PHD

On-line identification investigation

Ture, M.

Award date:
1992

Awarding institution:
University of Bath

ON-LINE IDENTIFICATION INVESTIGATION

Submitted by M. Türe, M.Sc., B.Sc.

for the degree of PhD

of the University of Bath

1992

" In the name of Allah. Most Gracious , Most Merciful. "

SUMMARY

In this study, on-line system identification methods are investigated for the continuous time model. The well known discrete time methods are reviewed for indirect methods. The transformation methods from discrete system to continuous system are given. Direct continuous model identification methods are explained and the quasilinearization of the Newton-Raphson method is implemented for the identification of the parameters of an aircraft.

The aircraft dynamics are reviewed to simulate the flight model. This review shows why the aircraft requires auto-control. The relations between the adaptive control for non-minimum phase and unstable systems and the identification are illustrated. The atmospheric turbulence effect on the identification is shown.

The hardware and the software of the implementation are developed for real time estimation. A personal computer and a TMS320C30 Digital Signal Processor are used for the flight modelling and the identification circuit, respectively.

CONTENTS

LIST OF SYMBOLS

The following symbols are used in the chapter 3 and 5.

Symbol	Meaning
A	Aircraft attitude
A	Aspect ratio, b^2/S
a	Aircraft lift curve slope, $dC_L/d\alpha$
b	Wing span (tip to tip)
C_D	Drag$/\frac{1}{2}\rho V^2 S$
C_L	Lift$/\frac{1}{2}\rho V^2 S$
C_{LT}	Tailplane lift$/\frac{1}{2}\rho V^2 S$
C_l	Rolling moment about $Ox/\frac{1}{2}\rho_e V_e^2 Sb$
C_m	Pitching moment about $Oy/\frac{1}{2}\rho V^2 S\bar{\bar{c}}$
C_n	Yawing moment about $Oz/\frac{1}{2}\rho_e V_e^2 Sb$
C_X, C_Z	Non-dimensional force coefficient
	$X/\frac{1}{2}\rho V^2 S$, $Z/\frac{1}{2}\rho V^2 S$
c	Wing chord
$\bar{\bar{c}}$	Mean aerodynamic chord of wing
D	Drag
D	Differential operator, d/dt
$\hat{D}$	Differential operator, $d/d\hat{t}$
e_x	$-I_{zx} / I_x$

Symbol	Meaning
e_z	$-I_{zx} / I_z$
G	Transfer function
g	Acceleration due to gravity
$\hat{g}$	$mg/\frac{1}{2}\rho_e V_e^2 S = C_L \sec\Theta$
g_x, g_y, g_z	Component of gravitational acceleration
g_θ, g_ψ, ...	Non-dimensional autopilot parameters
g_1	$g \cos \Theta_e$
$\hat{g}_1$	$\hat{g} \cos \Theta_e$
g_2	$g \sin \Theta_e$
$\hat{g}_2$	$\hat{g} \sin \Theta_e$
I_x	Moment of inertia about longitudinal (rolling) axis Ox
I_y	Moment of inertia about lateral (pitching) axis Ox
I_z	Moment of inertia about yawing axis Oz
I_{xy}	Product of inertia about Ox and Oy
I_{yz}	Product of inertia about Oy and Oz
I_{zx}	Product of inertia about Oz and Ox
i_x	I_x / mb^2
i_y	$I_y / m\bar{\bar{c}}^2$
i_z	I_z / mb^2

Symbol	Meaning
i_z	I_{zx} / mb^2
i_1	$(I_z - I_x) / I_y$
i_2	$(I_y - I_x) / I_z$
L	Lift
L_W, L_T	Wing lift, tail lift
L	Rolling moment about Ox
$\mathring{L}_p$, $\mathring{L}_r$, $\mathring{L}_v$, $\mathring{L}_\xi$, $\mathring{L}_\zeta$	Rolling moment derivatives, $\delta L/\delta p$, $\delta L/\delta r$, $\delta L/\delta v$, $\delta L/\delta \xi$, $\delta L/\delta \zeta$
L_p	Non-dimensional rolling moment derivative due to rate of roll, $\mathring{L}_p / \frac{1}{2}\rho_e V_e S b^2$
L_r	Non-dimensional rolling moment derivative due to rate of yaw, $\mathring{L}_r / \frac{1}{2}\rho_e V_e S b^2$
L_v	Non-dimensional rolling moment derivative due to rate of sideslip, $\mathring{L}_v / \frac{1}{2}\rho_e V_e S b$
L_ξ	Non-dimensional rolling moment derivative due to rate of ailerons, $\mathring{L}_p / \frac{1}{2}\rho_e V^2_e S b$
L_ζ	Non-dimensional rolling moment derivative due to rate of rudder, $\mathring{L}_\zeta / \frac{1}{2}\rho_e V_e^2 S b$
l	Distance of aerodynamic centre of tail plane aft of aeraerodynamic centre of aircraft without tail

Symbol	Meaning
l_F	Distance of aeodynamic centre of fin aft of c.g. of aircraft
l_T	Distance of aerodynamic centre of tailplane aft of c.g. of aircraft
l_p	$-L_p \; / \; i_x$
l_r	$-L_r \; / \; i_x$
l_v	$-\mu_2 L_v \; / \; i_x$
l_ξ	$-\mu_2 L_\xi \; / \; i_x$
l_ζ	$-\mu_2 L_\zeta \; / \; i_x$
M	Pitching moment about Oy
$\mathring{M}_q$, $\mathring{M}_u$, $\mathring{M}_w$, $\mathring{M}_{\dot{w}}$, $\mathring{M}_\eta$	Pitching moment derivatives, $\delta M/\delta q$, $\delta M/\delta u$, $\delta M/\delta w$, $\delta M/\delta \dot{w}$, $\delta M/\delta \eta$
M_q	Non-dimensional pitching moment derivative due to rate of pitch, $\mathring{M}_p \; / \; \frac{1}{2}\rho_e V_e S\bar{\bar{c}}^2$
M_u	Non-dimensional pitching moment derivative due to velocity increment along Ox, $\mathring{M}_u / \; \frac{1}{2}\rho_e V_e S\bar{\bar{c}}^2$
M_w	Non-dimensional pitching moment derivative due to velocity increment along Oz, $\mathring{M}_w / \; \frac{1}{2}\rho_e V_e S\bar{\bar{c}}^2$
$M_{\dot{w}}$	Non-dimensional pitching moment derivative due to rate of change of w, $\mathring{M}_{\dot{w}} / \; \frac{1}{2}\rho_e V_e S\bar{\bar{c}}^2$

Symbol	Meaning
M_η	Non-dimensional pitching moment derivative due to elevator, $\mathring{M}_\eta / \frac{1}{2}\rho_e V_e S\bar{\bar{c}}^2$
m	Aircraft mass
m_q	$-M_q \ / \ i_y$
m_u	$-\mu_1 M_u \ / \ i_y$
m_w	$-\mu_1 M_w \ / \ i_y$
$m_{\dot{w}}$	$-M_{\dot{w}} \ / \ i_y$
m_η	$-\mu_1 M_\eta \ / \ i_y$
N	Yawing moment about Oz
$\mathring{N}_p$, $\mathring{N}_r$, $\mathring{N}_v$, $\mathring{N}_\xi$, $\mathring{N}_\zeta$	Yawing moment derivatives, $\delta N/\delta p$, $\delta N/\delta r$, $\delta N/\delta v$, $\delta N/\delta \xi$, $\delta N/\delta \zeta$
N_p	Non-dimensional yawing moment derivative due to rate of roll, $\mathring{N}_p / \frac{1}{2}\rho_e V_e S b^2$
N_r	Non-dimensional yawing moment derivative due to rate of yaw, $\mathring{N}_r / \frac{1}{2}\rho_e V_e S b^2$
N_v	Non-dimensional yawing moment derivative due to rate of sideslip, $\mathring{N}_v / \frac{1}{2}\rho_e V_e S b$
N_ξ	Non-dimensional yawing moment derivative due to rate of ailerons, $\mathring{N}_p / \frac{1}{2}\rho_e V_e^2 S b$
N_ζ	Non-dimensional yawing moment derivative due to rate of rudder, $\mathring{N}_\zeta / \frac{1}{2}\rho_e V_e^2 S b$

Symbol	Meaning
n	Normal acceleration (in units of g)
	= normal load factor - 1
n_p	$-N_p / i_z$
n_r	$-N_r / i_z$
n_v	$-\mu_2 N_v / i_z$
n_ξ	$-\mu_2 N_\xi / i_z$
n_ζ	$-\mu_2 N_\zeta / i_z$
p	Aircraft angular velocity in roll
$\hat{p}$	τp
q	Aircraft angular velocity in pitch
$\hat{q}$	τq
r	Aircraft angular velocity in yaw
$\hat{r}$	τr
S	Wing area
s	Wind semi-span, $\frac{1}{2}b$
s	Laplace operator
t	Time
$\hat{t}$	Non-dimensional time, t/τ
U	Velocity component of c.g. along Ox in disturbed flight
U_e	Velocity component of c.g. along Ox in datum steady flight

