tag{3.4}$$

と定義すると、残差（式誤差）$v(t)$

$$v(t)=y(t)-\theta^T(t)\phi(t) \tag{3.5}$$

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

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

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

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

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

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

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、

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

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

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

と定義し、更にこの逆行列の平均値行列Pを

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

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。

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

［ＳＴＥＰ２］

Prediction Error e_pの計算。

$$e_P = y(t) - \hat{y}(t) \tag{3.13}$$

［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。$\dfrac{1}{\hat{C}(z)}$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.

Symbol	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。

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

［ＳＴＥＰ８］

ゲインベクトル$L(t)$の更新。

$$L(t)=P(t)\varphi(t) \tag{3.22}$$

［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。

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 $(0 \le \mu_k \le 1)$を与え、K_tを

$$K_t = 1 - \mu_k' \tag{3.24}$$

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

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

（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

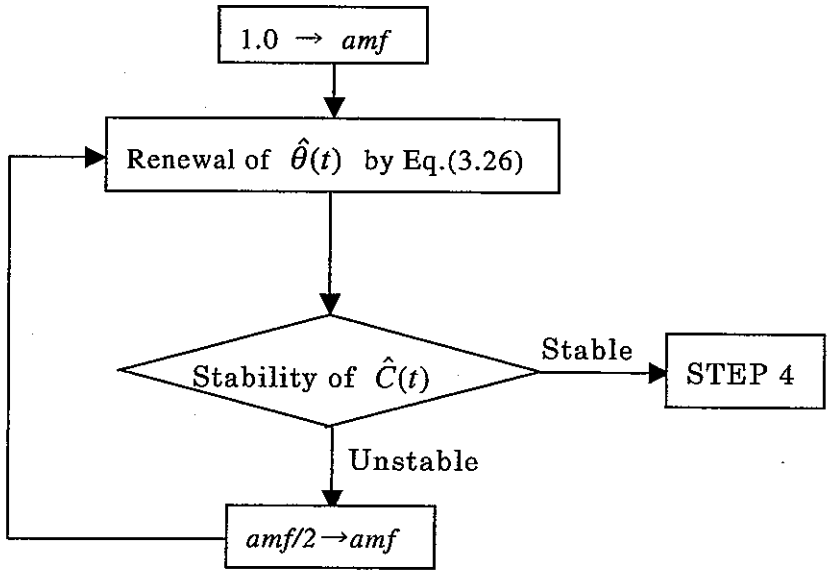
なタイプであると言え、ＲＰＥ法の中では最もシンプルな同定法であり計算量も最も少なくてすむ反面、同定対象依存性が強いなどの欠点をもつことが報告されている。

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、

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

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’］

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

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

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

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

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

となるように選べばよい。この様なθを$\hat{\theta}(N)$は、

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

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、

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

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

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

と定義し、更にこの逆行列の平均値行列Pは

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

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［STEP］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［STEP 9］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　ARMAX１プログラム

ここではＵＮＩＸ用に移植されたＡＲＭＡＸ１プログラムのリストを示しておく。
ＵＮＩＸ用のプログラムでは対話型のデータ入力プログラムが削除されているだけであり、基本的にはＭＳ－ＤＯＳ用のプログラムと同じで、より単純明解なプログラムとなっている。

以下にＡＲＭＡＸ１プログラムの構成を示す。

- armax1_main.c
- armax1_read.c
- armax1_call.c
- armax1_rpem.c
- armax1_output.c
- armax1_pole.c
- armax1_math.c
- armax1_util.c
- DFN_armax1.h
- VAR_armax1.h
- PRT_armax1.h

最後に、Report File の例を示した。

4.1 ARMAX1 Main Program

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/////////////////////////////////////////////////////////////////////////////////
/                                                                               /
/  ARMAX Parameter Calculation Program by RPE Method.                           /
/                                                                               /
/                                                                               /
/////////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7------- */

/* ----------+----------+----------+----------+----------+----------+---------- */
/* MAIN FUNCTION SETTING.                                                       */
/* ----------+----------+----------+----------+----------+----------+---------- */
#define ARMAX1_MAIN

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


/* ----------+----------+----------+----------+----------+----------+---------- */
/* private include defines.                                                     */
/* ----------+----------+----------+----------+----------+----------+---------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


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


/* ----------+----------+----------+----------+----------+----------+---------- */
/*  Format      : void main(int argc, char **argv)                              */
/*  prototype   : none                                                          */
/*  Function    : Main function for armax1 program.                             */
```

```
/*  Return Value: none                                                    */
/*  Variable    : argc  ; (i  ) Number of variable on the command line.   */
/*                **argv; (i  ) Pointer of strings on the command line.   */
/*  Revision    : 2000/01/01                                              */
/* ---------+---------+---------+---------+---------+---------+---------- */
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_armax1.h                                            */
/*  Function     : Display Usage                                           */
/*  Return Value: none                                                     */
/*  Variable     : none                                                    */
/*  Revision     : 2000/01/01                                              */
/* ---------+---------+---------+---------+---------+---------+---------- */
void  Print_Usage(void) {

    printf( "Usage:¥n" );
    printf( "  armax1  control_filename  report_filename¥n" );
    printf( "              control_filnema: path of control file.¥n" );
    printf( "              report_filnema : path of report file.¥n¥n" );

}
```

```
/* ---------+---------+---------+---------+---------+---------+---------- */
/*  Format       : void Title(void)                                        */
/*  prototype    : PRT_armax1.h                                            */
/*  Function     : Display Title                                           */
/*  Return Value: none                                                     */
/*  Variable     : none                                                    */
/*  Revision     : 2000/01/01                                              */
/* ---------+---------+---------+---------+---------+---------+---------- */
void  Title(void) {

    printf("¥n¥n");
   printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥
n");
    printf("_/
_/¥n");
    printf("_/ ARMAX1   ON-LINE IDENTIFICATION BY RECURSIVE PREDICTION ERROR METHOD   _/¥n");
    printf("_/
_/¥n");
    printf("_/      Version 1.01 : 2000/01/01 : transplantation from PC version.      _/¥n");
    printf("_/
_/¥n");
   printf("_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/¥
```

```
n");
    printf("¥n¥n");

}



/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format       : int AutoMain(void)                                              */
/*  prototype    : PRT_armax1.h                                                    */
/*  Function     : Display Title                                                   */
/*  Return Value: none                                                             */
/*  Variable     : none                                                            */
/*  Revision     : 2000/01/01                                                      */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
int  AutoMain(char **argv)
{
    FILE        *fin_control;
    char        setup_filename[256];
    int         count, check;
    char        answer;


    /* Initialize */
    count = 0;



    /* Indispensable 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],
        "—— ARMAX Identification
————————————————————————————————————————————————————¥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 = Armax_Call();
    }

    fclose(fin_control);

    return SUCCESS;
}
```

4.2 Parameter Calculation Program

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  ARMAX Parameter Calculation Program by RPE Method.                        _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */

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


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Formula     : void Armax_Call(void)                                      */
/*  Prototype   : PRT_armax1.h                                               */
/*  Description : ARMAX process routine.                                     */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]  */
/*  Arguments   : none                                                       */
/*  Revision    : 2000/01/01                                                 */
/* ------+---------+---------+---------+---------+---------+---------+------- */
int  Armax_Call(void)
{
    static char    *algorithm_string[] =  {"RPE", "PLR", "ELS", "RML"};
    static char    *activation_string[] = {"", "& PMA", "& PMDA"};
    FILE           *fin_y, *fin_u, *fout_result, *fout_pole[10], *fout_pl;
```

```
time_t          start_time, end_time;
char            day_buff[32], time_buff[32], polename_buff[256];
int             i, j, nnn;
int             method, u_order;
double          require_time;


/* Method information */
method = Method_Judgement();
if ( G_u_file.name[0] != NULL ) {
    u_order = G_rpemp.n;
} else {
    u_order = 0;
}
sprintf(G_reportbuff[0], "¥nARMAX(%d, %d, %d) Identification¥n",
        G_rpemp.n, u_order, G_rpemp.n );
sprintf(G_reportbuff[1], "Algorithm = %s %s¥n",
        algorithm_string[method], activation_string[G_subp.activation]);
W_Writing(CRT_DISK, G_reportbuff, 2, G_fout_report);


/* Start Check */
if ( Starting_Check() == FAULT ) {
    return FAULT;
}


/* Parameter Initilization */
if ( G_init_file.name[0] != NULL) {
    if ( Parameter_Init() == FAULT ) {
        return FAULT;
    }
} else {
    Parameter_Std_Init();
}


/* File open */
if ( (fin_y = fopen(G_y_file.name, "rb")) == NULL ) {
    sprintf(G_reportbuff[0], "Cannot open the outputted time series file.¥n");
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
```

```
        return FAULT;
    }
    if ( (fout_result = fopen(G_out_filename, "wt")) == NULL ) {
        sprintf(G_reportbuff[0], "Cannot open the output file.¥n");
        W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
        fclose(fin_y);
        return FAULT;
    }
    if ( G_u_file.name[0] != NULL ) {
        if ( (fin_u = fopen(G_u_file.name, "rb")) == NULL ) {
            sprintf(G_reportbuff[0], "Cannot open the inputted time series file.¥n");
            W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
            fclose(fin_y);
            fclose(fout_result);
            return FAULT;
        }
    }
    if ( G_pl_filename[0] != NULL ) {
        if ( (fout_pl = fopen(G_pl_filename, "wt")) == NULL ) {
            sprintf(G_reportbuff[0], "Cannot open the P/L containment file.¥n");
            W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
            fclose(fin_y);
            fclose(fout_result);
            if ( G_u_file.name[0] != NULL ) {
                fclose(fin_u);
            }
            return FAULT;
        }
    }
    if ( G_pole_filename[0] != NULL ) {
        for ( i = 0; i < G_rpemp.n; i++ ) {
            sprintf(polename_buff, "%s.%02d", G_pole_filename, (i+1));
            if ( (fout_pole[0] = fopen(polename_buff, "wb")) == NULL ) {
                sprintf(G_reportbuff[0], "Cannot open the pole containment file.¥n");
                W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
                fclose(fin_y);
                fclose(fout_result);
                if ( G_u_file.name[0] != NULL ) {
                    fclose(fin_u);
                }
                return FAULT;
```

```
        }
    }
}

/* Binary data skip */
fseek(fin_y, (G_y_file.skip_number * G_y_file.byte), SEEK_SET);
if ( G_u_file.name[0] != NULL) {
    fseek(fin_u, (G_y_file.skip_number * G_u_file.byte), SEEK_SET);
}


/* 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);


/* Start of Identification */
nnn = G_rpemp.n * 3;
G_rpemp.lambda = G_subp.lambda_init;
G_rpemp.K      = 0.0;

for ( i = 1; i <= G_y_file.identification_number; i++ ) {
    G_rpemp.y = Read_Data(fin_y, G_y_file.byte);
    if ( G_u_file.name[0] != NULL) {
        G_rpemp.u = Read_Data(fin_u, G_u_file.byte);
    } else {
        G_rpemp.u = 0.0;
    }
    Rpem_Routine();

    if ( G_pole_filename[0] != NULL ) {
        Output_Pole(fout_pole);
    }

    if ( i % G_interval_number == 0 ) {
        Output_Result(fout_result, i);

        if (G_pl_filename[0] = NULL ) {
            Output_PL(fout_pl, i, G_subp.activation);
        }
```

```
        }

        if ( G_subp.activation != OFF ) {
            if ( i % G_subp.activation_interval == 0 ) {
                switch ( G_subp.activation ) {
                    case PMA :
                        for ( j = 1; j <= nnn; j++ ) {
                            G_rpemp.UDL[j][j] *= G_subp.activation_power;
                        }
                        break;
                    case PMDA :
                        for ( j = 1; j <= nnn; j++ ) {
                            G_rpemp.P[j][j] *= G_subp.activation_power;
                        }
                        break;
                }
            }
        }

        G_rpemp.lambda = G_subp.lambda_mu * G_rpemp.lambda +1.0 - G_subp.lambda_mu;
        G_rpemp.K      = G_subp.contraction_mu * G_rpemp.K +1.0 - G_subp.contraction_mu;
    }
    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);

    fclose(fin_y);
    fclose(fout_result);
    if ( G_u_file.name[0] != NULL ) {
        fclose(fin_u);
    }
    if (G_pl_filename[0] != NULL ) {
        fclose(fout_pl);
    }
    if ( G_pole_filename[0] != NULL ) {
        for ( i = 0; i < G_rpemp.n; i++ ) {
            fclose(fout_pole[i]);
        }
    }
```

```
    return SUCCESS;
}


/* ---------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Starting_Check(void)                                 */
/*  Prototype   : PRT_armax1.h                                             */
/*  Description : Logical check of various parameter.                      */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h] */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h] */
/*  Variable    : none                                                     */
/*  Revision    : 2000/01/01                                               */
/* ---------+---------+---------+---------+---------+---------+---------- */
int  Starting_Check(void)
{
    if ( G_y_file.identification_number > G_y_file.useable_number ) {
        sprintf(G_reportbuff[0],
                "Number to identify exceed number of useable data.");
        goto ERROR_RECOVERY;
    }


    if ( G_interval_number > G_y_file.identification_number ) {
        sprintf(G_reportbuff[0],
                "Interval of output exceeds number to identify.");
        goto ERROR_RECOVERY;
    }

    return SUCCESS;

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


/* ---------+---------+---------+---------+---------+---------+---------- */
/*  Format      : int Parameter_Init(void)                                 */
/*  Prototype   : PRT_armax1.h                                             */
```

```
/*  Description : Initial parameter definition by reading parameter file.      */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]    */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]    */
/*  Variable    : none                                                          */
/*  Revision    : 2000/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
int  Parameter_Init(void)
{
    FILE            *fin_init;
    char            tmp_buff1[256], tmp_buff2[256];
    int             i, j, check, loop;
    double          dl_dummy;


    /* File open */
    if ( (fin_init = fopen( G_init_file.name, "rt" )) == NULL ) {
        sprintf(G_reportbuff[0], "Cannot open the initial parameter file.¥n");
        W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
        return FAULT;
    }

    /* Initialization */
    loop = G_rpemp.n * 3;


    /* Parameter part reading */
    check = 1;
    while ( fscanf(fin_init, "%s", tmp_buff1) != EOF ) {
        if ( (check = strcmp(tmp_buff1, "PARAMETER")) == 0 ) {
            break;
        }
    }
    if (check != 0) {
        goto ERROR_RECOVERY;
    }
    for (i = 1; i <= loop; i++) {
        fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
        dl_dummy = atof(tmp_buff2);
        if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
            sprintf(G_reportbuff[0], "Illegal value as initial parameter.");
            W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
```

```
            return FAULT;
        }
        G_rpemp.theta[i] = dl_dummy;
    }



    /* P_matrix or UDL matrix part reading */
    check = 1;
    while ( fscanf(fin_init, "%s", tmp_buff1) != EOF ) {
        if ( (check = strcmp(tmp_buff1, "P_MATRIX_or_UDL_MATRIX")) == 0 ) {
            break;
        }
    }
    if (check != 0) {
        goto ERROR_RECOVERY;
    }
    for (i = 1; i <= loop; i++) {
        for (j = 1; j <= loop; j++) {
            fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
            dl_dummy = atof(tmp_buff2);
            if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
                sprintf(G_reportbuff[0], "Illegal value as P Matrix.");
                W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
                return FAULT;
            }
            G_rpemp.UDL[i][j] = G_rpemp.P[i][j] = dl_dummy;
        }
    }


    /* L_vector part reading */
    check = 1;
    while ( fscanf(fin_init, "%s", tmp_buff1) != EOF ) {
        if ( (check = strcmp(tmp_buff1, "L_VECTOR")) == 0 ) {
            break;
        }
    }
    if (check != 0) {
        goto ERROR_RECOVERY;
    }
```

```
    for (i = 1; i <= loop; i++) {
        fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
        dl_dummy = atof(tmp_buff2);
        if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
            sprintf(G_reportbuff[0], "Illegal value as initial Kalman Gain.");
            W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
            return FAULT;
        }
        G_rpemp.L[i] = dl_dummy;
    }
    fclose(fin_init);
    return SUCCESS;

ERROR_RECOVERY:
    sprintf(G_reportbuff[0], "Iintial parameter file dose not obey defined format.");
    W_Writing(CRT_DISK, G_reportbuff, 1, G_fout_report);
    fclose(fin_init);
    return FAULT;
}



/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : int Parameter_Std_Init(void)                                  */
/*  Prototype   : PRT_armax1.h                                                  */
/*  Description : Standard initial parameter definition.                        */
/*  Return Value: none                                                          */
/*  Variable    : none                                                          */
/*  Revision    : 2000/01/01                                                    */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void  Parameter_Std_Init(void)
{
    int     i, j, loop;


    loop = G_rpemp.n * 3;

    for (i = 1; i <= loop; i++) {
        G_rpemp.theta[i] = 0.0;
        G_rpemp.L[i]     = 0.0;
```

```
        G_rpemp.phi[i]   = 0.0;
        G_rpemp.psi[i]   = 0.0;
        for (j = 1; j <= loop; j++) {
            G_rpemp.UDL[i][j] = 0.0;
            G_rpemp.P[i][j]   = 0.0;
        }
        G_rpemp.P[i][i]   = G_rpemp.pmatrix0;
        G_rpemp.UDL[i][i] = G_rpemp.pmatrix0;
    }

}



/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : double Read_Data(FILE *fin, int size)                            */
/*  Prototype   : PRT_armax1.h                                                     */
/*  Description : 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    : 2000/01/01                                                       */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
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      : int Method_Judgement(void)                                     */
/*  Prototype   : PRT_armax1.h                                                   */
/*  Description : Algorithm type Judgement.                                      */
/*  Return Value: PLR    [defined in DFN_armax1.h]                               */
/*                ELS    [defined in DFN_armax1.h]                               */
/*                RML    [defined in DFN_armax1.h]                               */
/*                RPEM   [defined in DFN_armax1.h]                               */
/*  Variable    : none                                                           */
/*  Revision    : 2000/01/01                                                     */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
int  Method_Judgement(void)
{
    int     method;

    if (G_subp.contraction_mu == 1.0) {
        method = PLR;
        if (G_subp.lambda_init == 1.0) {
            method = ELS;
        }
    } else {
        if (G_subp.contraction_mu == 1.0 && G_subp.lambda_init == 1.0) {
            method = RML;
        } else {
            method = RPEM;
        }
    }

    return method;
}
```