Symbol	Meaning
u	$U - U_e$
$\hat{u}$	u / V_e
V	Velocity component of c.g. along Oy in disturbed flight
V	Resultant velocity of c.g. in disturbed longitudinal flight
V_e	Resultant velocity of aircraft c.g. in datum steady flight
v	Component of velocity increment of c.g. along Oy in disturbed flight.
$\hat{v}$	v / V_e (=β, for small angles of sideslip)
W	Aircraft weight, mg
W	Velocity component of c.g. along Oz in disturbed flight
W_e	Velocity component of aircraft c.g. in datum steady flight
w	Component of velocity increment of c.g. along Oy in disturbed flight.
$\hat{w}$	w / V_e
X	Force component along Ox
$\mathring{X}_q$, $\mathring{X}_u$, $\mathring{X}_w$, $\mathring{X}_{\dot{w}}$, $\mathring{X}_\eta$	Force component derivatives, $\delta X/\delta q$, $\delta X/\delta u$, $\delta X/\delta w$, $\delta X/\delta \dot{w}$, $\delta X/\delta \eta$

Symbol	Meaning
X_q	Non-dimensional force derivative due to rate of pitch, $\mathring{X}_q / \frac{1}{2}\rho_e V_e S\bar{\bar{c}}$
X_u	Non-dimensional force derivative due to velocity increment along Ox, $\mathring{X}_u / \frac{1}{2}\rho_e V_e S$
X_w	Non-dimensional force derivative due to velocity increment along Ox, $\mathring{X}_w / \frac{1}{2}\rho_e V_e S$
$X_{\dot{w}}$	Non-dimensional force derivative due to rate of change of w, $\mathring{X}_{\dot{w}} / \frac{1}{2}\rho_e S\bar{\bar{c}}$
X_η	Non-dimensional force derivative due to elevator, $\mathring{X}_\eta / \frac{1}{2}\rho_e V_e^2 Sb$
Ox	Axis through aircraft c.g. fixed in the aircraft in the forward direction in the plane symmetry. (For wind axes, in the steady state, Ox coincides direction of motion of c.g.)
x_q	$-X_q / \mu_1$
x_u	$-X_u$
x_w	$-X_w$
$x_{\dot{w}}$	$-X_{\dot{w}} / \mu_1$
x_η	$-X_\eta$
Y	Force component along Oy

Symbol	Meaning
$\mathring{Y}_p$, $\mathring{Y}_r$, $\mathring{Y}_v$, $\mathring{Y}_\xi$, $\mathring{Y}_\zeta$	Force component derivatives, $\delta Y/\delta p$, $\delta Y/\delta r$, $\delta Y/\delta v$, $\delta Y/\delta \xi$, $\delta Y/\delta \zeta$
Y_p	Non-dimensional force derivative due to rate of roll, $\mathring{Y}_p / \frac{1}{2}\rho_e V_e Sb$
Y_r	Non-dimensional force derivative due to rate of yaw, $\mathring{Y}_r / \frac{1}{2}\rho_e V_e Sb$
Y_v	Non-dimensional force derivative due to sideslip, $\mathring{Y}_v / \frac{1}{2}\rho_e V_e S$
Y_ξ	Non-dimensional force derivative due to ailerons, $\mathring{Y}_\xi / \frac{1}{2}\rho_e V_e^2 S$
Y_ζ	Non-dimensional force derivative due to rudder, $\mathring{Y}_\zeta / \frac{1}{2}\rho_e V_e^2 S$
Oy	Axis through aircraft c.g. fixed in the aircraft in the lateral direction, perpendicular to the plane of symmetry and positive to starboard.
y_p	$-Y_p / \mu_1$
y_r	$-Y_r$
y_v	$-Y_v$
y_ξ	$-Y_\xi / \mu_1$
y_ζ	$-Y_\zeta$
Z	Force component along Oz

Symbol	Meaning
$\mathring{Z}_q$, $\mathring{Z}_u$, $\mathring{Z}_w$, $\mathring{Z}_{\dot{w}}$, $\mathring{Z}_\eta$	Force component derivatives, $\delta Z/\delta q$, $\delta Z/\delta u$, $\delta Z/\delta w$, $\delta Z/\delta \dot{w}$, $\delta Z/\delta \eta$
Z_q	Non-dimensional force derivative due to rate of pitch, $\mathring{Z}_q / \frac{1}{2}\rho_e V_e S\bar{\bar{c}}$
Z_u	Non-dimensional force derivative due to velocity increment along Ox, $\mathring{Z}_u / \frac{1}{2}\rho_e V_e S$
Z_w	Non-dimensional force derivative due to velocity increment along Oz, $\mathring{Z}_w / \frac{1}{2}\rho_e V_e S$
$Z_{\dot{w}}$	Non-dimensional force derivative due to rate of change of w, $\mathring{Z}_{\dot{w}} / \frac{1}{2}\rho_e S\bar{\bar{c}}$
Z_η	Non-dimensional force derivative due to elevator, $\mathring{Z}_\eta / \frac{1}{2}\rho_e V_e^2 S$
Oz	Axis through aircraft c.g. fixed in the aircraft in the downward direction and perpendicular to Ox and Oy.
z_q	$-Z_q / \mu_1$
z_u	$-Z_u$
z_w	$-Z_w$
$z_{\dot{w}}$	$-Z_{\dot{w}} / \mu_1$
z_η	$-Z_\eta$
α	Incidence (angle of attack) of mean aerodynamic chord of wing

Symbol	Meaning
α_e	Incidence of Ox to the flight path in the steady state (positive upwards) ($\alpha_e = 0$ for wind axes)
β	Angle of sideslip (the angle the direction of the motion of the aircraft c.g. makes with th plane of the symmetry Oxz)
δ	Displacement
ξ, η, ζ	Angular displacements of ailerons , elevator and rudder, respectively
$\bar{\eta}$	Elevator angle to trim
η'	Increment elevetor angle from trimmed position
Θ_e	Inclination of Ox to the horizantal in the datum steady flight (positive upwards)
θ	Angle of pitch
μ_1	Longitudinal relative density parameter $m / \frac{1}{2}\rho_e S\bar{\bar{c}}$
μ_2	Lateral relative density parameter $m / \frac{1}{2}\rho_e Sb$
ρ	Air density
ρ_e	Air density in datum steady flight
τ	Magnitude of time unit

Symbol	Meaning
ϕ	Angle of bank
ψ	Angle of yaw

CHAPTER 1. INTRODUCTION

A new period of control theory was started with the introduction of the adaptive control in 1960's. Adaptive control was first proposed as a model reference controller by Whitaker and his colleagues [1]. Then, many researches were done to develop the adaptive system theory . In 1970's, the self tuning controller was presented to adaptive control by Aström [2]. Self tuning regulators (STRs) are very suitable to optimize the adaptive system, especially non-minimum phase and unstable systems.

On-line determination of process parameters is a key element in the adaptive control. It is an important part of a self tuning controller. It is also used in implicit model reference adaptive system. Therefore the identification methods have been developed in parallel with the adaptive control. Some old estimation methods have been progressed for the on-line systems. Generally, the identification method have been developed based on the discrete model of the systems because of the sampled data. The discrete model was very suitable for the first microprocessors, which were very slow and primitive when compared with todays. In the application, rather accurate process model are required for very sensitive systems. The continuous model can represent the system as a theoretical model [3]. Therefore the continuous model parameter estimation was used to begin the research. Some identification methods for the continuous time used the indirect approach via using the discrete-time model identification. This approach has the advantage of using the parameter estimation methods, but it requires extensive computation. Some of the identification methods are presented in this study.

An aircraft dynamics can change very rapidly and needs a more accurate model to control it. It is also desirable to avoid the use of special inputs for the identification.

In this study, an aircraft continuous model identification method in a real-time is developed and implemented. This implementation involved the integration of electronic components as well as a software simulation. This work is explained in the chapters as follows.

In the second chapter, the well known on-line discrete model identification algorithms are discussed. The transfer method from the discrete model to the continuous model and its accuracy is inquired with an example. The direct continuous model is reviewed via boundary value problem approaching.

In the third chapter, the aircraft dynamic is given to learn the parameters and its effect on the system. It is shown that the aircraft can not be controlled by only pilot unless auto pilot is engaged.

In the fourth chapter, the model reference adaptive system and self-tuning controller and their development are presented to control the system. The non-minimum phase and unstable systems are considered to optimize the adaptive control. A non-minimum aircraft adaptive control is given as an example.

In the fifth chapter, the continuous model aircraft identification is implemented for the different parameters values which are time constant and time variable. This method is used for the noise-mixed system as well as noise-free system. Its result for different noise level are given in this chapter. The gust response and its effect on the dynamic response and the identification are reviewed. All this operation are achieved as a real time simulation.

In the sixth chapter, the electronic circuits of the implementation are presented in detail. The aircraft is modelled by a computer, and it is identified by an identification board. The electronic circuit of the communication board between the model computer and the identification board is also given. In addition, other electronic circuit boards and their principals are given in detail.

In the seventh chapter, the model computer software and the identifier processor software are explained for subsequent use by other users. The necessary matrix operations and all programmes are discussed and are given full assemble code.

In the eighth chapter of this work, the result are described and some further studies are recommended.

CHAPTER 2: ON-LINE SYSTEM IDENTIFICATION

2.1. INTRODUCTION

In this chapter, on-line system identification and its methods are considered by different algorithms. An identification method may be classified as an "on-line" method if it satisfies the following criteria [4]:

(i) it must not require the application of a special input to the process in order that it can be used with the process under operation,

(ii) it does not require the storage of all the data

(iii) it uses a recursive algorithm so that one does not have to wait for the accumulation of large amounts of data to make the identification possible, but may start with an initial estimate of the parameters even after the first set of data has been obtained, and then keep on updating the estimate as more data arrives

(iv) the amount of computation required for each iteration of the recursive algorithm must be such that it can be carried out within one sampling interval.

The systems to be identified are divided into two types ; continuous-time and discrete time. In many practical situation: the identification of a continuous-time system is desired from samples of input-output data.

There are two approaches to the identification of a continuous-time-model. In the so-called indirect approach a discrete time model is obtained from samples, and then an equivalent continuous model is determined. The other approach attempts to obtain the continuous model directly.

Continuous-time systems are studied in later chapters. Although discrete time identification algorithms will also be mentioned because of the indirect method.

A continuous-time system is represented by the equation

$$\dot{x} = Ax + Bu$$

$$y = Cx + w(t)$$

where x is an (nx1) state variables vector, A is (nxn) coefficient vector, B is (nxm) control vector, C is transient vector, W(t) is noise.

If the assumption is that the inputs are held constant during the sampling interval, the discrete time model is

$$x(k+1) = Fx(k) + Gu(k)$$

The relationship between A,B and F,G will be explained later.

Firstly, discrete time algorithms are reviewed because the same algorithms can be used for continuous-time systems.

2.2. ON-LINE IDENTIFICATION METHODS FOR DISCRETE TIME MODELS

Many different identification and parameter estimation methods for dynamic processes have been described in the literature. The relationships between many of these methods are relatively well known as far as the theoretical background is concerned. Only some algorithms of those which are well known, are considered.

2.2.1. LEAST SQUARES

It is assumed that a linear process can be described by the model

$$y(k)+a_1 y(k-1)+\ldots+a_m y(k-m) = b_1 u(k-d-1)+\ldots+b_m u(k-d-m) \qquad (2.1)$$

respectively by

$$y_k a = u_{k-d} b$$

with

$$y_k = [\ y(k)\ y(k-1)\ \ldots\ y(k-m)\ ]$$

$$u_{k-d} = [\ u(k-d-1)\ u(k-d-2)\ \ldots u(k-d-m)\ ]$$

$$a^T = [\ 1\ a_1\ a_2\ \ldots\ a_m\ ]$$

$$b^T = [\ b_1\ b_2\ b_3\ \ldots b_m\ ]$$

or by the pulse transfer function

$$G_m = \frac{Y(z)}{U(z)} = \frac{B_m(z^{-1})}{A_m(z^{-1})} \tag{2.2}$$

$$= \frac{b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + \ldots + a_m z^{-m}}\ z^{-d} \tag{2.3}$$

Taking the measured input u(k) and measured output y(k) of the real process the generalized error used for parameter estimation is defined as

$$e(k) = y(k) + a_1 y(k-1) + \ldots + a_m y(k-m) - b_1 u(k-d-1) - \ldots - b_m u(k-d-m) \tag{2.4}$$

$$e(k) = y(k) - y_M(k) \tag{2.5}$$

where

$$y_M(k) = \Psi(k)\Theta \tag{2.6}$$

is the prediction of y(k) of the model based on the observation y(k-m), ... , y(k-1) with

$$\Psi(k) = [\ -y(k-1)\ \ldots\ y(k-m)\ \vdots\ u(k-d-1)\ \ldots\ u(k-d-m)\ ]$$

$$\Theta^T = [\ a_1\ a_2\ \ldots\ a_m\ \vdots\ b_1\ \ldots\ b_m\ ]$$

Minimising the cost function

$$V = \sum_{k=m+d}^{N+m+d} e^2(k) = \sum_{k=m+d}^{N+m+d} (y(k) - \Psi(k)\Theta)^2 \qquad (2.7)$$

and using the notation

$$y^T = [y(m+d) \; y(m+d+1) \; \ldots \; y(m+d+N)]$$

$$\Psi = \left[\begin{array}{cccc|cccc} -y(m+d-1) & -y(m+d-2) & \ldots & -y(d) & u(m-1) & u(m-2) & \ldots & u(0) \\ -y(m+d) & -y(m+d-1) & \ldots & -y(d+1) & u(m) & u(m-1) & \ldots & u(1) \\ . & & & . & . & & & . \\ . & & & . & . & & & . \\ . & & & . & . & & & . \\ -y(m+d+N-1) & \ldots & \ldots & -y(d+n) & u(m+N-1) & \ldots & \ldots & u(N) \end{array} \right]$$

The cost function can be rewritten as

$$V = (y - y_M)^T (y - y_M) \qquad (2.8)$$

$$= (y - \Psi\Theta)^T (y - \Psi\Theta)$$

$$= y^T y - y^T\Psi\Theta - \Theta^T\Psi^T y + \Theta^T\Psi^T\Psi\Theta \qquad (2.9)$$

Since the matrix $\Psi^T\Psi$ is always nonnegative definite, the function V has a minimum. The loss function is quadratic in Θ. By completing the square, it is possible to find the minimum.

$$V = y^T y - y^T\Psi\Theta - \Theta^T\Psi^T y + \Theta^T\Psi^T\Psi\Theta$$

$$+ y^T\Psi(\Psi^T\Psi)^{-1}\Psi^T y - y^T\Psi(\Psi^T\Psi)^{-1}\Psi^T y$$

$$= y^T(I-\Psi(\Psi^T\Psi)^{-1}\Psi^T)y + (\Theta-(\Psi^T\Psi)^{-1}\Psi^T y)\Psi^T\Psi(\Theta-\Psi(\Psi^T\Psi)^{-1}\Psi^T y)$$

The first term on the right-hand side is independent of Θ. The second term is always positive. The minimum is obtained for

$$\Theta = \hat{\Theta} = (\Psi^T\Psi)^{-1}\Psi^T y \qquad (2.10)$$

$$P = (\Psi^T\Psi)^{-1} \qquad (2.11)$$

$$\hat{\Theta} = P\,\Psi^T\,y \qquad (2.12)$$

For recursive computations,

$$P^{-1}(k+1) = P^{-1}(k) + \Psi(k+1)\Psi^T(k+1) \qquad (2.13)$$

The least-squares estimate $\Theta(k+1)$ is given by

$$\hat{\Theta}(k+1) = P(k+1)\left(\sum_{i=1}^{k+1}\Psi(i)y(i)\right) \qquad (2.14)$$

$$= P(k+1)\left(\sum_{i=1}^{k}\Psi(i)y(i) + \Psi(k+1)y(k+1)\right)$$

$$\sum_{i=1}^{k}\Psi(i)y(i) = P^{-1}(k)\,\hat{\Theta}(k)$$

$$= \left(P^{-1}(k+1) - \Psi(k+1)\Psi^T(k+1)\right)\hat{\Theta}(k) \qquad (2.15)$$

The estimate at moment(k+1) can now be written as

$$\hat{\Theta}(k+1) = \hat{\Theta}(k) - P(k+1)\Psi(k+1)\Psi^T(k+1) + P(k+1)\Psi(k+1)y(k+1)$$

$$= \hat{\Theta}(k) + \hat{P}(k+1)\Psi(k+1)\left(y(k+1) - \Psi^T(k+1)\hat{\Theta}(k) \right) \quad (2.16)$$

Applying matrix inversion lemma to P(k+1) and using Eq(2.13) gives [5]

$$P(k+1) = P(k)-P(k)\Psi(k+1)\left[I+\Psi^T(k)P(k)\Psi(k+1)\right]^{-1}\Psi^T(k+1)P(k) \quad (2.17)$$

$$\hat{\Theta}(k+1) = \Theta(k) + P(k)\Psi(k+1)\left(I + \Psi^T(k+1)P(k)\Psi(k+1) \right)^{-1} \quad (2.18)$$

2.2.2. GENERALIZED LEAST SQUARES

The generalized least square algorithm was introduced by CLARKE (1967)[6]. It is developed from least squares for the case when white noise is present. Algorithm for on-line generalized least squares estimation have been developed by HASTING-JAMES and SAGE (1969)[7].The system to be explained for this algorithm is shown in Fig.2.1.