4.3 Input data Read Program

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  ARMAX Parameter Calculation Program by RPE Method.                        _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7--------- */


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


/* ----------+----------+----------+----------+----------+----------+---------- */
/* private include defines.                                                     */
/* ----------+----------+----------+----------+----------+----------+---------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ----------+----------+----------+----------+----------+----------+---------- */
/*  Format      : int ReadFileCall(char *filename)                              */
/*  Prototype   : PRT_armax1.h                                                  */
/*  Description : Reading job file and setting parameter.                       */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]     */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]     */
/*  Variable    : *filename; (i  ) pointer to filenmae buffer.                  */
/*  Revision    : 2000/01/01                                                    */
/* ----------+----------+----------+----------+----------+----------+---------- */
int  ReadFileCall(char *filename)
{
   FILE        *fin_setupfile, *fin_test, *fout_test;
   char        input_buff[256], tmp_buff1[256], tmp_buff2[256];
```

```
int         i;
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_rpemp,          0x00, sizeof(S_RPEM_PARAMETER));
memset(&G_u_file,         0x00, sizeof(S_BINARY_FILE));
memset(&G_y_file,         0x00, sizeof(S_BINARY_FILE));
memset(&G_init_file,      0x00, sizeof(S_INIT_FILE));
memset(G_out_filename,    0x00, 256);
memset(G_pole_filename,   0x00, 256);
memset(G_pl_filename,     0x00, 256);

/* Reading parameter in setup file */
for ( i = 0; i < 20; 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: /* ID = 0: Reading of identification order */
            G_rpemp.n = atoi(input_buff);
            if ( G_rpemp.n < 0 || G_rpemp.n > 9 ) {
                sprintf( G_reportbuff[0], "Illegal value as identification order.¥n" );
                goto ERROR_RECOVERY;
            }
            break;

        case 1: /* ID = 1: Reading of record length of binary time series */
            G_u_file.byte = G_y_file.byte = atoi(input_buff);
```

```
        if ( G_u_file.byte != 2 && G_u_file.byte != 8 ) {
            sprintf(G_reportbuff[0],
                        "Illegal value as record length of time series data.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 2: /* ID = 2: Reading of U filename */
        if ( strcmp(input_buff, "NULL") == 0 ) {
            useable_number = 2147483647;
        } else {
            if ( (fin_test = fopen(input_buff, "rb") ) == NULL ) {
                sprintf(G_reportbuff[0],
                    "Cannot open the time series data file inputted to system.¥n");
                goto ERROR_RECOVERY;
            } else {
                strcpy(G_u_file.name, input_buff);
                G_u_file.number = Binary_File_Check(fin_test, G_u_file.byte);
                fclose(fin_test);
                if ( G_u_file.number == -1 ) {
                    sprintf(G_reportbuff[0],
                        "Probably, [ %s ] is not binary data file of %d byte",
                        input_buff, G_u_file.byte);
                    goto ERROR_RECOVERY;
                }
                useable_number = G_u_file.number;
            }
        }
        break;

    case 3: /* ID = 3: Reading of Y filename */
        if ( (fin_test = fopen(input_buff, "rb")) == NULL ) {
            sprintf(G_reportbuff[0],
                "Cannot open the time series data file outputted to system.¥n" );
            goto ERROR_RECOVERY;
        } else {
            strcpy(G_y_file.name, input_buff);
            G_y_file.number = Binary_File_Check(fin_test, G_y_file.byte);
            fclose(fin_test);
            if ( G_y_file.number == -1 ) {
                sprintf(G_reportbuff[0],
```

```
                "Probably, [ %s ] is not binary data file of %d byte¥n",
                input_buff, G_y_file.byte);
            goto ERROR_RECOVERY;
        }
        if ( G_y_file.number < useable_number ) {
            useable_number = G_y_file.number;
        }
    }
    break;

case 4: /* ID = 4: Reading of initial parameter setting filename */
    if ( strcmp(input_buff, "NULL") != 0 ) {
        if ( (fin_test = fopen(input_buff, "rt")) == NULL ) {
            sprintf(G_reportbuff[0], "Canont open initial parameter file.¥n");
            goto ERROR_RECOVERY;
        }
        strcpy(G_init_file.name, input_buff);
        fscanf(fin_test, "%s %s", tmp_buff1, tmp_buff2);
        fclose(fin_test);
        if ( strcmp(tmp_buff1, "INITIAL_PARAMETER_FILE") != 0 ) {
            sprintf(G_reportbuff[0],
                "First declaration of the initial parameter file is unsuitable.¥n");
            goto ERROR_RECOVERY;
        }
        if ( ((int)(*tmp_buff2) - 48) != G_rpemp.n ) {
            sprintf(G_reportbuff[0],
                "There is discrepancy as to identification order.¥n");
            goto ERROR_RECOVERY;
        }
        G_init_file.order = (int)(*tmp_buff2) - 48;
    }
    break;

case 5: /* Reading of number of skipping data */
    G_y_file.skip_number = atol(input_buff);
    if ( G_y_file.skip_number < 0 || G_y_file.skip_number > useable_number ) {
        sprintf(G_reportbuff[0], "Illegal value as number of skipping data.¥n");
        goto ERROR_RECOVERY;
    }
    useable_number -= G_y_file.skip_number;
    break;
```

```
case 6: /* Reading of number of identification */
    G_y_file.identification_number = atol(input_buff);
    if ( G_y_file.identification_number <= 0 ||
                G_y_file.identification_number > useable_number ) {
        sprintf(G_reportbuff[0], "Illegal value as number of identification.¥n");
        goto ERROR_RECOVERY;
    }
    break;

case 7: /* Reading of forgetting factor mu */
    G_subp.lambda_mu = atof(input_buff);
    if ( G_subp.lambda_mu <= 0.0 || G_subp.lambda_mu > 1.0 ) {
        sprintf(G_reportbuff[0],
                "Illegal value as constant part of forgetting factor.¥n");
        goto ERROR_RECOVERY;
    }
    break;

case 8: /* Reading of forgetting factor init */
    G_subp.lambda_init = atof(input_buff);
    if ( G_subp.lambda_init <= 0.0 || G_subp.lambda_init > 1.0 ) {
        sprintf(G_reportbuff[0],
                "Illegal value as initial forgetting factor.¥n");
        goto ERROR_RECOVERY;
    }
    break;

case 9: /* Reading of contraction factor mu */
    G_subp.contraction_mu = atof(input_buff);
    if (G_subp.contraction_mu < 0.0 || G_subp.contraction_mu > 1.0 ) {
        sprintf(G_reportbuff[0], "Illegal value as contraction factor mu.¥n");
        goto ERROR_RECOVERY;
    }
    break;

case 10: /* Reading of prevention value */
    G_rpemp.c = atof(input_buff);
    if (G_rpemp.c <= 0.0 ) {
        sprintf(G_reportbuff[0],
                "Illegal value as constant for divergent prevention.¥n");
```

```
            goto ERROR_RECOVERY;
        }
        break;

    case 11: /* Reading of stable check switch */
        if ( strcmp(input_buff, "ON") == 0 ) {
            G_rpemp.stable = ON;
        } else if ( strcmp(input_buff, "OFF") == 0 ) {
            G_rpemp.stable = OFF;
        } else {
            sprintf(G_reportbuff[0], "Illegal switch as stable check.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 12: /* Reading of initialization value of P matrix */
        G_rpemp.pmatrix0 = atof(input_buff);
        if ( G_rpemp.pmatrix0 <= 0.0 ) {
            sprintf(G_reportbuff[0],
                    "Illegal value as diagonal elements of initial P matrix.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 13: /* Reading of activation method switch */
        if ( strcmp(input_buff, "PMDA") == 0 ) {
            G_subp.activation = PMDA;
        } else if ( strcmp( input_buff, "PMA" ) == 0 ) {
            G_subp.activation = PMA;
        } else if ( strcmp( input_buff, "OFF" ) == 0 ) {
            G_subp.activation = OFF;
        } else {
            sprintf(G_reportbuff[0],
                    "Illegal switch as selection of activation method.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 14: /* Reading of activation interval */
        G_subp.activation_interval = atol(input_buff);
        if ( G_subp.activation_interval <= 0 ||
```

```
                G_subp.activation_interval > useable_number ) {
            sprintf(G_reportbuff[0],
                "Illegal value as number of activation interval.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 15: /* Reading of activation power */
        G_subp.activation_power = atof(input_buff);
        if (G_subp.activation_power < 1.0 ) {
            sprintf(G_reportbuff[0], "Illegal value as activation power.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 16: /* Reading of 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;

    case 17: /* Reading of output interval */
        G_interval_number = atol(input_buff);
        if ( G_interval_number <= 0 || G_interval_number > useable_number ) {
            sprintf(G_reportbuff[0], "Illegal value as number of interval.¥n");
            goto ERROR_RECOVERY;
        }
        break;

    case 18: /* Reading of output filename of binary pole */
        if ( strcmp(input_buff, "NULL") != 0 ) {
            if ( (fout_test = fopen(input_buff, "wb")) == NULL ) {
                sprintf(G_reportbuff[0], "Unsuitable Output File Name.¥n");
                goto ERROR_RECOVERY;
            }
            fclose(fout_test);
            strcpy(G_pole_filename, input_buff);
        }
```

```
            break;

        case 19: /* Reading fo output filename of P/L */
            if ( strcmp(input_buff, "NULL") != 0 ) {
                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_pl_filename, input_buff);
            }
            break;
        }
    }
    fclose(fin_setupfile);
    G_y_file.useable_number = useable_number;

    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_armax1.h                                                   */
/*  Description : 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    : 2000/01/01                                                     */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
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_armax1.h                                               */
/*  Description : Standard reading function of setup file.                   */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]  */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.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    : 2000/01/01                                                 */
/* ---------+---------+---------+---------+---------+---------+---------- */
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;
}
```

4.4 ARMAX by RPE Method

```
/*--------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  ARMAX Parameter Calculation Program by RPE Method.                      _/
_/                                                                          _/
_/                                                                          _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7--------- */


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


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* private include defines.                                                       */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : void Rpem_routine(void)                                         */
/*  Prototype   : PRT_armax1.h                                                    */
/*  Description : Calculation by RPEM Algorithm.                                  */
/*  Return Value: none                                                            */
/*  Variable    : none                                                            */
/*  Revision    : 2000/01/01                                                      */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
void  Rpem_Routine(void)
{
    int     i, j, k, nnn, ex_part_s, ma_part_s;
    double dl_dummy, amy, ma_part[28], y1_hat, u1_hat, e1_hat, c_i;
```

```
nnn = G_rpemp.n * 3;

ex_part_s = G_rpemp.n + 1;
ma_part_s = 2 * G_rpemp.n + 1;


/* Calculation of Prediction Error */
G_rpemp.prediction_error = G_rpemp.y;
for (i = 1; i <= nnn; i++)
    G_rpemp.prediction_error -= G_rpemp.phi[i] * G_rpemp.theta[i];



/* Stablility Check and Decision of AMY */
amy = 1.0;
if ( G_rpemp.stable == ON ) {
    while (1) {
        for (i = 0; i < G_rpemp.n; i++) {
            j = ma_part_s + i;
            ma_part[i+2] = G_rpemp.theta[j] + G_rpemp.L[j] *
                                    G_rpemp.prediction_error * amy;
        }
        ma_part[1] = 1.0;
        if ( Stable_Check(ma_part, G_rpemp.n) == SUCCESS )
            break;
        amy /= 2.0;
    }
}



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



/* Calculation of Current Residual */
G_rpemp.residual = G_rpemp.y;
for (i = 1; i <= nnn; i++)
```

```
    G_rpemp.residual -= G_rpemp.phi[i] * G_rpemp.theta[i];


/* Whitning of Residual */
y1_hat = G_rpemp.y;
u1_hat = G_rpemp.u;
e1_hat = G_rpemp.residual;

for (i = 0; i < G_rpemp.n; i++) {
    d1_dummy = (double)i + 1.0;
    c_i = G_rpemp.theta[ma_part_s + i] * pow(G_rpemp.K, d1_dummy);
    y1_hat += c_i * G_rpemp.psi[1          + i];
    u1_hat -= c_i * G_rpemp.psi[ex_part_s + i];
    e1_hat -= c_i * G_rpemp.psi[ma_part_s + i];
}


/* Update of Time Series Containment Vector phi[] psi[] */
for (i = 2; i <= G_rpemp.n; i++)
    for (j = 1; j <= 3; j++) {
        k = j * G_rpemp.n + 2 - i;
        G_rpemp.phi[k] = G_rpemp.phi[k-1];
        G_rpemp.psi[k] = G_rpemp.psi[k-1];
    }

G_rpemp.phi[1]          = -G_rpemp.y;
G_rpemp.phi[ex_part_s] = G_rpemp.u;
G_rpemp.phi[ma_part_s] = G_rpemp.residual;

G_rpemp.psi[1]          = -y1_hat;
G_rpemp.psi[ex_part_s] = u1_hat;
G_rpemp.psi[ma_part_s] = e1_hat;


/* Update of P Matrix or UDL Matrix and Gain Vector L*/
switch ( G_subp.activation ) {
    case OFF :
    case PMA :
```

```
            UDL_Factorization(&G_rpemp);
            break;
        case PMDA :
            P_Matrix_Calc(&G_rpemp);
            break;
    }
}


/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : int Stable_Check(double *work, int n)                          */
/*  Prototype   : PRT_armax1.h                                                   */
/*  Description : Stable Check Algorithm.                                        */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]      */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]      */
/*  Variable    : *work; (  o) pointer to calculated data.                       */
/*                n    ; (i  ) identification order.                             */
/*  Revision    : 2000/01/01                                                     */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
int  Stable_Check(double *work, int n)
{

    int    i, k, n1, nk1, nk;
    double al;


    n1 = n + 1;
    for ( i = 1; i <= n1; i++ ) {
        work[n1+i] = 0.0;
    }

    for ( k = 0; k < n; k++ ) {
        nk1 = n1 - k;
        for ( i = 1; i <= nk1; i++ ) {
            work[n1+i] = work[nk1-i+1];
        }
        if ( work[n1+nk1] == 0.0 ) {
            return FAULT;
        }
        al = work[nk1] / work[n1+nk1];
```

```
        if (fabs(al) >= 1.0) {
            return FAULT;
        }
        nk = n - k;
        for ( i = 1; i <= nk; i++ ) {
            work[i] -= al*work[n1+i];
        }
    }
    return SUCCESS;

}



/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Format      : int UDL_Factorization(S_RPEM_PARAMETER *rpemp)                    */
/*  Prototype   : PRT_armax1.h                                                      */
/*  Description : UDL matrix factorization Algorithm.                              */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]         */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]         */
/*  Variable    : *rpemp; (i/o) pointer to RPE parameter item structure.            */
/*  Revision    : 2000/01/01                                                        */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
int  UDL_Factorization(S_RPEM_PARAMETER  *rpemp)
{
    int     i, j, k, l, nnn;
    double f[25], g[25], previous_beta, beta_inverse, beta, previous_g,
           previous_udl, dl_dummy;


    nnn = rpemp->n * 3;


    for (i = 2; i <= nnn; i++) {
        j = nnn + 2 - i;
        f[j] = rpemp->psi[j];
        k = j - 1;
        for (l = 1; l <= k; l++)
            f[j] += rpemp->UDL[l][j] * rpemp->psi[l];
        g[j] = rpemp->UDL[j][j] * f[j];
    }
```

```
    g[1] = rpemp->UDL[1][1] * rpemp->psi[1];
    f[1] = rpemp->psi[1];

    previous_beta = rpemp->lambda + f[1]*g[1];
    beta_inverse = 0.0;
    if (previous_beta > 0.0)
        beta_inverse = 1.0 / previous_beta;
    if (g[1] != 0.0)
        rpemp->UDL[1][1] = beta_inverse * rpemp->UDL[1][1];

    for (i = 2; i <= nnn; i++) {
        beta = previous_beta;
        previous_g = g[i];
        previous_beta += previous_g * f[i];
        if (previous_beta == 0.0)
            continue;
        d1_dummy = -f[i] * beta_inverse;
        j = i - 1;
        for (k = 1; k <= j; k++) {
            previous_udl = rpemp->UDL[k][i];
            rpemp->UDL[k][i] += d1_dummy * g[k];
            g[k] += previous_udl * previous_g;
        }

        beta_inverse = 1.0 / previous_beta;

        rpemp->UDL[i][i] *= beta * beta_inverse / rpemp->lambda;
        rpemp->UDL[i][i]  = Min_Calc(rpemp->UDL[i][i], rpemp->c);
    }

    for (i = 1; i <= nnn; i++)
        rpemp->L[i] = g[i] / previous_beta;


    return SUCCESS;
}


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

```
/*  Format      : int P_Matrix_Calc(S_RPEM_PARAMETER *rpemp)                  */
/*  Prototype   : PRT_armax1.h                                                */
/*  Description : P matrix calcuration Algorithm.                             */
/*  Return Value: normal termination   -> SUCCSESS[defined in DFN_armax1.h]   */
/*                abnormal termination -> FAULT   [defined in DFN_armax1.h]   */
/*  Variable    : *rpemp; (i/o) pointer to RPE parameter item structure.      */
/*  Revision    : 2000/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------ */
int  P_Matrix_Calc(S_RPEM_PARAMETER  *rpemp)
{
    int     i, j, nnn;
    double p_psi[25], psi_p[25], p_psi_psi_p[25][25], bunbo;


    nnn = rpemp->n * 3;


    for (i = 1; i <= nnn; i++) {
        rpemp->L[i] = 0.0;
        p_psi[i] = 0.0;
        psi_p[i] = 0.0;
    }

    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi[i] += rpemp->P[i][j] * rpemp->psi[j];


    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            psi_p[i] += rpemp->psi[j] * rpemp->P[j][i];


    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi_psi_p[i][j] = p_psi[i] * psi_p[j];


    bunbo = rpemp->lambda;
    for (i = 1; i <= nnn; i++)
        bunbo += rpemp->psi[i] * p_psi[i];
```

```
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi_psi_p[i][j] /= bunbo;


    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->P[i][j] -= p_psi_psi_p[i][j];


    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->P[i][j] /= rpemp->lambda;


    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->L[i] += rpemp->P[i][j] * rpemp->psi[j];


    return SUCCESS;
}
```