The noise-free output y(k) of the process at any time is given as a weighted sum of the past process outputs,

y(k-1), y(k-2), ..., y(k-n)

and past process inputs

$$u(k-l-1),\ u(k-l-2),\ \ldots,\ u(k-l-n)$$

where n is the order of the process, and l represents the pure time delay of the process in terms of an integral number of sampling intervals.

Thus

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_n y(k-n)$$
$$+ b_1 u(k-l-1) + b_2 u(k-l-2) + \ldots + b_n u(k-l-n)$$

The shifting operator z is defined as seen below

$$z^{-1} y(k) = y(k-1)$$

The process transfer function can be written ;

$$[\ 1 + A(z^{-1})\ ]y(k) = z^{-k}\ B(z^{-1})u(k) \qquad (2.19)$$

The output from the noise process filter ε_k at any time k is given by a weighted sum of past outputs.

$$\varepsilon(k-1),\ \varepsilon(k-2),\ \ldots\ ,\ \varepsilon(k-p)$$

and past and present inputs to the filter

$$\xi(k),\ \xi(k-1),\ \xi(k-2),\ \ldots\ ,\xi(k-p)$$

Thus as above

$$[\ 1 + F(z^{-1})\ ]\ \varepsilon(k) = [\ 1 + G(z^{-1})\ ]\ \xi(k)$$

The measured process output is given by

$$v(k) = y(k) + \varepsilon(k)$$

and therefore from Eq(2.19)

$$[\ 1 + A(z^{-1})]v(k) = z^{-k}\ B(z^{-1})u(k) + e(k) \qquad (2.20)$$

where

$$e(k) = \frac{[\ 1 + A(z^{-1})\ ]\ [\ 1 + G(z^{-1})\ ]}{[\ 1 + F(z^{-1})\ ]}\ \xi(k) \qquad (2.21)$$

In order to write the process equation (2.20) in the standard vector-matrix notation of regression analysis, consider a sequence of observation { u, v } to have been made of the process input and process output variables. Then

$$v(k) = -a_1 v(k-1) - a_2 v(k-2) - \dots - a_n v(k-n)$$
$$+ b_1 u(k-l-1) + \dots + b_n u(k-l-n) + e(k)$$

$$v(k+1) = -a_1 v(k) - a_2 v(k-1) - \dots - a_n v(k-n+1)$$
$$+ b_1 u(k-l) + \dots + b_n u(k-l-n+1) + e(k+1)$$

.

.

$$v(k+N) = -a_1 v(k+N-1) - a_2 v(k+N-2) - \ldots - a_n v(k+N-n)$$
$$+ b_1 u(k+N-l-1) + \ldots + b_n u(k+N-l-n) + e(k+N)$$

This can be written

$$v = X\Theta + e \tag{2.21}$$

where

$$X = [-z^{-1}v \,\vdots\, -z^{-2}v \,\vdots \ldots \vdots\, -z^{-n}v \,\vdots\,\vdots\, z^{-1-l}u \,\vdots\, z^{-2-l}u \,\vdots \ldots \vdots\, z^{-n-l}u]$$

$$\Theta^T = [\ a_1\ a_2\ a_3 \ldots a_n\ b_1\ b_2 \ldots\ b_n\ ]$$

and v, u and e are the output, input and noise vector, respectively. The jth element of each vector represents the sampled value of the respective variable at the k+jth sampling instant.

The residual errors are now minimized between the measured plant output and the output predicted by the model over the set of data being considered. On taking a cost function (see Fig.2.1.) $\eta^T\eta$ and minimizing with respect to parameter vector Θ, the least squares estimate of Θ is given by

$$\hat{\Theta}_{LS} = [\ X^T X\ ]^{-1} X^T v \tag{2.22}$$

This estimate of Θ can be shown to be biased, for since

$$v = y + \varepsilon$$

Eq(2.22) leads to

$$\hat{\Theta}_{LS} = [\ X^TX\ ]^{-1}X^Ty + [\ X^TX\ ]^{-1}X^Te \qquad (2.23)$$

From Fig.2.1.

$$y = X\Theta \qquad (2.24)$$

so eq(2.23) reduced to

$$\Theta_{LS} = \Theta + [\ X^TX\ ]^{-1}X^Te \qquad (2.25)$$

showing that the bias on the least-squares estimate of Θ is given by $E\{\ [X^TX]^{-1}X^Te\ \}$. E the is expectation operator. Therefore noise parameters are sought by system parameters.

The prediction error can be written by

$$\hat{e} \cong \eta = v - X\hat{\Theta} \qquad (2.26)$$

The Eq(2.26) is a valid approach, since in limit, as $\hat{\Theta}$ approaches Θ, η approaches the corrupting noise e. The corrupting noise e is desired to represent by a autoregressive model. The model used is

$$\Big[\ 1 + C(z^{-1})\ \Big]\ e(k) = \xi(k) \qquad (2.27)$$

Thus, the assumption is made that

$$\left[1 + C(z^{-1}) \right] \cong \left[\frac{[1 + A(z^{-1})][1 + G(z^{-1})]}{[1 + F(z^{-1})]} \right]^{-1} \tag{2.28}$$

Eq(2.27) is rewritten in a vector matrix form as

$$e = - \mathbf{E}C + \xi \tag{2.29}$$

where

$$\mathbf{E} = [z^{-1}e \,\vdots\, z^{-2}e \,\vdots\, \dots \,\vdots z^{-m}e]$$

and

$$C^T = [C_1 \,\vdots\, C_2 \,\vdots\, \dots \,\vdots C_m]$$

The least-square estimate of c is given by

$$C_{LS} = - [\mathbf{E}^T\mathbf{E}]^{-1} \mathbf{E}^T e \tag{2.30}$$

This estimate is unbiased, as the element of **E** are independent of ξ, the uncorrelated error term Eq(2.29).

This estimation of noise parameters allows the desired transformation of the data u and v to be made which leads in the limit, as $\hat{C}$ approaches C, to an unbiased estimate of Θ. This can be seen by replacing the elements of u and v by

$$u^F(k) = [1 + C(z^{-1})] u(k) \tag{2.31}$$

$$v^F(k) = [\ 1 + C(z^{-1})\ ]\ v(k)$$

Eq(2.19) can be written using Eq(2.30),

$$[\ 1 + A(z^{-1})\ ]v^F(k) = z^{-k}\ B(z^{-1})u^F(k) + \xi(k) \qquad (2.32)$$

because of Eq(2.29).

Eq(2.32) is rewritten in vector form

$$v^F = X^F\Theta + \xi \qquad (2.33)$$

where X^F is the regression matrix of filtered data, is written in notation form by

$$X^{FT} = [\ -z^{-1}v^F \vdots -z^{-2}v^F \vdots \dots \vdots -z^{-n}v^F \vdots \vdots z^{-1-l}u^F \vdots \dots \vdots z^{-n-l}u^F\ ]$$

An unbiased estimate of Θ is now available as

$$\Theta = [\ X^{FT}X^F]^{-1}\ X^{FT}v^F \qquad (2.34)$$

Thus, by obtaining a least-square estimate of Θ to start the procedure with Eq(2.22) and then iterating between the estimates of the noise and process parameters $\hat{C}$ and $\hat{\Theta}$, using Eqs(2.26), (2.30), (2.31) and (2.34), unbiased estimates of the system parameters can be obtained. The iteration is continued until the minimum error is achieved.

u^F and v^F are described as

$$u^F_{N+1} = [\ 1 + C(z^{-1})\ ]\ u_{N+1} \tag{2.35}$$

$$v^F_{N+1} = [\ 1 + C(z^{-1})\ ]\ v_{N+1}$$

for recursive estimation.

The effect of the new data on the present process parameter estimate $\hat{\Theta}$ can be seen by writing Eq(2.33) in the partitioned form.

$$\begin{bmatrix} v^F \\ u^F_{N+1} \end{bmatrix} = \begin{bmatrix} X^F \\ \chi^{FT}_{N+1} \end{bmatrix} \Theta + \begin{bmatrix} \xi \\ \xi_{N+1} \end{bmatrix} \tag{2.36}$$

$$\chi^{FT}_{N+1} = [\ -v^F_N \ \vdots -z^{-1}v^F_N \ \vdots \ \ldots \vdots -z^{-n}v^F_N \ \vdots\vdots\ z^{-1}\ u^F_N \ \vdots \ldots \vdots\ z^{-n-1}u^F_N\ ]$$

$$\hat{\Theta}_{N+1} = \left(\begin{bmatrix} X^F \\ \chi^{FT}_{N+1} \end{bmatrix}^T \begin{bmatrix} X^F \\ \chi^{FT}_{N+1} \end{bmatrix} \right)^{-1} \begin{bmatrix} X^F \\ \chi^{FT}_{N+1} \end{bmatrix}^T \begin{bmatrix} v^F \\ v^{FT}_{N+1} \end{bmatrix} \tag{2.37}$$

$$[\ X^TX + \chi\chi^T\ ] = [\ X^TX\ ]^{-1} - \frac{[\ X^TX\ ]^{-1}\ \chi\chi^T\ [\ X^TX\ ]}{1 + \chi^{FT}\ [\ X^TX\ ]^{-1}\chi} \tag{2.38}$$