4.5 Output Program

```
/* ---------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                                _/
_/  ARMAX Parameter Calculation Program by RPE Method.                            _/
_/                                                                                _/
_/                                                                                _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7--------- */


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


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* private include defines.                                                         */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Formula     : void Output_Result(FILE *fout_result, long loop)                  */
/*  Prototype   : PRT_armax1.h                                                      */
/*  Description : Output calculated results.                                        */
/*  Return Value: none                                                              */
/*  Arguments   : *fout_result; (i  ) file pointer of output file.                  */
/*                loop        ; (i  ) loop number.                                  */
/*  Revision    : 2000/01/01                                                        */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
void  Output_Result(FILE *fout_result, long loop)
{
    static char        *converge_string[] = {"CANNOTCALC", "CONVERGE", "NOCONVERGE"};
```

```
    S_COMPLEX          pole[10];
    int                i, start_point, converge_check;


    converge_check = Pole_Calculation(G_rpemp.n, G_rpemp.theta, pole);
    printf("%-71d  POLE:%s¥n", loop, converge_string[converge_check+1]);
    fprintf(fout_result, "%-71d  POLE:%s¥n", loop, converge_string[converge_check+1]);


    /* Writing of AR Parameters */
    Parameter_Writing(fout_result, G_rpemp.n, 1, G_rpemp.theta);


    /* Writing of Exgeneous Parameters */
    if ( G_u_file.name[0] != NULL) {
        start_point = G_rpemp.n + 1;
        Parameter_Writing(fout_result, G_rpemp.n, start_point, G_rpemp.theta);
    }

    /* Writing of MA Parameters */
    start_point = G_rpemp.n * 2 + 1;
    Parameter_Writing(fout_result, G_rpemp.n, start_point, G_rpemp.theta);


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

    printf("¥n");
    fprintf(fout_result, "¥n¥n");

}


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Formula     : void Parameter_Writing(FILE *fout_result,                        */
/*                                       int number,                               */
/*                                       int start_point,                          */
```

```
/*                                          double theta[])                    */
/*  Prototype   : PRT_armax1.h                                                 */
/*  Description : Output calculated results. <- sub process.                   */
/*  Return Value: none                                                         */
/*  Arguments   : *fout_result; (i  ) file pointer to output file.             */
/*                number      ; (i  ) number of output parameter.              */
/*                start_point ; (i  ) start point for array position           */
/*                hteta[]     ; (i  ) array of ARMAX parameter.                */
/*  Revision    : 2000/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
void  Parameter_Writing(FILE *fout_result, int number, int start_point, double theta[])
{
    int            i, j;


    if (number > 4) {
        for (i = 0; i <= 3; i++) {
            j = i + start_point;
            printf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }
        printf("¥n");
        fprintf(fout_result, "¥n");

        for (i = 4; i < number; i++) {
            j = i + start_point;
            printf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }
    } else
        for (i = 0; i < number; i++) {
            j = i + start_point;
            printf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }

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

```
/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Formula     : void Output_Pole(FILE *fout_pole[])                          */
/*  Prototype   : PRT_armax1.h                                                 */
/*  Description : Output calculated pole by binary data.                       */
/*  Return Value: none                                                         */
/*  Arguments   : *fout_pole; (i  ) file pointer array of output file.         */
/*  Revision    : 2000/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  Output_Pole(FILE *fout_pole[])
{

    S_COMPLEX          pole[10];
    int                i, j;

    Pole_Calculation(G_rpemp.n, G_rpemp.theta, pole);

    for (i = 0; i < G_rpemp.n; i++) {
        j = i + 1;
        fwrite(&pole[i].real, 8, 1, fout_pole[j]);
    }
}


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Formula     : void Output_PL(FILE *fout_pl, long loop, int method)         */
/*  Prototype   : PRT_armax1.h                                                 */
/*  Description : Output calculated P/L.                                       */
/*  Return Value: none                                                         */
/*  Arguments   : *fout_pl; (i  ) file pointer of output file.                 */
/*                loop    ; (i  ) loop number.                                 */
/*                method  ; (i  ) activation method type.                      */
/*  Revision    : 2000/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
void  Output_PL(FILE *fout_pl, long loop, int method)
{

    int                    i, j, nnn;
```

```
nnn = G_rpemp.n * 3;

fprintf(fout_pl,
        "Value of Y_Data at %ld Data            = %13.6e¥n",
                                                  loop, G_rpemp.y);
fprintf(fout_pl,
        "Value of Prediction Error at %ld Data = %13.6e¥n",
                                  loop, G_rpemp.prediction_error);

fprintf(fout_pl,
        "Value of The Residual at %ld Data      = %13.6e¥n¥n",
                                       loop, G_rpemp.residual);

fprintf(fout_pl, "Elements of P Matrix at %ld Data¥n", loop);

switch (method) {
    case OFF :
    case PMA :
        Udl_to_P(nnn, G_rpemp.UDL, G_rpemp.P);
        for (i = 1; i <= nnn; i++)
            for (j = 1; j <= nnn; j++)
                fprintf(fout_pl, "P[%d][%d] = % 13.6e      UDL[%d][%d] = % 13.6e¥n",
                        i, j, G_rpemp.P[i][j], i, j, G_rpemp.UDL[i][j]);
        break;
    case PMDA :
        for (i = 1; i <= nnn; i++)
            for (j = 1; j <= nnn; j++)
                fprintf(fout_pl, "P[%d][%d] = % 13.6e¥n",
                                       i, j, G_rpemp.P[i][j]);
        break;
}

fprintf(fout_pl, "Elements of L Vector at %ld Data¥n", loop);

for (i = 1; i <= nnn; i++)
```

```
        fprintf(fout_pl, "L[%d]      = % 13.6e¥n", i, G_rpemp.L[i]);

    fprintf(fout_pl, "¥n¥n");
}
```

4.6 Pole Calculation Program

```
/* ---------1---------2---------3---------4---------5---------6---------7---------8
/*                                                                                  */
/*  ARMAX Parameter Calculation Program by RPE Method.                              */
/*                                                                                  */
/*                                                                                  */
/*                                                                                  */
---------1---------2---------3---------4---------5---------6---------7--------- */


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


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* private include defines.                                                         */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Formula     : int Pole_Calculation(int order,                                   */
/*                                     double theta[],                              */
/*                                     S_COMPLEX pole[])                            */
/*  Prototype   : PRT_armax1.h                                                      */
/*  Description : System pole calculation.                                          */
/*  Return Value: normal termination -> CONVERGE[defined in DFN_armax1.h]           */
/*                cannot calculation -> CANNOTCALC[defined in DFN_armax1.h]         */
/*                noconverged result -> NOCONVERGE[defined in DFN_armax1.h]         */
/*  Arguments   : order  ; (i  ) identification order.                              */
/*                theta[]; (i  ) array of elements of AR paramter.                  */
/*                pole[] ; (  o) array of calculated value.                         */
```

```
/*  Revision     : 2000/01/01                                                    */
/* ---------+---------+---------+---------+---------+---------+---------+------- */
int  Pole_Calculation(int order, double theta[], S_COMPLEX pole[])
{
    int     i, converge_check;
    double  coefficient[10];


    coefficient[0] = 1.0;

    for ( i = 1; i <= order; i++ ) {
        coefficient[i] = theta[i];
    }

    for ( i = 0; i <= order; i++ ) {
        pole[i].image = 0.0;
    }

    converge_check = Bairstow(coefficient, order, pole);

    if ( converge_check == MAXLOOP ) {
        return NOCONVERGE;
    }
    if ( converge_check == CANNOTCALC ) {
        return CANNOTCALC;
    }

    return CONVERGE;
}
```

4.7 Bairstow Method for Roots Finding

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
/_                                                                            /_
/_  ARMAX Parameter Calculation Program by RPE Method.                        /_
/_                                                                            /_
/_                                                                            /_
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7------- */


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


/* ------+---------+---------+---------+---------+---------+---------+------- */
/* private include defines.                                                   */
/* ------+---------+---------+---------+---------+---------+---------+------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ------+---------+---------+---------+---------+---------+---------+------- */
/*  Function    : double Min_Calc(double data1, double data2)                 */
/*  Prototype   : PRT_armax1.h                                                */
/*  Description : Return the lesser of the two values.                        */
/*  Return Value: the lesser of the two values.                               */
/*  Arguments   : data1; (i  ) the first double.                              */
/*                data2; (i  ) the second double.                             */
/*  Revision    : 2000/01/01                                                  */
/* ------+---------+---------+---------+---------+---------+---------+------- */
double  Min_Calc(double data1, double data2)
{
```

```
        if ( data1 < data2 ) {
                return data1;
        }

        return data2;
}


/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Function    : int Bairstow(double ta[], int n, S_COMPLEX x[])              */
/*  Prototype   : PRT_armax1.h                                                 */
/*  Description : Solving the equation by bairstow method.                     */
/*  Return Value: normal termination   -> number of loop                       */
/*                abnormal termination -> CANNOTCALC [defined in DFN_armax1.h]*/
/*                                        this value is applyed when the cal- */
/*                                        culation is not converged.          */
/*  Arguments   : ta[]; (i  ) array of elements for equation.                  */
/*                n   ; (i  ) order.                                           */
/*                x[] ; (  o) array of solution.                               */
/*  Revision    : 2000/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
int     Bairstow(double ta[], int n, S_COMPLEX x[])
{
        double  a[10], b[10], c[10];
        double  p0, q0, p, q, D, delta_p, delta_q;
        double  tmp[3];
        int     i, j, k, l, max_loop = 0, count = 0;

        for (i = 1; i <= n; i++)
                a[i] = ta[i] / ta[0];

        a[0] = 1.0;
        tmp[0] = 1.0;
        while (n > 2) {
        p = p0 = 1.0;
                q = q0 = 1.0;
                for (l = 1;l < MAXLOOP; l++) {
                        b[0] = 1.0;
                        c[0] = 1.0;
```

```
                b[1] = a[1] - p;
                c[1] = b[1] - p;
                for(k = 2; k <= n; k++) {
                        b[k] = a[k] - p*b[k-1] - q*b[k-2];
                        c[k] = b[k] - p*c[k-1] - q*c[k-2];
                }

                D = c[n-2]*c[n-2] - c[n-3]*(c[n-1] - b[n-1]);

                if (D == 0.0)
                        return CANNOTCALC;

                delta_p = ( b[n-1]*c[n-2] - b[n]*c[n-3] ) / D;
                delta_q = ( b[n]*c[n-2] - b[n-1]*(c[n-1] - b[n-1]) ) / D;

                p = p0 + delta_p;
                q = q0 + delta_q;

                if( (fabs(delta_p) < EPS) && ( fabs(delta_q) < EPS) )
                        break ;
                p0 = p;
                q0 = q;
}
        if (l > max_loop)
                max_loop = l;
tmp[1] = p ;    tmp[2] = q ;
Equation_2(tmp, &x[count]) ;
n -= 2;count += 2;
for(i = 0; i <= n; i++)
                a[i] = b[i];

}

if (n == 2) {
        tmp[1] = a[1];
        tmp[2] = a[2];
        Equation_2(tmp, &x[count]);
}

if ( n & 1 ) {
        x[count].real = -a[1];
```

```
                x[count].image = 0.0;
        }

        return max_loop;

}


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/*  Function     : void Equation_2(double a[], S_COMPLEX x[])                      */
/*  Prototype    : PRT_armax1.h                                                    */
/*  Description  : Solving the second order equation by the formula.               */
/*  Return Value: none                                                             */
/*  Arguments    : a[]; (i  ) array of elements for equation.                      */
/*                 x[]; (  o) array of solution.                                   */
/*  Revision     : 2000/01/01                                                      */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
void  Equation_2(double a[], S_COMPLEX x[])
{
        double  tmp, D;


        D = a[1]*a[1] - 4*a[0]*a[2] ;

        if(D >= 0.0) {
                tmp = -a[1] + sqrt(D);
                x[0].real = tmp / (2 * a[0]);
                tmp = -a[1] - sqrt(D);
                x[1].real = tmp / (2 * a[0]);
        }
        else {
                x[0].real = -a[1] / (2 * a[0]) ;
                x[0].image = sqrt(-D) / (2 * a[0]) ;
                x[1].real = x[0].real ;
                x[1].image = -x[0].image ;
        }

}
```

```
/* ------+---------+---------+---------+---------+---------+---------+--------- */
/*  Formula     : void Udl_to_P(int order, double udl[][25], double p[][25])  */
/*  Prototype   : PRT_armax1.h                                                 */
/*  Description : Converting form UDL matrix to P Matrix.                      */
/*  Return Value: none                                                         */
/*  Arguments   : order     ; (i  ) size of matrix.                            */
/*                udl[][25]; (i  ) second order array of udl matrix.           */
/*                p[][25]  ; (  o) second order array of p matrix.             */
/*  Revision    : 2000/01/01                                                   */
/* ------+---------+---------+---------+---------+---------+---------+--------- */
void  Udl_to_P(int order, double udl[][25], double p[][25])
{
        int     i, j, k, start_row;

        double  u[25][25], d[25][25], l[25][25], dummy[25][25];


        for (i = 0; i <= 24; i++)
                for (j = 0; j <= 24; j++) {
                        u[i][j]     = 0.0;
                        d[i][j]     = 0.0;
                        l[i][j]     = 0.0;
                        dummy[i][j] = 0.0;
                        p[i][j]     = 0.0;
                }


        for (i = 1; i < order; i++) {
                start_row = i + 1;
                for (j = start_row; j <= order; j++) {
                        u[i][j] = udl[i][j];
                        l[j][i] = udl[i][j];
                }
        }
        for (i = 1; i <= order; i++) {
                u[i][i] = 1.0;
                l[i][i] = 1.0;
        }

        for (i = 1; i <= order; i++)
```

```
            d[i][i] = udl[i][i];


        for (i = 1; i <= order; i++)
            for (j = 1; j <= order; j++)
                for (k = 1; k <= order; k++)
                    dummy[i][j] += u[i][k] * d[k][j];


        for (i = 1; i <= order; i++)
            for (j = 1; j <= order; j++)
                for (k = 1; k <= order; k++)
                    p[i][j] += dummy[i][k] * l[k][j];

}
```

4.8 Print Out Program

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
/////////////////////////////////////////////////////////////////////////////////
/                                                                               /
/  ARMAX Parameter Calculation Program by RPE Method.                           /
/                                                                               /
/                                                                               /
/////////////////////////////////////////////////////////////////////////////////
---------1---------2---------3---------4---------5---------6---------7-------- */


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


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/* private include defines.                                                    */
/* ------+---------+---------+---------+---------+---------+---------+-------- */
#include "DFN_armax1.h"
#include "VAR_armax1.h"
#include "PRT_armax1.h"


/* ------+---------+---------+---------+---------+---------+---------+-------- */
/*  Format      : void W_Writing(int type,                                     */
/*                               char **string,                                 */
/*                               int number                                     */
/*                               FILE *fout)                                    */
/*  Prototype   : PRT_armax1.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    : 2000/01/01                                                   */
```

```
/* ----+----+----+----+----+----+----+---- */
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_armax1.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     : 2000/01/01                                              */
/* ----+----+----+----+----+----+----+---- */
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;
}
```

4.9 Define Header File

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                          _/
_/  ARMAX Parameter Calculation Program by RPE 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

/* Activation method */
#define PMA         1
#define PMDA        2
```

```
/* Algorithm type */
#define RPEM        0
#define PLR         1
#define ELS         2
#define RML         3

/* 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 */
    long    useable_number;         /* Useable number in total data */
    long    identification_number; /* Number of identification data */
} S_BINARY_FILE;

typedef struct {
    char    name[256];               /* Fine name */
    int     order;                  /* Identification order */
} S_INIT_FILE;

typedef struct {
    double  theta[25];              /* ARMAX parameter */
    double  UDL[25][25];            /* UDL matrix */
    double  P[25][25];              /* Prediction error variance matrix */
```

```
    double L[25];                /* Gain vector */
    double phi[25];              /* WORK */
    double psi[25];              /* WORK */
    int    n;                    /* Identification order */
    double u;                    /* Time seriase of U (newest) */
    double y;                    /* Time seriase of Y (newest) */
    double lambda;               /* Forgetting factor */
    double K;                    /* Contraction factor */
    double c;                    /* Preventing divergency parameter */
    int    stable;               /* Switch of stable check */
    double prediction_error;     /* The prediction error */
    double residual;             /* The residual */
    double pmatrix0;             /* Initial value of P matrix */
} S_RPEM_PARAMETER;

typedef struct {
    double lambda_mu;            /* Forgetting factor constant mu */
    double lambda_init;          /* Initial value for forgetting factor */
    double contraction_mu;       /* Contraction factor constant mu */
    int    activation;           /* Switch of activation method */
    long   activation_interval;  /* Interval of activation */
    double activation_power;     /* Activation power */
} S_RPEM_SUB;

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;
```

4.10 Variable Define Header File

```
/* ————1————2————3————4————5————6————7————8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                             _/
_/  ARMAX Parameter Calculation Program by RPE Method.                          _/
_/                                                                             _/
_/                                                                             _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
————1————2————3————4————5————6————7———— */

/* ====================================================================== */
/* extern define.                                                         */
/* ====================================================================== */
#ifdef ARMAX1_MAIN
#define EXT
#else  #define EXT
   extern
#endif

/* ====================================================================== */
/* variable define.                                                     */
/* ====================================================================== */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* global.                                                                */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */

/* RPEM main parameter */
EXT S_RPEM_PARAMETER  G_rpemp;

/* RPEM sub parameter */
EXT S_RPEM_SUB        G_subp;

/* input binary file definiation */
EXT S_BINARY_FILE     G_u_file;
EXT S_BINARY_FILE     G_y_file;

/* initial parameter file definition */
EXT S_INIT_FILE       G_init_file;
```

```
/* output file name */
EXT char            G_out_filename[256];
EXT char            G_pole_filename[256];
EXT char            G_pl_filename[256];

/* output interval */
EXT long            G_interval_number;

/* string array of reading parameter item */
#ifdef ARMAX1_MAIN
char                *G_parameter_item[] = {
                                    "IDENTIFICATION_ORDER:::::::::::::",
                                    "RECORD_LENGTH:::::::::::::::::::",
                                    "U_DATA_FILENAME:::::::::::::::::",
                                    "Y_DATA_FILENAME:::::::::::::::::",
                                    "INIT_PARAMETER_SETTING_FILENAME::",
                                    "NUMBER_OF_SKIPPING_DATA::::::::::",
                                    "NUMBER_OF_IDENTIFICATION:::::::::",
                                    "FORGETTING_FACTOR_MU:::::::::::::",
                                    "FORGETTING_FACTOR_INIT::::::::::::",
                                    "CONTRACTION_FACTOR_MU:::::::::::::",
                                    "PREVENTION_OF_DIVERGNECE:::::::::",
                                    "STABLE_CHECK_SWITCH::::::::::::::",
                                    "INIT_VALUE_OF_DIAGONAL_P_MATRIX::",
                                    "ACTIVATION_METHOD_SWITCH:::::::::",
                                    "ACTIVATION_INTERVAL::::::::::::::",
                                    "ACTIVATION_POWER::::::::::::::::",
                                    "OUTPUT_FILENAME:::::::::::::::::",
                                    "NUMBER_OF_OUTPUT_INTERVAL::::::::",
                                    "BINARY_POLE_OUTPUT_FILENAME::::::",
                                    "P/L_OUTPUT_FILENAME::::::::::::::" };
#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
/* ------+---------+---------+---------+---------+---------+---------+------- */
```

4.11 Prototype Header File

```
/* ------1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/  ARMAX Parameter Calculation Program by RPE Method.                        _/
_/                                                                            _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
---------1---------2---------3---------4---------5---------6---------7-------- */


/* ================================================================================ */
/* define.                                                                          */
/* ================================================================================ */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* armax1_main.c                                                                    */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
extern void    Print_Usage(void);
extern void    Title(void);
extern int     AutoMain(char **);


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* armax1_read.c                                                                    */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
extern int     ReadFileCall(char *);
extern long    Binary_File_Check(FILE *, int);
extern int     Std_Reading(FILE *, char *, int);


/* ---------+---------+---------+---------+---------+---------+---------+--------- */
/* armax1_call.c                                                                    */
/* ---------+---------+---------+---------+---------+---------+---------+--------- */
extern int     Armax_Call(void);
extern int     Starting_Check(void);
extern int     Parameter_Init(void);
extern void    Parameter_Std_Init(void);
extern double  Read_Data(FILE *, int);
```

```
extern int      Method_Judgement(void);

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

/* ------+---------+---------+---------+---------+---------+---------+--------- */
/* 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]);
```

4.12 Report File Example

```
── ARMAX Identification ──────────────────────────────────────────
JOB No. 1
Setup File Name = test1

IDENTIFICATION_ORDER:::::::::::: 2
RECORD_LENGTH:::::::::::::::::::: 2
U_DATA_FILENAME:::::::::::::::::: NULL
Y_DATA_FILENAME:::::::::::::::::: /home/asuka/DATA/test
INIT_PARAMETER_SETTING_FILENAME:: NULL
NUMBER_OF_SKIPPING_DATA:::::::::: 0
NUMBER_OF_IDENTIFICATION::::::::: 600000
FORGETTING_FACTOR_MU:::::::::::: 1.0
FORGETTING_FACTOR_INIT::::::::::: 1.0
CONTRACTION_FACTOR_MU:::::::::::: 1.0
PREVENTION_OF_DIVERGNECE::::::::: 1000.0
STABLE_CHECK_SWITCH:::::::::::::: ON
INIT_VALUE_OF_DIAGONAL_P_MATRIX:: 1000.0
ACTIVATION_METHOD_SWITCH::::::::: OFF
ACTIVATION_INTERVAL:::::::::::::: 100
ACTIVATION_POWER::::::::::::::::: 1
OUTPUT_FILENAME:::::::::::::::::: ARMA1
NUMBER_OF_OUTPUT_INTERVAL::::::::: 500
BINARY_POLE_OUTPUT_FILENAME:::::: NULL
P/L_OUTPUT_FILENAME:::::::::::::: NULL

ARMAX(2, 0, 2) Identification
Algorithm = ELS
Start of Calculation = 2000/02/13 15:50:50
End of Calculation   = 2000/02/13 15:51:24
Requirement Time     = 34.000000 [sec]


── ARMAX Identification ──────────────────────────────────────────
JOB No. 2
Setup File Name = test2
```

```
IDENTIFICATION_ORDER:::::::::::::: 2
RECORD_LENGTH:::::::::::::::::::::: 2
U_DATA_FILENAME:::::::::::::::::::: NULL
Y_DATA_FILENAME:::::::::::::::::::: /home/asuka/DATA/test
INIT_PARAMETER_SETTING_FILENAME:: NULL
NUMBER_OF_SKIPPING_DATA:::::::::: 0
NUMBER_OF_IDENTIFICATION::::::::: 600000
FORGETTING_FACTOR_MU:::::::::::::: 0.999
FORGETTING_FACTOR_INIT::::::::::: 0.999
CONTRACTION_FACTOR_MU::::::::::::: 0.999
PREVENTION_OF_DIVERGNECE::::::::: 1000.0
STABLE_CHECK_SWITCH:::::::::::::: ON
INIT_VALUE_OF_DIAGONAL_P_MATRIX:: 1000.0
ACTIVATION_METHOD_SWITCH::::::::: PMDA
ACTIVATION_INTERVAL:::::::::::::: 100
ACTIVATION_POWER::::::::::::::::: 1
OUTPUT_FILENAME:::::::::::::::::: ARMA2
NUMBER_OF_OUTPUT_INTERVAL:::::::: 500
BINARY_POLE_OUTPUT_FILENAME:::::: NULL
P/L_OUTPUT_FILENAME:::::::::::::: NULL

ARMAX(2, 0, 2) Identification
Algorithm = RPE & PMDA
Start of Calculation = 2000/02/13 15:51:29
End of Calculation   = 2000/02/13 15:52:09
Requirement Time     = 40.000000 [sec]
============================ NORMAL TERMINATION ============================
```

5　バイナリ・データ変換プログラム

5.1　バイナリ・データ変換プログラムの使用法

これまで我々はプログラム開発をＭＳ＿ＤＯＳ上で行ってきたため、シミュレーションデータ、並びに実験データのほとんどがＭＳ＿ＤＯＳ形式の２バイトデータとして保存されている。ところで、ＭＳ＿ＤＯＳ上の２バイトバイナリデータとＵＮＩＸ上での２バイトバイナリデータでは、互いに上位１バイトと下位１バイトの順序が順序が逆転しており、プログラムをＵＮＩＸプラトホームに移行したことにより、これまで用いてきたデータもＵＮＩＸ上で利用できるように変換する必要がある。そこで、Ａｐｐｅｎｄｉｘ　Ｃに示すプログラムEn_conv.outを作成し、これまでのＭＳ＿ＤＯＳ上の２バイトバイナリデータを有効にＵＮＩＸ上で利用できるようにした。

En_conv.outの利用法

ＭＳ＿ＤＯＳ上の２バイトバイナリデータ を endian_conv.out ファイルのあるディレクトリにコピーし、データ変換後のデータファイル名を指定して endian_conv.out を実行するだけでよい。

実行プログラムがＵＮＩＸマシンのディレクトリ asuka/DATA に

ＭＳ＿ＤＯＳ上の２バイトバイナリデータ：ID010.ch1

があり、一方、En_conv.ou が asuka/TOOL/bin のなかに在るとする。

また、変換されたデータを書き込むファイル名を：ID010_ch1.bin

とすれば、

```
cd home/asuka/DATA
$home/asuka/TOOL/bin endian_conv.out    ID010.ch1    ID010_ch1.bin
```