Eq(2.38) is written according to Hasting-Sage (1969)[7]. (2.37) is rewritten with using (2.38),

$$\hat{\Theta}_{N+1} = \hat{\Theta}_N + \frac{[\,X^{FT}X\,]^{-1}\,\chi^{F}_{N+1}\,(\,v^{F}_{N+1} - \chi^{FT}_{N+1}\hat{\Theta}_N\,)}{1 + \chi^{FT}_{N+1}\,[\,X^{FT}X^{F}\,]^{-1}\,\chi^{F}_{N+1}} \tag{2.39}$$

Equations (2.26) and (2.29) are also rewritten as

$$\hat{e}_{N+1} = v_{N+1} - \chi^{T}_{N+1}\hat{\Theta}_{N+1} \tag{2.40}$$

$$\begin{bmatrix} \hat{e} \\ e^{T}_{N+1} \end{bmatrix} = - \begin{bmatrix} E \\ \varepsilon^{T}_{N+1} \end{bmatrix} C + \begin{bmatrix} \xi \\ \xi_{N+1} \end{bmatrix} \tag{2.41}$$

with $\varepsilon^{T}_{N+1} = [\; e_N \;\vdots\; e_{N-1} \;\vdots\; \dots \;\vdots\; e_{N-M+1} \;]$.

The recursive estimation noise parameters can be described by using (2.38).

$$\hat{C}_{N+1} = \hat{C}_N - \frac{[\,E^{T}E\,]^{-1}_{N}\,\varepsilon_{N+1}\,(\,e_{N+1} + \varepsilon^{T}_{N+1}\hat{C}_N\,)}{1 + \varepsilon^{T}_{N+1}\,[\,E^{T}E\,]^{-1}_{N}\,\varepsilon_{N+1}} \tag{2.42}$$

The inverse of matrices is obtained using equation (2.38)

$$[\,X^{FT}X^{F}\,]^{-1}_{N+1} = [\,X^{FT}X^{F}\,]^{-1}_{N} - \frac{[\,X^{FT}X^{F}\,]^{-1}_{N}\chi^{F}_{N+1}\chi^{FT}_{N+1}[\,X^{FT}X^{F}\,]^{-1}_{N}}{1 + \chi^{FT}_{N+1}[\,X^{FT}X^{F}\,]^{-1}\,\chi^{F}_{N+1}} \tag{2.43}$$

$$[\,E^{T}E\,]^{-1}_{N+1} = [\,E^{T}E\,]^{-1}_{N} - \frac{[\,E^{T}E\,]^{-1}\varepsilon_{N+1}\varepsilon^{T}_{N+1}[\,E^{T}E\,]^{-1}_{N}}{1 + \varepsilon^{T}_{N+1}\,[\,E^{T}E\,]^{-1}_{N}\,\varepsilon_{N+1}} \tag{2.44}$$

Hasting and Sage have developed Eqs(2.39) and (2.42) by using an exponential weighting factor. This equations have been changed with the weighting factor as seen below,

$$\hat{\Theta}_{N+1} = \hat{\Theta}_N + \frac{[\, X^{FT}X \,]^{-1}_{N+1}\chi_{N+1} \,(\, v^F_{N+1} - \chi^{FT}_{N+1}\hat{\Theta}_N \,)}{\rho + \chi^{FT}_{N+1} \,[\, X^{FT}X^F \,]^{-1}_{N+1}\chi^F_{N+1}} \tag{2.45}$$

$$\hat{C}_{N+1} = \hat{C}_N - \frac{[\, E^TE \,]^{-1}_N \,\varepsilon_{N+1} \,(\, e_{N+1} + \varepsilon^T_{N+1}\hat{C}_N \,)}{\delta + \varepsilon^T_{N+1} \,[\, E^TE \,]^{-1}_N \,\varepsilon_{N+1}} \tag{2.46}$$

where $0 < \rho < 1$ and $0 < \delta < 1$.

The inverse matrices are changed with the same weighting factors.

$$[\, X^{FT}X^F \,]^{-1}_{N+1} = \frac{1}{\rho}\left([\, X^{FT}X^F \,]^{-1}_N - \frac{[\, X^{FT}X^F \,]^{-1}_N\chi^F_{N+1}\chi^{FT}_{N+1}[\, X^{FT}X^F \,]^{-1}_N}{\rho + \chi^{FT}_{N+1}[\, X^{FT}X^F \,]^{-1}\chi^F_{N+1}} \right) \tag{2.47}$$

$$[\, E^TE \,]^{-1}_{N+1} = \frac{1}{\delta}\left([\, E^TE \,]^{-1}_N - \frac{[\, E^TE \,]^{-1}\varepsilon_{N+1}\varepsilon^T_{N+1}[\, E^TE \,]^{-1}}{\delta + \varepsilon^T_{N+1} \,[\, E^TE \,]^{-1} \,\varepsilon_{N+1}} \right) \tag{2.48}$$

Eqs (2.35), (2.36), (2.47), (2.45), (2.40), (2,41), (2.48) and (2.46) are used for recursive estimate iterations.

2.2.3. ON-LINE MAXIMUM LIKELIHOOD METHOD

The on-line maximum likelihood identification method was described by Aström and Bohlin [8]. It gives good estimates even when the noise level is quite high for off-line identification. Gertler and Banyasiz [9] developed an algorithm based upon maximum likelihood method for on-line identification.

The process model is assumed to be as in Fig.2.2.. Input-output relationship is thought to be as below

$$M(z) = \frac{1}{\left(1 + H(z^{-1})\right)\left(1 + D(z^{-1})\right)}$$

therefore

$$y(k) = \frac{G(z^{-1})}{1 + H(z^{-1})} u(k) + \frac{1}{\left(1 + H(z^{-1})\right)\left(1 + D(z^{-1})\right)} e(k)$$

G, H and D are polynomials.

The parameter vector is now

$$\Theta^T = [\ g^T\ h^T\ d^T\ ]$$

The noise is obtained

$$e(k) = [\ 1 + D(z^{-1})\ ]\ \{\ [\ 1 + H(z^{-1})\ ]y(k) - G(z^{-1})u_k\ \}$$

The partial derivatives of the noise with respect to the various parameters are given by

$$\frac{\partial e(k)}{\partial g_j} = - [\ 1 + D(z^{-1})\ ]\ z^{-j}\ u(k) = -u^F(k-j)$$

$$\frac{\partial e(k)}{\partial h_j} = [\ 1 + D(z^{-1})\ ]\ z^{-j}\ y(k) = y^F(k-j)$$

$$\frac{\partial e(k)}{\partial d_j} = z^{-j} \Big([\ 1 + H(z^{-1})\ ]y(k) - G(z^{-1})u(k) \Big) = -\mu(k-j)$$

where

$$u^F(k) = [\ 1 + D(z^{-1})\ ]\ u(k)$$

$$y^F(k) = [\ 1 + D(z^{-1})\ ]\ y(k)$$

and

$$\mu(k) = G(z^{-1})u(k) - [\ 1 + H(z^{-1})\ ]\ y(k)$$

These derivatives are generated by moving-average filtering and shifting. The vector of the first noise derivatives is built up as

$$\underline{V}(k) = \begin{bmatrix} -u^F(n+1) \\ y^F(n) \\ -u(n) \end{bmatrix}$$

where

$$\underline{u}^F(n+1) = \begin{bmatrix} u^F(k) \\ u^F(k-1) \\ . \\ u^F(k-n) \end{bmatrix}, \quad \underline{y}^F(n) = \begin{bmatrix} y^F(k+1) \\ . \\ . \\ y^F(k-n) \end{bmatrix}, \quad \underline{u}(n) = \begin{bmatrix} u(k-1) \\ u(k-2) \\ . \\ . \\ u(k-n) \end{bmatrix}$$

The nonzero second order derivatives are

$$\frac{\partial^2 e(k)}{\partial g_j \partial d_m} = -z^{-(j+m)}u(k) = -u(k-j-m)$$

$$\frac{\partial^2 e(k)}{\partial h_k \partial d_m} = z^{-(k+m)}y(k) = y(k-j-m)$$

The second derivatives are generated simply by double shifting. The matrix of the second derivatives is

$$W(k) = \begin{bmatrix} 0 & X(k) \\ X^T(k) & 0 \end{bmatrix}$$

where

$$X(k) = \begin{bmatrix} -U(k-1)_{n,n+1} \\ Y(k-2)_{n,n} \end{bmatrix}$$

and the double subscript indicates the respective number of previous consecutive values in the columns and rows of the matrix.