を実行するだけでよい。次頁に変換プログラムのリストを示す。

5.2 Conversion Program

```
/* -----1---------2---------3---------4---------5---------6---------7---------8
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
_/                                                                            _/
_/   Conversion Program of Binary Data in MS_DOS Format to UNIX Format        _/
_/                                                                            _/
_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/*/



#include <stdio.h>
#include <stdlib.h>

void main(int argc, char **argv)
{
        FILE    *fin, *fout;
        char    c1, c2;
        int             check;


        if ( argc != 3 ) {
                printf("¥nUsage: endian_conv.out  input_file  output_file¥n¥n");
                exit(0);
        }

        if ( (fin = fopen(argv[1], "rb")) == NULL ) {
                printf("Cannot open the input file [%s].¥n", argv[1] );
                exit(0);
        }
        if ( (fout = fopen(argv[2], "wb")) == NULL ) {
                printf("Cannot open the output file [%s].¥n", argv[2] );
                exit(0);
        }

        while ( 1 ) {
                check = 0;
                check += fread(&c1, 1, 1, fin);
                check += fread(&c2, 1, 1, fin);
```

```
                if ( check != 2 ) {
                        break;
                }
                fwrite(&c2, 1, 1, fout);
                fwrite(&c1, 1, 1, fout);
        }

        fclose(fin);
        fclose(fout);
}
```

付録

ＡＲＭＡＸ１使用マニュアル

目　次

はじめに

【1】 ＡＲＭＡＸ とは

ＡＲＭＡＸ (ARMAX one)は，離散系のシステムに対してその入出力時系列データからＡＲＭＡＸ (Auto Regressive Moving Average eXogeneous) モデルで適応フィルタリングを実行するプログラムです．

ＡＲＭＡＸモデルの推定パラメータを算出するベースアルゴリズムは再帰予測誤差 (Recursive Prediction Error：ＲＰＥ)法です．計算条件を与えるパラメータの設定次第ではＲＰＥ法の特殊な方法であるＰＬＲ (PseudoLinear Regression) 法・ＥＬＳ (Extended Least Squares) 法・ＲＭＬ (Recursive maximum likelihood) 法として計算させることも可能です．また，非定常状態システムのパラメータ変化への追従性を向上させるアルゴリズムとしてＰＭＡ (P-Matrix Activation) 法及びＰＭＤＡ (P-Matrix Diagonal Activation) 法をＲＰＥ法に採り入れることもできます．

【2】 ＡＲＭＡＸ の特徴

ＡＲＭＡＸ には以下のような特徴があります．

計算条件等必要なパラメータは基本的にはユーザーがジョブファイルを作成して入力することになりますが，その後のパラメータの変更などはコンソール上から行えるようにしています．

ジョブファイルの選択はＡＲＭＡＸ 側でファイルのリストを表示しますのでその中からカーソルキーで選べます．

できるだけエラー判定を密に行い，ユーザーのケアレスミスを背負ったまま計算しないようにしています．

結果の出力にはＣＲＴに表示するとともに，テキストファイル出力します．また，オプションによりポール (極) の値を計算してその結果をバイナリーファイルとして出力したり，Ｐマトリックス及びゲインベクトルの値をテキストファイルとして出力可能です．

第１部　必要な環境

ＵＮＩＸワークステーション：

ＯＳ：Ｓｏｌａｒｉｓ　2.6

【２】　使用していく上で必要となるもの

フルスクリーンエディタ：
　　ジョブファイルの作成に必要です．ワープロでも作成は可能ですが，その場合の保存形式は必ず「テキスト形式」にして下さい．

出力データ処理用のユーティリティプログラム：
　　計算結果は逐次ＣＲＴ上で確認することもできますが，計算間隔が短い場合などは，画面上を早足で流れてしまいますので，最終的には同時に出力される計算結果ファイルに頼ることになると思います．その場合，研究内容によってはＡＲ部だけとかＭＡ部だけの数列としてデータが欲しいというようなこともあり得ると思います．残念ながらＡＲＭＡＸ　ではそこまでサポートしていませんので，ユーザーが計算結果ファイルからデータを切り出すプログラムを用意する必要があります．しかし，計算結果ファイル内のデータの並びは，切り出しが容易に行えるように記述していますので簡単なプログラムで処理できます．
　　また，バイナリーのポール出力ファイル(オプションで出力できます)に関しては８バイトバイナリーファイルとして出力しますので，そのままファイルをオープンしても人間には記述内容が分かりません．ですから必要に応じてテキスト形式にコンバートするプログラムが必要です．

第２部　基本的な操作方法

【１】　プログラム方式

プログラム方式は，ＡＲＭＡＸ　の複数個のジョブファイルを逐次読み込みながらバッチ処理により作業を進めていく方式です．一旦プログラム方式で起動されると，強制的に終了させる場合をのぞいてはユーザーがコンソールから指示を与えることはありません．従って夜間や外出時などＣＰＵが他の作業に占拠されない時に，効率よくＡＲＭＡＸモデル同定を行うことができます．

【２】　起動方法とオプションファイル

コンソールから

ＡＲＭＡＸ１　　コントロールファイル名　　リポートファイル名

と入力する事でプログラム方式による起動が行われます．以下に，コントロールファイルとリポートファイルについて説明します．

コントロールファイル：

単に，処理したいジョブファイル名を羅列したものです．例えば，

```
test1
test2
test3
test4
```

と記述されたコントロールファイルを指定した場合，"test1"から順に"test4"まで順にジョブファイルをロードして実行します．

リポートファイル：

コンソール方式では，エラーが生じればそこでエラーメッセージを表示しユーザーがエラーの原因を確認できますが，プログラム方式の場合にはエラーが生じた場合にはその処理を放棄して次の処理を行うことになりますからユーザーがエラーの発生とその原因を確認できません．そこでバッチ処理に基づく実行結果をリポートファイルとして出力することにより，実行状況の確認とエラーが生じた場合の原因解明の一助とするようにしています．

【3】 リポートファイルの例

以下に，上記のコントロールファイルによるプログラム方式の実行時のリポートファイル出力結果の例について説明します．なお，ここでいう Set Up File Name は，ジョブファイル名のことです．

例１：

```
Process Number   : 1
Set Up File Name : test1
Result           : SUCCESS

Process Number   : 2
Set Up File Name : test2
Result           : ERROR
Comment          : On Reading Set Up File

Process Number   : 3
Set Up File Name : test3
Result           : SUCCESS

Process Number   : 4
Set Up File Name : test4
Result           : SUCCESS
```

この例の場合，"test2"というジョブが実行できなかったことを示しています．またその原因としては，ジョブファイルの読み込み時にエラーが発生していることから，ジョブファイルの記述ミスが考えられます．その他のジョブに関してはすべて正常に終了しています．

例２：

```
Process Number   : 1
Set Up File Name : test1
Result           : SUCCESS

Process Number   : 2
Set Up File Name : Manual Termintion
Result           : SUCCESS
```

この例の場合，"test1"実行時に，ユーザーから強制終了の命令がコンソールより指令されたために，"test1"のみ実行して手動終了による処理によりバッチ処理を終了していることを示しています．

第３部　ジョブファイル

【１】　ジョブファイルとは

ＡＲＭＡＸ　に計算させるためには，２０種類の様々なパラメータ等の指定を行わなければなりません．コンソールから逐一入力することも考えられますが，合理的ではありません．そこでフォーマットが定まったジョブファイルにしておくことにより，コンソール上での負担を軽減するとともに，プログラム方式での起動を可能にしています．

ＡＲＭＡＸ　はジョブファイルの指示に従って処理を行います．従って起動後は，まずジョブファイルの読み込みを行う必要があります．

【２】　ジョブファイルの構成

ジョブファイルの記述構成自体はごく簡単なものです．構成としては１枚のカードとしてみなしていただければ結構です．

以下にその例を示します．

```
          識　別　領　域                 ﾊﾟﾗﾒｰﾀ領域
←――――――――――――――――――――――→  ←――→
IDENTIFICATION_ORDER:::::::::::::: 2
RECORD_LENGTH:::::::::::::::::::::: 8
U_DATA_FILENAME:::::::::::::::::::: NULL
Y_DATA_FILENAME:::::::::::::::::::: WNC.BIN
INIT_PARAMETER_SETTING_FILENAME:: NULL
NUMBER_OF_SKIPPING_DATA:::::::::: 0
NUMBER_OF_IDENTIFICATION:::::::::: 10000
FORGETTING_FACTOR_MU:::::::::::::: 1.0
FORGETTING_FACTOR_INIT::::::::::: 1.0
CONTRACTION_FACTOR_MU:::::::::::: 1.0
PREVENTION_OF_DIVERGNECE:::::::::: 1000.0
STABLE_CHECK_SWITCH:::::::::::::: ON
INIT_VALUE_OF_DIAGONAL_P_MATRIX:: 1000.0
ACTIVATION_METHOD_SWITCH::::::::: OFF
ACTIVATION_INTERVAL:::::::::::::: 100
ACTIVATION_POWER::::::::::::::::: 1
OUTPUT_FILENAME::::::::::::::::::: ARMA1
NUMBER_OF_OUTPUT_INTERVAL:::::::: 500
BINARY_POLE_OUTPUT_FILENAME:::::: pole1
P/L_OUTPUT_FILENAME:::::::::::::: NULL
```

【3】 ジョブファイル作成上の注意事項

識別領域とパラメータ領域：

上記の例のようにジョブファイルの構成としては，識別領域とパラメータ領域の２つの領域から構成されています．

識別領域は，ＡＲＭＡＸ が入力解析を行うために必要ですので，ユーザーがこの部分を一語たりとも書き換えることはできません．書き換えてしまった場合にはＡＲＭＡＸ は自分用のジョブファイルとはみなしませんので以降のアクセスを拒否します．一方パラメータ領域は，逆にユーザーが定義する領域です．ＡＲＭＡＸ に計算させたい条件で適当な数値等を記述して下さい．ただし，識別領域とパラメータ領域の間には必ず半角スペースを１つ以上挿入して下さい．

ジョブファイル名の拡張子：

ジョブファイルはジョブファイル読み込みモードで読み込むことになりますが，ＡＲＭＡＸ では，慣習上とうていジョブファイルとは思われない拡張子を有したファイルは選択候補の中から除外しています．選択除外拡張子をつけた場合ロードできなくなりますので注意して下さい．

選択除外拡張子は以下の通りです．

・ソースプログラム関連	＝	H , C , PAS, FOR, BAS
・実行ファイル関連	＝	COM, EXE, OBJ, BAT
・環境関連	＝	SYS
・文章関連	＝	DOC, BAK
・データ関連	＝	FFT, BIN
・その他	＝	PRJ, CFG, MAK, DSK
・初期パラメータファイル	＝	111, 222, 333, 444, 555, 666, 777, 888, 999

第４部　各パラメータの意味と制限

ジョブファイルに記述する各パラメータの意味やその内容または制限などについて以下に示します．ただし，各パラメータの物理的な意味については，述べませんので，必要な方は「おわりに」で述べています参考資料を参照して下さい．

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	IDENTIFICATION_ORDER::::::::::::
意　味　・　内　容	ＡＲＭＡＸ同定の同定次数を設定します．
制　　限	１以上９以下の整数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(0) Order

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	RECORD_LENGTH::::::::::::::::::::
意　味　・　内　容	対象時系列データのレコード長を設定します． 選択できる時系列データは，２バイト(整数型)か８バイト(倍精度実数型)のいずれかです．
制　　限	２ or ８
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(B) Record Length

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	U_DATA_FILENAME:::::::::::::::::
意　味　・　内　容	ＡＲＭＡＸ同定対象システムへの入力時系列データファイル名を設定します．入力時系列データがない場合はARMAX(X, 0, X)同定となります．
制　　限	入力時系列データがない場合は　NULL　と記述
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(U) U Filename　or　f･4

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	Y_DATA_FILENAME::::::::::::::::::
意　味　・　内　容	ＡＲＭＡＸ同定対象システムからの出力時系列データファイル名を設定します．出力時系列データが存在しない状況は許されません．
制　　限	特になし
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(Y) Y Filename

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	INIT_PARAMETER_SETTING_FILENAME::
意　味　・　内　容	ＡＲＭＡＸ同定時における初期パラメータを与えるファイル名を設定します．初期パラメータファイルがない場合はデフォルト値で設定されます．
制　　限	初期パラメータファイルがない場合 NULL と記述
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(I) Inital File

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	NUMBER_OF_SKIPPING_DATA:::::::::::
意　味　・　内　容	時系列データによっては，中間部だけのデータをＡＲＭＡＸ同定したい場合などがあります．そこで最初にスキップするデータ数を設定します．
制　　限	０以上かつ，用意されている時系列の総データ数未満の整数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(S) Skip Data

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	NUMBER_OF_IDENTIFICATION::::::::::
意　味　・　内　容	ＡＲＭＡＸ同定対象データ数を設定します．
制　　限	１以上かつ，時系列データにおける総データ数からスキップデータ数を引いたデータ数以下の整数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(N) Using Number

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	FORGETTING_FACTOR_MU::::::::::::
意　味　・　内　容	忘却係数λの更新ファクターμを設定します.
制　　限	０より大きく１以下の実数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(1) F. Factor Mu

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	FORGETTING_FACTOR_INIT:::::::::::
意　味　・　内　容	忘却係数λの初期値λ_0を設定します.
制　　限	０より大きく１以下の実数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(2) F. Factor Init

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	CONTRACTION_FACTOR_MU::::::::::::
意　味　・　内　容	縮約係数Ｋの更新ファクターμを設定します.
制　　限	０以上１以下の実数
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(3) C. Factor Mu

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	PREVENTION_OF_DIVERGNECE:::::::::
意　味　・　内　容	ＡＲＭＡＸ同定時，Ｐマトリックスの値が発散することを防ぐパラメータです．このパラメータは非活性化法及びＰＭＡ法に関してはＵＤＬマトリックスの対角要素，ＰＭＤＡ法に関してはＰマトリックスの対角要素に作用します．それぞれの要素が設定した値を超えたときには，設定値に置き換えます．
制　　限	０より大きい実数 (MAX:1.7E+308)
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(4) Devergent Prev.