The on-line equivalent of the iterative equation is

$$\hat{\Theta}(k) = \hat{\Theta}(k-1) - R^{-1}(k,n,\hat{\Theta}(k-1))\ q(k,n,\hat{\Theta}(k-1))$$

where

$$q(k,N,\hat{\Theta}(k-1)) = \left(\frac{\partial}{\partial\Theta^T} \left(e^T(k)_N e(k)_N\right) \right)^T_{\Theta=\hat{\Theta}(k-1)}$$

$$R(k,N,\hat{\Theta}(k-1)) = \left| \frac{\partial}{\partial\Theta} \; q(k,N) \right|_{\Theta=\hat{\Theta}(k-1)}$$

N indicates a vector of N previous consecutive values. Introducing conditional arguments for the sake of brevity

$$\hat{\Theta}(k) = \hat{\Theta}(k-1) - R^{-1}(k;k-1,N)q(k;k-1,N) \qquad (2.49)$$

$$q(k;k-1,N) = q(k-1;k-2,N-1) + V(k)e(k) \qquad (2.50)$$

$$R(k;k-1,N) = R(k-1;k-2,N-1) + V(k)V^T(k) + W(k)e(k) \qquad (2.51)$$

Introducing a filtering factor λ ($0 < \lambda < 1$) to suppress exponentially previous measurements, the last equations are rewritten as

$$q(k;k-1,N) = \lambda q(k-1;k-2,N-1) + V(k)e(k) \qquad (2.52)$$

$$R(k;k-1,N) = \lambda R(k-1;k-2,N-1) + V(k)V^T(k) + W(k)e(k) \qquad (2.53)$$

We may use matrix inversion lemma to obtain the inverse of matrix in Eq(2.49);

$$R^{-1}(k;k-1,N) = R_I^{-1}(k;k-1,N)\left(I - W(k)e(k)R_I^{-1}(k;k-1,N) \right) \qquad (2.54)$$

where

$$R_I^{-1} = \frac{1}{\lambda} R^{-1}(k-1;k-2,N-1) - \frac{R^{-1}(k-1;k-2,N-1)V(k)V^T(k)R^{-1}(k-1;k-2,N-1)}{\lambda + V^T(k)R^{-1}(k-1;k-2,N-1)V(k)} \tag{2.55}$$

Eqs(2.49),(2.52),(2.55) and (2.54) form is an on-line algorithm for maximum likelihood identification.

2.2.4. INSTRUMENTAL VARIABLE METHOD

This method is nearly the ordinary least-squares method. The considered system equation is taken as well as other methods.

$$x(k) + a_1 x(k-1) + \ldots + a_m x(k-m) = b_1 u(k-1) + \ldots + b_m u(k-m) \tag{2.56}$$

The measured output is

$$y_k = \Psi_k \Theta + v_k \tag{2.57}$$

Eq(2.57) becomes to Eq(2.58) by premultiplying both side of the equation with W_k^T matrix,

$$W_k^T y_k = W_k^T \Psi_k \Theta + W_k^T v_k \tag{2.58}$$

where W is called instrument matrix which satisfies

$$E[\, W_k^T v_k \,] = 0 \tag{2.59}$$

$E\,[\,W^T_k\,\Psi_k\,]$ is nonsingular.

The elements of W_k therefore are chosen to be uncorrelated with residuals v_k . Then from Eq(2.57),

$$\hat{\Theta}_k = (\,W_k^T\Psi_k\,)^{-1}\,W_k^T y_k \qquad (2.60)$$

The main problem of this method is that of finding W_k matrix. The method proposed by Wong, Polak [10] and Young [11]. This is illustrated in Fig. 2.3.

It consists of taking the instrumental variables as the disturbed output, h_k, of an auxiliary model to which the same input, u_k, is applied. This { h_k } will be correlated with { u_k } but uncorrelated with { n_k } and, therefore, with { v_k }. The matrix W_k then takes the form,

$$W_k = \begin{bmatrix} u_0 & u_{-1}\cdots & u_{1-m} & -h_d & -h_{d-1} & \cdots & -h_{d+1-n} \\ u_1 & u_0 & u_{2-m} & -h_{d+1} & -h_d & \cdots & -h_{d+2-n} \\ \vdots & & & & & & \\ u_{k-1} & u_{k-2} & u_{k-3} & -h_{d+k-1} & -h_{d+k-2} & & -h_{d+k-n} \end{bmatrix} \qquad (2.61)$$

A recursive algorithm of Eq(2.60) is obtained analogous to Eqs(2.17) and (2.18).

$$\hat{\Theta}(k+1) = \Theta(k) + [\;\Psi(k+1)P(k)w^T(k+1) + 1\;]^{-1}$$
$$*\;p(k)w^T(k+1)\;[\;y(k+1) - \Psi(k+1)\hat{\Theta}(k)] \qquad (2.62)$$

$$P(k+1) = P(k) \left(I-\Psi (k)w^T(k+1)P(k) \left[\Psi(k+1)P(k)w^T(k+1)+1\right]^{-1}\right) \tag{2.63}$$

$$P(k)= \left[W(k)\Psi(k) \right]^{-1} \tag{2.64}$$

$$w(k) = [u_{k-1} \; u_{k-2} \ldots u_{k-m} \;\; -h_{k-d-1} \;\; -h_{k-d-2} \ldots \;\; -h_{k-d-n}] \tag{2.65}$$

Then, Young (1972) [12] introduced a time delay and low-pass filter before updating the auxiliary model, to ensure that the auxiliary model parameters are not correlated with v(k) at the same instant and to smooth the estimates. This low-pass may be of the form

$$\hat{\Theta}_{aux}(k) = (1 - \nu) \hat{\Theta}_{aux}(k-1) + \nu \hat{\Theta}(k) \tag{2.66}$$

where ν = 0.03 to 0.05 has to be chosen to prevent instability in the estimation. The initial matrix P(0) can be chosen as a diagonal matrix with elements as large as possible. The initial values of the parameter vector $\hat{\Theta}(0)$ of the model and of $\hat{\Theta}(0)_{aux}$ of the auxiliary model can be zero. A new block diagram of the method is shown in Fig.2.4.

2.2.5. IDENTIFICATION OF LINEAR MULTIVARIABLE SYSTEMS

A multivariable system can be represented by some types of models. Generally, the state-space model is considered, because it may be transformed to the continuous time model in the next step. The state-space model can be definition by the equations

$$\underline{x}(k+1) = F\,\underline{x}(k) + G\,\underline{u}(k)$$
$$\underline{y}(k) = H\,\underline{x}(k) \qquad (2.67)$$

where $\underline{x}(k)$ is n dimensional state vector F, G and H are $n\times n$, $n\times m$ and $p\times n$ constant matrices, respectively.

The total number of the parameters of the matrices F, G, and H are

$$N = n\ (n+m+p)$$

The identification of the matrices F, G and H are not unique. Several canonical forms of the state-space have been developed for the identification purpose. The linear multivariable system identification in the state-space form is much more difficult. Budin (1971) [13] has proposed a method for the realization of the system with minimum calculation. The system equation is rewritten for explanation of this method.

$$x\ (k+1) = [\ F,\ G\ ] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \qquad (2.68)$$
$$y\ (k) = H\ x\ (k)$$

If H is identity matrix y(k)=x(k), equation (2.68) becomes

$$y\,(k+1) = [\ F,\ G\ ]\begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \tag{2.69}$$

F and G can be obtained only if both sides' dimensions are the same. In this condition equation(2.69) can be rewritten

$$\sum_{k=1}^{N} [y(k+1)\ \dots\ y(k+n+m)] = [F,G]\sum_{k=1}^{N}\begin{bmatrix} y(k) & \dots & y(k+n+m-1) \\ u(k) & \dots & u(k+n+m-1) \end{bmatrix} \tag{2.70}$$

F and G that can be solved equation(2.70), are not state-vector. They are for input-output form. Firstly, the selector matrix must be introduced for state-space form.

A selector matrix S is a (k x ℓ) matrix ($k \le \ell$) with the property that when multiplying an (ℓ x m) matrix A the resulting (k x m) matrix SA consist of k of the rows of A ordered as they are in A. From this definition it follows that;

1) $s_{ij} = 0$ or 1, $\forall i, j$;

2) $\forall i$, there is one only one value of j, j_i such that

   $s_{ij_i} = 1$

3) $j_1 < j_2 < \dots < j_k$.

It also follows that if a $p \times q$ matrix R has rank m then there are two selector matrices S_1 (an $m \times p$) and S_2 (an $m \times q$) such that $S_1RS_2^T$ is nonsingular. If $m = q$ then $S_2 = I$ and if $m = p$ then

$S_1 = I$.

The notation $S(i_1, i_2, \ldots, i_k)$ is used to specify the selector matrix which deletes the rows $i_1, i_2, \ldots, i_k$ from the matrix it multiplies.

Budin has also shown that for a completely observable system there exists an $n \times n^* p$ ($n^* = n$ - rank of H + 1) selector matrix, S, such that

$$SW = T \tag{2.71}$$

is nonsingular, where W is the observability matrix of the system. It can be assumed that T = I since a linear transformation of the state vector will not effect the input-output description.

Using any selector matrix that satisfies equation(2.71), the following direct input-output relation for a completely observable system is obtained.

$$S\bar{y}_{n^*}(k+1) = [\ F, R\ ] \begin{bmatrix} S\bar{y}_{n^*}(k) \\ \bar{u}_{n^*}(k) \end{bmatrix} \tag{2.72}$$

where

$$\bar{y}_{n^*}(k) = [\ y^T(k),\ y^T(k+1),\ \ldots\ ,\ y^T(k+n^*-1)\ ] \tag{2.73}$$

$$\bar{u}_{n^*}(k) = [\ u^T(k),\ u^T(k+1),\ \ldots\ ,\ u^T(k+n^* - 1)\ ] \qquad (2.74)$$

$$R = -FSS(pn^* + 1,\ pn^* + 2,\ \ldots,\ pn^* + p)\, R_{n^*} + SS(1,\ 2,\ \ldots,\ p)\, R_{n^*}$$

(2.75)

$$R_{n^*} = \begin{bmatrix} 0 & 0 & & 0 \\ HF & 0 & & 0 \\ HFG & HG & & 0 \\ \cdot & \cdot & & \\ \cdot & \cdot & & 0 \\ HF^{n^*-1}G & & HFG & HG \end{bmatrix} \qquad (2.76)$$

Using equation(2.72), we have

$$S\,[\ y_{n^*}(k+1)\ \ldots\ y_{n^*}(k+n+mn^*)\ ] = [F,R] \begin{bmatrix} S & \vdots & 0 \\ & & \\ 0 & \vdots & I_{mn^*} \end{bmatrix} \begin{bmatrix} y_{n^*}(k)\ldots y_{n^*}(k+n+mn^*-1) \\ \\ u_{n^*}(k)\ldots u_{n^*}(k+n+mn^*-1) \end{bmatrix}$$

(2.77)

which can be written more compactly as

$$SA_{n^*}(k+1) = [\ F\ \vdots\ R\ ]\ \mathcal{S}B_{n^*}(k) \qquad (2.78)$$

where the correspondence is obvious. $\mathcal{S}$ is an $(mn^* + n) \times (m + p)n^*$ selector matrix. A unique solution for $[F\ \vdots\ R]$ exits whenever

$\mathcal{S}B_{n^*}(k)$ is nonsingular. Budin has proved that $B_{n^*}(k)$ must have rank $n + mn^*$ for $\mathcal{S}G_{n^*}(k)$ to be nonsingular.

This method is applied using the following steps :

1) Construct the $B_{N^*}(k)$ matrix, where $N^* = N+1$ - rank of H, and N is upper upper bound of minimal dimension. $B_{N^*}(k)$ is given by

$$B_{N^*}(k) = \begin{bmatrix} \underline{u}(k) & \cdots\cdots & \underline{u}(k+n+mN^*-1) \\ \underline{y}(k) & \cdots\cdots & \underline{\dot{y}}(k+n+mN^*-1) \\ \vdots & \vdots & \\ \underline{u}(k+N^*-1) & \cdots & \underline{u}(k+N+mN^*+N^*-2) \\ \underline{y}(k) & \cdots\cdots & \underline{y}(k+n+mN^*+N^*-2) \end{bmatrix}$$

2) Obtain $\mathfrak{S}_1[\ B_{N^*}\ (k)\ ]$ The mN^* rows of $B_{N^*}(k)$ consisting of input observations will be among the independent rows, and m of the first of pn^* rows of $B_{N^*}(k)$ consisting of output observations will complete the set of the independent rows.

3) Determine the order of the minimal realization is given by

n = number of the independent rows of $\mathfrak{S}_1[B_{N^*}(k)] - mN^*$

4) Construct the $(pn^* \times n)$ submatrix K of $\mathfrak{S}_1[B_{N^*}(k)]$ consisting of the first pn^* output rows and first n columns not containing 1's associated input rows.

$S = K^T$

5) Construct the matrices $A_{n^*}(k+1)$, $B_{n^*}(k)$ and $\mathcal{S}$ and obtain F from the equation.

$[\ F \ \vdots\ R\ ] = SA_{n^*}(k+1)\ [\ \mathcal{S}B_{n^*}(k)\ ]^{-1}$

6) Obtain G from the equation

$$G = R_0 + F R_1 + \ldots + F^{n^*-1} R_{n^*-1}$$

$$R = [\, R_0 \; R_1 \ldots R_{n^*-1}]$$

2.3. ON-LINE IDENTIFICATION METHODS FOR CONTINUOUS-TIME MODEL

2.3.1. INDIRECT METHODS

To obtain a continuous-time model from discrete-time, equivalents have been introduced for economical research. The indirect method via discrete-time model identification has also attracted some attention in control theory. Sinha (1973) [14] has obtained a continuous-time model from the equivalent discrete-time model by using bilinear z transform. More recently Sinha and Lastman (1981) [15] have proposed a transformation algorithm from discrete-time model to continuous time model.

A linear continuous-time system is described by the equation,

$$\dot{x} = Ax + Bu$$

$$y = Cx + w(t) \qquad (2.79)$$

The identification of A, B and C are determined samples u(kT) and y(kT) of the measured input and output vectors. The equivalent discrete-time model of Eq(2.67) is given by

$$\begin{aligned} x(k+1) &= Fx(k) + Gu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.80}$$

where

$$F = e^{AT} = I + AT + \frac{1}{2!}(AT)^2 + \frac{1}{3!}(AT)^3 + \ldots. \tag{2.81}$$

$$G = \int_0^T e^{AT}B \, dt = \left(IT + \frac{1}{2!}AT^2 + \frac{1}{3!}A^2T^3 + \ldots \right) B \tag{2.82}$$

It will be assumed that the sampling rate is satisfactory and the spectral norm of AT satisfied the following inequality;

$$| \lambda T | \le 0.5 \tag{2.83}$$

where λ is the eigenvalue of the continuous-time system farthest away from the origin of the complex plane.

The Sinha & Lastman transformation algorithm which was mentioned, is given by

$$\begin{aligned} (AT)^{(k+1)} &= (AT)^k + F^{-1} (F - F^{(k)}) \\ &= (AT)^k + I - F^{-1} F^{(k)} \end{aligned} \tag{2.84}$$

where $(AT)^k$ is the value of AT at k th iteration and

$$F^{(k)} = e^{(AT)^{(k)}} \qquad (2.85)$$

which can be calculated using the power series similar to Eq(2.81). This method requires that the spectral radius of AT and F be less than one. Eq(2.83) is enough for both this conditions.

Initial guess can be taken

$$(AT)^{0} = \frac{1}{2} (F - F^{-1}) \qquad (2.86)$$

After AT has been obtained, the matrix b can be obtained from the relationship

$$B = R^{-1}G \qquad (2.87)$$

where

$$R = [I + \frac{1}{2!} AT + \frac{1}{3!} (AT)^{2} + \dots] T \qquad (2.88)$$

This algorithm is best shown by an example;

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{bmatrix} \qquad \text{and} \quad T = 0.1$$

$$\lambda_{1} = -0.22324$$

$$\lambda_{2,3} = -0.0338 \mp j0.0562$$

$$F = e^{AT} = \begin{bmatrix} 0.99984527 & 0.09968661 & 0.00452788 \\ -0.00452788 & 0.99078950 & 0.08610296 \\ -0.08610926 & -0.17673381 & 0.73248062 \end{bmatrix}$$

$$F^{-1} = \begin{bmatrix} 1.00017977 & -0.09964488 & 0.00553055 \\ -0.00553055 & 0.98911866 & -0.11623654 \\ 0.11623654 & 0.22694252 & 1.33782828 \end{bmatrix}$$

$$(AT)^0 = \frac{1}{2}(F - F^{-1})$$

$$(AT)^0 = \begin{bmatrix} -0.00016725 & 0.99665748 & -0.00050133 \\ 0.00050133 & 0.00083541 & 0.10116975 \\ -1.01169753 & -0.21018381 & -0.30267384 \end{bmatrix}$$

$$(AT)^1 = (AT)^0 + I - F^{-1}F^0$$

$$F^0 = e^{(AT)^0}$$

$$F^0 = \begin{bmatrix} 0.99971053 & 0.09940703 & 0.00410266 \\ -0.00410266 & 0.99150520 & 0.08709903 \\ -0.08709903 & -0.17830073 & 0.73020810 \end{bmatrix}$$

$$(AT)^1 = \begin{bmatrix} 0.00001538 & 0.10002553 & 0.00003577 \\ -0.00003577 & -0.00005617 & 0.09991802 \\ -0.09991802 & -0.19998718 & -0.29998102 \end{bmatrix}$$

$$F^{1} = \begin{bmatrix} 0.99845551 & 0.09968705 & 0.00452853 \\ -0.04528531 & 0.99078848 & 0.08610146 \\ -0.08610146 & -0.17673145 & 0.73248441 \end{bmatrix}$$

$$(AT)^{2} = \begin{bmatrix} 0.00001503 & 0.10002481 & 0.00003495 \\ -0.00003495 & -0.00005488 & 0.09991993 \\ -0.09991993 & -0.19987482 & -0.29981468 \end{bmatrix}$$

The maximum error which can be shown, is less than 10^{-3}.