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	STABLE_CHECK_SWITCH::::::::::::::
意　味　・　内　容	ＲＰＥ法のアルゴリズムにおいて，プロジェクションアルゴリズムを作用させるかどうかのスイッチです．
制　　限	作用させるとき　ON　と記述 作用させないとき OFF と記述
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(5) Stability Check

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	INIT_VALUE_OF_DIAGONAL_P_MATRIX::
意　味　・　内　容	Ｐマトリックスの対角要素の初期値を設定します
制　　限	０より大きい実数 (MAX:1.7E+308)
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(6) P Matrix Init.

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	ACTIVATION_METHOD_SWITCH:::::::::
意　味　・　内　容	活性化アルゴリズムを作用させるかどうか，また作用させる場合ＰＭＡ法を選択するのかＰＭＤＡ法を選択するのかを設定するスイッチです．
制　　限	作用させないとき　OFF　と記述 ＰＭＡ法を作用させるとき　PMA　と記述 ＰＭＤＡ法を作用させるとき PMDA と記述
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(7) Activation

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	ACTIVATION_INTERVAL::::::::::::: 100
意　味　・　内　容	活性化アルゴリズムを作用させる際の，活性化間隔を設定します．(OFF の場合も必要です．)
制　　限	１以上の整数 (MAX:2,147,483,647)
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(8) Act. Interval

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	ACTIVATION_POWER::::::::::::::::::
意　味　・　内　容	活性化アルゴリズムを作用させる際の，活性化係数を設定します．(OFF の場合も必要です．)
制　　限	１以上の実数 (MAX:1.7E+308)
起動後変更モード	ﾊﾟﾗﾒｰﾀﾁｪﾝｼﾞﾓｰﾄﾞ　：(9) Act. Power

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	OUTPUT_FILENAME:::::::::::::::::
意　味　・　内　容	計算結果出力ファイルのパス名を設定します．
制　　限	１２８文字以内
起動後変更モード	出力ﾌｧｲﾙﾁｪﾝｼﾞﾓｰﾄﾞ：(0) Filename

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	NUMBER_OF_OUTPUT_INTERVAL:::::::: 500
意　味　・　内　容	計算結果を逐一出力ファイルにした場合，同定対象データ数が多いとファイルサイズは膨大なものになります．また実行時間も長くなります．そこでこのパラメータは出力間隔を設定します．
制　　限	１以上かつ，時系列データにおける総データ数からスキップデータ数を引いたデータ数以下の整数
起動後変更モード	出力ﾌｧｲﾙﾁｪﾝｼﾞﾓｰﾄﾞ：(1) Output Interval

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	BINARY_POLE_OUTPUT_FILENAME::::::
意　味　・　内　容	ＡＲＭＡＸ同定パラメータのＡＲ部よりポールの値を逐次計算し，その結果(実部のみ)を出力するバイナリーファイルのパス名を設定します．
制　　限	１２８文字以内で拡張子の設定禁止 出力しない場合は NULL と記述
起動後変更モード	出力ﾌｧｲﾙﾁｪﾝｼﾞﾓｰﾄﾞ：(2) Filename

ｼﾞｮﾌﾞﾌｧｲﾙでの識別	P/L_OUTPUT_FILENAME::::::::::::::
意　味　・　内　容	計算結果出力ファイルとともに，その時点でのＰマトリックス及びゲインベクトルの値を出力します．
制　　限	１２８文字以内 出力しない場合は NULL と記述
起動後変更モード	出力ﾌｧｲﾙﾁｪﾝｼﾞﾓｰﾄﾞ : (3) Filename

第５部　初期パラメータファイル

【１】　初期パラメータファイルとは

ＡＲＭＡＸ　により逐次同定する際，あらかじめシステムのおおよその状態が既知である場合には各同定パラメータに対して適切な初期値を与えることによって同定初期の発散傾向が緩和されパラメータの収束に要する時間を短縮することが可能です．ＡＲＭＡＸ　にその初期値を教えてやる手段となるのが初期パラメータファイルです．

初期パラメータファイルにより同定パラメータの初期値として与えられるものは，ＡＲＭＡＸパラメータ・Ｐマトリックス(またはＵＤＬ分解マトリックス)・ゲインベクトルの３種類についてです．

なお，初期パラメータファイルが存在しない場合これらの値は，

・ＡＲＭＡＸパラメータ　：　0.0
・Ｐ(またはＵＤＬ分解)マトリックスの非対角要素：　0.0
・Ｐ(もたはＵＤＬ分解)マトリックスの対角要素　：　ジョブデータ設定値
・ゲインベクトル　：　0.0

とプログラム内部で与えられます．

【２】　初期パラメータファイルの構成

初期パラメータファイルの記述で注意しなければならないことは，同定次数に依存しているということです．従って同じ時系列データに対する同定であっても２次で同定する場合と３次で同定する場合では，初期パラメータファイルは同一のものは使用できないということです．２次の同定については，２次用の初期パラメータファイルを，３次の同定については３次用の初期パラメータファイルを用意する必要があります．

初期パラメータファイルの主要な構成としては，

・宣言部
・ＡＲＭＡＸパラメータベクトル部
・Ｐ(またはＵＤＬ分解)マトリックス部
・ゲインベクトル部

の４カ所からなっています．書式が定義されていますのでジョブファイルと同様ユーザーが変更できる部分は限られており，それ以外の部分を書き換えてしまうと正常に読み込まれなくなります．

具体的な例として２次の同定の場合の例を以下に示します．他の同定次数の場合も基本的には同じような形式となりますのでユーザーの方で拡張して下さい．

INITIAL_PARAMETER_FILE　　*2*_IDENTIFICATION

PARAMETER
theta[1] = *-0.68061E+00*
theta[2] = *-0.17901E-01*
theta[3] = *0.0*
theta[4] = *0.0*
theta[5] = *-0.91462E-01*
theta[6] = *-0.32013E-02*

P_MATRIX_or_UDL_MATRIX
P[1][1] = *1000.0*
P[1][2] = *-0.6944181977E+00*
P[1][3] = *0.0000000000E+00*
P[1][4] = *0.0000000000E+00*
P[1][5] = *0.9965090479E+00*
P[1][6] = *-0.7116678003E+01*
P[2][1] = *0.0000000000E+00*
P[2][2] = *1000.0*
P[2][3] = *0.0000000000E+00*
P[2][4] = *0.0000000000E+00*
P[2][5] = *-0.6373965169E+00*
P[2][6] = *0.5106580718E+01*
P[3][1] = *0.0000000000E+00*
P[3][2] = *0.0000000000E+00*
P[3][3] = *1000.0*
P[3][4] = *0.0000000000E+00*
P[3][5] = *0.0000000000E+00*
P[3][6] = *0.0000000000E+00*
P[4][1] = *0.0000000000E+00*
P[4][2] = *0.0000000000E+00*
P[4][3] = *0.0000000000E+00*
P[4][4] = *1000.0*
P[4][5] = *0.0000000000E+00*
P[4][6] = *0.0000000000E+00*
P[5][1] = *0.0000000000E+00*
P[5][2] = *0.0000000000E+00*
P[5][3] = *0.0000000000E+00*
P[5][4] = *0.0000000000E+00*
P[5][5] = *1000.0*
P[5][6] = *-0.7129643460E+01*
P[6][1] = *0.0000000000E+00*
P[6][2] = *0.0000000000E+00*
P[6][3] = *0.0000000000E+00*
P[6][4] = *0.0000000000E+00*
P[6][5] = *0.0000000000E+00*
P[6][6] = *1000.0*

L_VECTOR
L[1] = *0.2206451348E-06*
L[2] = *-0.1176002930E-06*
L[3] = *0.0000000000E+00*
L[4] = *0.0000000000E+00*
L[5] = *0.2286913151E-06*
L[6] = *0.1882946871E-07*

上記の斜字体の部分がユーザーが書き換える部分です．特に宣言部の同定次数の表記を忘れないようにして下さい．

【３】　初期パラメータファイル作成上の注意事項

初期パラメータファイル名の拡張子：

前述しましたように初期パラメータファイルは同定次数依存であることからその認識を容易にして誤った初期パラメータファイルを選択しないように，初期パラメータファイル名の拡張子の部分をＡＲＭＡＸ　の方で定型指定しています．この指定拡張子以外の拡張子をつけたファイルはＡＲＭＡＸ　は初期パラメータファイルとルとはみなしませんのでロードもできません．従って，必ず指定された拡張子をファイル名に付して下さい．

指定拡張子は以下の通りです．

・１次同定の場合　＝　111
・２次同定の場合　＝　222
・３次同定の場合　＝　333
・４次同定の場合　＝　444
・５次同定の場合　＝　555
・６次同定の場合　＝　666
・７次同定の場合　＝　777
・８次同定の場合　＝　888
・９次同定の場合　＝　999

第６部　出力ファイル

【１】　計算結果出力ファイル例

・ＡＲＭＡ(２，２)同定の場合

```
500       POLE:CONVERGE                          ;データ位置　ﾎﾟｰﾙ計算状態
      -1.818465e-01        1.145498e-01 ;ＡＲパラメータ
      -1.979683e-01        1.355911e-01 ;ＭＡパラメータ
       9.092327e-02    3.260103e-01     ;ポール(実部　　虚部)
       9.092327e-02   -3.260103e-01     ;ポール(実部　　虚部)

                  ⋮                     ;５００データおきに表示

10000     POLE:CONVERGE
      -2.017336e-02        1.013687e-01
      -3.139404e-02        1.097989e-01
       1.008668e-02    3.182248e-01
       1.008668e-02   -3.182248e-01
```

Condition of Identification

```
ARMAX(2, 0, 2) Identification          ;同定次数
Used Algorithm = ELS                   ;アルゴリズム

IDENTIFICATION_ORDER:............. 2
RECORD_LENGTH:.................... 8
U_DATA_FILENAME:................... NULL
Y_DATA_FILENAME:................... WNC.BIN
INIT_PARAMETER_SETTING_FILENAME:: NULL
NUMBER_OF_SKIPPING_DATA:.......... 0
NUMBER_OF_IDENTIFICATION:........ 10000
FORGETTING_FACTOR_MU:............ 1.000000
FORGETTING_FACTOR_INIT:........... 1.000000
CONTRACTION_FACTOR_MU:........... 1.000000
PREVENTION_OF_DIVERGNECE:........ 1000.000000
STABLE_CHECK_SWITCH:............. ON
INIT_VALUE_OF_DIAGONAL_P_MATRIX:: 1000.000000
ACTIVATION_METHOD_SWITCH:........ OFF
ACTIVATION_INTERVAL:............. 100
ACTIVATION_POWER:................ 1.000000
OUTPUT_FILENAME:................. ARMA1
NUMBER_OF_OUTPUT_INTERVAL:....... 500
BINARY_POLE_OUTPUT_FILENAME:..... NULL
P/L_OUTPUT_FILENAME:............. plfile
```

ジョブデータ

システムの入力時系列データが存在する場合は，eXogeneousパラメータをＡＲ

パラメータとＭＡパラメータの間に表示します.

【２】　Ｐ／Ｌ出力ファイルの例

・ＡＲＭＡ(２，２)同定の場合

```
Value of Y_Data at 500 Data              =  2.179200e+00
Value of Prediction Error at 500 Data =  2.180556e+00
Value of The Residual at 500 Data      =  2.168875e+00

Elements of P Matrix at 500 Data
P[1][1] =  4.801549e-02      UDL[1][1] =  1.932953e-03
P[1][2] = -3.360774e-03      UDL[1][2] =  3.254003e-03
P[1][3] =  0.000000e+00      UDL[1][3] =  0.000000e+00
P[1][4] =  0.000000e+00      UDL[1][4] =  0.000000e+00
P[1][5] =  4.766749e-02      UDL[1][5] =  9.653512e-01
P[1][6] = -7.551400e-03      UDL[1][6] = -1.782738e-01
P[2][1] = -3.360774e-03      UDL[2][1] =  0.000000e+00
P[2][2] =  4.053164e-02      UDL[2][2] =  1.950882e-03
P[2][3] =  0.000000e+00      UDL[2][3] =  0.000000e+00
P[2][4] =  0.000000e+00      UDL[2][4] =  0.000000e+00
P[2][5] = -3.119924e-03      UDL[2][5] =  8.220027e-02
P[2][6] =  4.025520e-02      UDL[2][6] =  9.503465e-01
P[3][1] =  0.000000e+00      UDL[3][1] =  0.000000e+00
P[3][2] =  0.000000e+00      UDL[3][2] =  0.000000e+00
P[3][3] =  1.000000e+03      UDL[3][3] =  1.000000e+03
P[3][4] =  0.000000e+00      UDL[3][4] =  0.000000e+00
P[3][5] =  0.000000e+00      UDL[3][5] =  0.000000e+00
P[3][6] =  0.000000e+00      UDL[3][6] =  0.000000e+00
P[4][1] =  0.000000e+00      UDL[4][1] =  0.000000e+00
P[4][2] =  0.000000e+00      UDL[4][2] =  0.000000e+00
P[4][3] =  0.000000e+00      UDL[4][3] =  0.000000e+00
P[4][4] =  1.000000e+03      UDL[4][4] =  1.000000e+03
P[4][5] =  0.000000e+00      UDL[4][5] =  0.000000e+00
P[4][6] =  0.000000e+00      UDL[4][6] =  0.000000e+00
P[5][1] =  4.766749e-02      UDL[5][1] =  0.000000e+00
P[5][2] = -3.119924e-03      UDL[5][2] =  0.000000e+00
P[5][3] =  0.000000e+00      UDL[5][3] =  0.000000e+00
P[5][4] =  0.000000e+00      UDL[5][4] =  0.000000e+00
P[5][5] =  4.931041e-02      UDL[5][5] =  4.800533e-02
P[5][6] = -7.435156e-03      UDL[5][6] = -1.755295e-01
P[6][1] = -7.551400e-03      UDL[6][1] =  0.000000e+00
P[6][2] =  4.025520e-02      UDL[6][2] =  0.000000e+00
P[6][3] =  0.000000e+00      UDL[6][3] =  0.000000e+00
P[6][4] =  0.000000e+00      UDL[6][4] =  0.000000e+00
P[6][5] = -7.435156e-03      UDL[6][5] =  0.000000e+00
P[6][6] =  4.235845e-02      UDL[6][6] =  4.235845e-02
Elements of L Vector at 500 Data
L[1]      = -5.695638e-03
L[2]      =  1.027023e-03
L[3]      =  0.000000e+00
L[4]      =  0.000000e+00
L[5]      = -1.499768e-03
L[6]      =  3.283308e-03
```

⋮　　設定出力間隔ごとに表示

ＵＤＬ分解マトリックスの表示は活性化法を使用しない場合かまたは，ＰＭＡ法を使用した場合にのみ表示します．

【３】　ポール出力ファイル

ポール出力ファイルは，上述の計算結果出力ファイル及びＰ／Ｌ出力ファイルが設定出力間隔でテキスト形式で出力されるのに対して設定出力間隔に無関係に１回の推定ごとに８バイト倍精度実数型でバイナリ形式で書き込まれたものです．ただし，ファイルに格納されるのはポールの実部のみです．

また，ポールはその同定次数と同じ数だけ存在しますが，この処理としてはポールと同数のファイルを作成することによって解決しています．具体的に説明しますと，「第４部　各パラメータの意味と制限」でも述べていますが，ポール出力ファイル名の設定には拡張子の付加を禁止しています．なぜならば拡張子はＡＲＭＡＸ　で自動的に１，２，３，……とポールの存在個数だけ付加するようにしているからです．従って，ポール出力ファイル名をＰＯＬＥ１とした２次の同定の場合には，'POLE1.1'及び'POLE1.2'という二つのポール出力ファイルが作成されます．繰り返しになりますが，ポール出力ファイルはバイナリーファイルであるため，そのままではエディタなどで結果を見ることはできません．適当なテキストファイル変換ツールを用意して下さい．

第７部　エラーメッセージ一覧

【１】　起動時エラー

Auto Control File […] Cannot Open
プログラム方式で起動した際，コントロールファイルがオープンできない時に発報されます．コントロールファイル名に誤りがないかまたは指定ディレクトリ上に存在しているか確認して下さい．

Auto System Report File […] Cannot Open
プログラム方式で起動した際，リポートファイルがオープンできない時に発報されます．リポートファイル名に誤りがないかまたは指定ディレクトリ上に存在しているか確認して下さい．

【２】　ジョブファイル読込時エラー

Fitting File Dose Not Exist on The Specified PATH
ジョブファイルを選択する際，指定されているパス上にジョブファイルとして適合するファイルが見つからない時に発報されます．ジョブファイルに選択除外拡張子がつけられているか，実際に指定したファイルが存在していない可能性があります．

Cannot Open Set Up File […]
選択されたジョブファイルがオープンできない時に発報されます．ＭＳ－ＤＯＳ上のシステムエラーの可能性があります．

Illegal Value as Identification Order
ジョブファイル中の同定次数設定値が規定外数値の時に発報されます.

Illegal Value as Record Length of Time Series Data
ジョブファイル中の時系列データレコード長設定値が規定外数値の時に発報されます. レコード長として適合するのは, 2または8です.

Cannot Open Time Series Data File Inputted to System
ジョブファイル中で設定されているシステム入力時系列データファイルがオープンできない時に発報されます. ファイル名が誤っているか, 設定されたパス上に実際に存在しない可能性があります.

Cannot Open Time Series Data File Outputted from System
ジョブファイル中で設定されているシステム出力時系列データファイルがオープンできない時に発報されます. ファイル名が誤っているか, 設定されたパス上に実際に存在しない可能性があります.

Canont Open Initial Parameter File
ジョブファイル中で設定されている初期パラメータファイルがオープンできない時に発報されます. ファイル名が誤っているか, 設定されたパス上に実際に存在しない可能性があります.

First Declaration of The Initial Parameter File Is Unsuitable
設定された初期パラメータファイルの宣言部が所定の書式に従っていない時に発報されます.

There Is Discrepancy As to Identification Order
設定された初期パラメータファイルが，すでに設定されている同定次数に対応するものでない時に発報されます．

Illegal Value as Number of Skipping Data
ジョブファイル中のスキップデータ数設定値が規定外数値の時に発報されます．

Illegal Value as Number of Identification
ジョブファイル中の同定対象データ数設定値が規定外数値の時に発報されます．

Illegal Value as Constant Part of Forgetting Factor
ジョブファイル中の忘却係数λの更新ファクターμの設定値が規定外数値の時に発報されます．

Illegal Value as Initial Forgetting Factor
ジョブファイル中の忘却係数λの初期値λ_0の設定値が規定外数値の時に発報されます．

Illegal Value as Contraction Factor Mu
ジョブファイル中の縮約係数Kの更新ファクターμの設定値が規定外数値の時に発報されます．

Illegal Value as Constant for Divergent Prevention
ジョブファイル中のＰ(またはＵＤＬ分解)マトリックスの対角要素の発散防止の上限設定値が規定外数値の時に発報されます.

Illegal Switch as Stable Check
ジョブファイル中のプロジェクションアルゴリズムスイッチの設定が不適切な時に発報されます. スイッチとして適合するのは ON か OFF のいずれかです.

Illegal Value as Diagonal Elements of Initial P Matrix
ジョブファイル中のＰ(またはＵＤＬ分解)マトリックスの対角要素の初期設定値が規定外数値の時に発報されます.

Illegal Switch as Selection of Activation Method
ジョブファイル中の活性化法選択スイッチの設定が不適切な時に発報されます. スイッチとして適合するのは OFF か PMA または PMDA のいずれかです.

Illegal Value as Number of Activation Interval
ジョブファイル中の活性化間隔設定値が規定外数値の時に発報されます.

Illegal Value as Activation Power
ジョブファイル中の活性化量設定値が規定外数値の時に発報されます.

Unsuitable Output File Name.　Please Ovey MS-DOS Format
ジョブファイル中の計算結果出力ファイル名もしくはポール出力ファイル名の設定が不適切な時に発報されます.

Illegal Value as Number of Interval
ジョブファイル中の計算結果出力間隔設定値が規定外数値の時に発報されます.

Unsuitable P/L File Name.　Please Obey MS-DOS Format
ジョブファイル中のＰ／Ｌ出力ファイル名の設定が不適切な時に発報されます.

Probably, […] Is Not Binary Data File of … Byte
システム入出力時系列データファイルが既に設定されているレコード長で記述されたバイナリーデータファイルでないとＡＲＭＡＸ　が判断した時に発報されます．この判断はＡＲＭＡＸ　にとってのフェイタルエラーとみなします.

Format of Set Up File Is Unsuitable
ジョブファイルの記述が所定の書式でない時に発報されます．この原因としては，ジョブファイルとは全くの別のファイルを読み込んでいるか，ジョブファイル内で識別子に変更が加えられているか，または識別子領域とパラメータ領域の間にスペースが挿入されていないことなどが考えられます.

Cannot Open Specified File […]
システム入出力時系列データファイル名または初期パラメータファイル名を変更した際，変更されたファイルがオープンきない時に発報されます.

First Declaration of The Initial Parameter File Is Unsuitable
初期パラメータファイルを変更した際，変更された初期パラメータファイルの宣言部が所定の書式に従っていない時に発報されます．

Second Declaration of The Initial Parameter File Is Unsuitable
初期パラメータファイルを変更した際，変更された初期パラメータファイルで宣言されている対応次数が規定外数値の時に発報されます．

Fitting File Dose Not Exist on The Specified PATH
システム入出力時系列データファイルまたは初期パラメータファイルを変更する際，指定されているパス上に適合するファイルが見つからない時に発報されます．ファイルに選択除外拡張子がつけられているか，実際に指定したファイルが存在していない可能性があります．

[…] Is Illegal Value.　Cannot Renew
数値パラメータを変更した際，変更された値が規定外数値である時に発報されます．数値は更新されません．

[…] Is Unsuitable Filename
計算結果出力ファイル・ポール出力ファイル・Ｐ／Ｌ出力ファイル名を変更した際変更されたファイル名が不適切な時に発報されます．

【3】 実行時エラー

Number to Identify Exceed Number of Useable Data
同定対象データ数が時系列データの総データ数からスキップデータ数を差し引いた使用可能なデータ数を超過している時に発報されます.

Interval of Output Exceeds Number to Identify
計算結果出力間隔が同定対象データ数を超過している時に発報されます.

Check PATH of Output Filename(Result) or Disk Condition
計算結果出力ファイルがオープンできない時に発報されます. 指定ドライブ等の状態が正常でない可能性もあります.

Check PATH of Output Filename(Pole) or Disk Condition
ポール出力ファイルがオープンできない時に発報されます. 指定ドライブ等の状態が正常でない可能性もあります.

Check PATH of Output File(P/L) or Disk Condition
P／L出力ファイルがオープンできない時に発報されます. 指定ドライブ等の状態が正常でない可能性もあります.

Cannot Open Time Series File […]. Check Your Machine
システム入出力時系列データファイルがオープンできない時に発報されます. このエラーが発報された場合, ドライブ及びマシンのチェックが必要と思われます.

Now, Setting of Record Length Is 8 Byte. But It Seems to Be Your Mistake
システム入出力時系列データファイルのレコード長が８バイトに設定されている場合，ＡＲＭＡＸ が検査の結果２バイトのデータではないかと判断した時に発報される警告です．これはあくまで警告であり処理は継続可能です．

Now, Setting of Record Length Is 8 Byte. But It is Your Mistake Clearly
システム入出力時系列データファイルのレコード長が８バイトに設定されている場合，ＡＲＭＡＸ が検査の結果明らかに８バイトのデータとは考えられないと判断した時に発報されます．これはフェイタルエラーですので処理は中止されます．

U-Filename = Y-Filename. The Same Name Is Used
システム入出力時系列データファイルのファイル名が同一の時に発報されます．

There Is Discrepancy As to Identification Order
選択されている初期パラメータファイルの対応次数が，実際の同定次数に対応するものでない時に発報されます．

Iintial Parameter File Dose Not Obey Defined Format
選択されている初期パラメータファイルの記述が所定の書式に従っていない時に発報されます．

Illegal Value as Initial Parameter
選択されている初期パラメータファイルに与えられている初期値が規定外数値の時に発報されます．

【4】 ジョブファイルリメイク時エラー

New Set Up File […] Cannot Open
新たに設定れたリメイクジョブファイルがオープンできない時に発報されます.