All discrete-time identification methods which are appropriate for the system, can be used to obtain discrete-time model for transformation to the continuous-time model.

When F is taken below, AT is calculated and given by

$$F = \begin{bmatrix} 0.9950 & 0.09990 & 0.00450 \\ -0.0045 & 0.99500 & 0.08630 \\ -0.0860 & -0.17700 & 0.73000 \end{bmatrix}$$

$$AT = \begin{bmatrix} 0.0054 & 0.0998 & -0.0470 \\ -0.0001 & 0.0036 & 0.1001 \\ -0.1002 & -0.1532 & -0.3062 \end{bmatrix}$$

2.3.2. DIRECT METHOD

Some identification methods of continuous-time have been suggested for the calculation of the parameters, directly. In this study, the quasilinearisation approach method is considered because of more accuracy.

2.3.2.1. QUASILINEARIZATION

The quasilinearization approach was first introduced by BELLMAN and KALABA [16], [17] for solving boundary value problems arising in nonlinear differential equation. Its application to the identification of parameters of nonlinear systems is mainly due to Kumar and Sridhar, Sage and Eisenberg [18], [19], [20], Detchmond and Shridar [21].

Quasilinearization is in essence a method for transforming a nonlinear multi-point boundary value problem which is basically stationary into a linear nonstationary such problem. It is applicable to continuous and to discrete process. The quasilinearization approach is of an iterative nature and requires no special inputs, this being suitable for on-line application.

Consider a vector differential equation

$$\dot{x} = f(x,a,u,t) \qquad t_0 \le t \le t_T \qquad (2.89)$$

be given with boundary condition.

$$< a(t_i),\ x(t_i) > = b_i \qquad i = 1,\ 2,\ \dots,\ n \qquad (2.90)$$

$$t_0 \le t_1 \le t_2 \le \dots \le t_T$$

where a is an unknown parameter, x is the state variable and b is the known initial condition. It is assumed that (2.89) and (2.90) have a unique solution on (t_0, t_T).

Consider the parameter a to be a function of time that satisfies the differential equation

$$\dot{a} = 0 \qquad (2.91)$$

Let $x_0(t)$ be an initial guess to solution of (2.89) on [t_0, t_T]. Eq(2.89) is linearized around the k th approximation by expanding the function f in a Taylor series and retaining only the linear terms. The (k+1) st approximation is

$$\dot{x}_{k+1} = f\ (x_k) + J\ (x_k)\ (\ x_{k+1} - x_k\ ) \qquad (2.92)$$

where J (x) is the Jacobian matrix with elements

$$J_i = \frac{\partial f_i}{\partial x_j} \qquad (2.93)$$

The components of the initial approximation vector $x_0(t)$ constants, suitably chosen function of time, polynomials in t, etc. The first approximation $x_1(t)$ is obtained as a solution of

$$\dot{x}_1 = f\ (x_0, t) + J\ (\ f(x_0, t)\ )\ (\ x_1 - x_0\ ) \qquad (2.94)$$

$$= J\ (\ f(x_0,t)\ )\ x_1 + f(x_0,t) - J\ (\ f(x_0,t)\ )\ x_0 \tag{2.95}$$

satisfying (2.90).

Let h(t) be the homogeneous solution matrix

$$\dot{h} = J\ (\ f(x,t)\ )\ h \tag{2.96}$$

Let p(t) be the particular solution vector of

$$\dot{p} = J\ (\ f(x_0,t)\ )\ p + f\ (x_0,t) - J\ (\ f(x_0,t)\ )\ x_0 \tag{2.97}$$

Then the solution of (2.91) is written as

$$x_{k+1}(t) = h(t)\ a_{k+1} + p(t) \tag{2.98}$$

The unknown parameter a is calculated by minimizing the sum of Q:

$$Q = \sum_{i=1}^{N} [\ x(t_i,a) - b_i\ ]^2, \tag{2.99}$$

where all variable were described with Eq(2.90). Eq(2.98) is placed into Eq(2.99), and it becomes

$$Q = \sum_{i=1}^{N} [\ p(t_i) + a_{k+1}h(t_i) - b_i\ ]^2 \tag{2.100}$$

The condition for minimum Q is given by

$$\frac{\partial Q}{\partial a_{k+1}} = 2 \sum_{i=1}^{N} h(t_i) \left[p(t_i) + a_{k+1} h(t_i) - b_i \right] = 0 \qquad (2.101)$$

solving for a_{k+1} gives

$$a_{k+1} = \frac{\sum_{i=1}^{N} \left[b_i - p(t_i) \right] h(t_i)}{\sum_{i=1}^{N} h^2(t_i)} \qquad (2.102)$$

When the observable value is c, the initial conditions can be taken as below,

$x(0) = c$

$p(0) = c$

$H(0) = 0$

p and h are integrated and stored from $t=0$ to $t=t_N$. The unknown parameter a_{k+1} is solved by using Eq(2.102). This iteration is repeated until a_k approaches a_{k+1}.

The equations (2.90) to (2.102) are for the first order system. For higher order system, however, state vector replaces by state variable and unknown parameters vector is found as seen

$$[\, a \,]_{k+1} = \Big(\sum_{i=1}^{N} [\, h(t\ )\ ]^{T} [\, h(t\ )\ ] \Big)^{-1}$$

$$\sum_{i=1}^{N} [\, h(t_i)\ ]^{T} \{\ [b(t_i)]-[p(t_i)]\ \} \qquad (2.103)$$

J.K.M.MacCORMAC [22] has approached that calculation by using Newton-Raphson algorithm. In this method, a particular system which has same structure with actual system, is chosen. It is represented by

$$p\ (\ t\ ) = g\ (\ p,\ \hat{a},\ u,\ t\ ) \qquad (2.104)$$

The particular system initial condition has to be equal to the actual system initial condition and integrated from initial time t_0 to the next observed time or to the desired time t_T. The actual system result is achievied by adding the product of the homogeneous system result and quantity constant to the particular system result. For this operation, the homogeneous system is described by the Jacobian of the particular system. It can be written by

$$\dot{h}\ (\ t\ ) = J\ [\ g\ (\ p,\ \hat{a},\ u,\ t\ )\ ]\ h \qquad (2.105)$$

The homogeneous system initial condition is zero because the particular system initial condition is same as the actual system. The homogeneous system is integrated between boundary of particular system. The results of integrations is given

$$x_{k+1}(t_T) = p(t_T) + c_{k+1}h(t_T) \tag{2.106}$$

where c is called quantity constant. The next step particular parameter is obtained:

$$\hat{a}_{k+1} = \hat{a}_k + c_{k+1} \tag{2.107}$$

The unknown quantity constant can be solved from Eq(2.106) in reference [22]. It gives

$$c = \frac{x_{k+1}(t_T) - p(t_T)}{h(t_T)} \tag{2.108}$$

c is placed in to Eq(2.107):

$$\hat{a}_{k+1} = \hat{a}_k + \frac{x_{k+1}(t_T) - p(t_T)}{h(t_T)} \tag{2.109}$$

J.K.M.MacCORMAC has also shown that this method approaches the actual value in the second iteration. Eq(2.109) can be changed for the identification of higher order system:

$$[\hat{a}]_{k+1} = [\hat{a}]_k + [h(t_T)]^{-1} \{ [x(t_T)] - [p(t_T)] \} \tag{2.110}$$

The calculation of quantity constant has been developed for a system with noise present by approaching Eq(2.102) that has improved by KALABA (1983) [23]. In this situation, equation(2.108) becomes:

$$c_{k+1} = \frac{\sum_{i=1}^{N} [\, b_i - p(t_i)\,]\, h(t_i)}{\sum_{i=1}^{N} h_i^2(t\ )}$$

It is changed for higher order system,

$$[\ c\ ]_{k+1} = \Big(\sum_{i=1}^{N} [h(t_i)]^T\ [\ h(t_i)\ ] \Big)^{-1}$$

$$\sum_{i=1}^{N} [\ h(t_i)]^T \{\ [b(t_i)] - [p(t_i)]\ \} \qquad (2.111)$$

This method is explained further with following application.

2.3.2.1.1. THE APPLICATION OF TIME-INVARIANT SYSTEM

Considered system can be taken by

$$\dot{x} = A\, x + B\, u \qquad (2.112)$$

$$\dot{x} = f\ (\ x, a\ )$$

The system may be canonical form or another form, but the state variable must be observed. The controllable canonical form was chosen for the explanation. Another form will be discussed in the next chapters. When the system equation is written in the notation form.

$$\begin{bmatrix} x_1 \\ \dot{x}_2 \\ \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \\ -a_1 & -a_2 & -a_3 & \dots & -a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \\ a_1 \end{bmatrix} u \qquad (2.113)$$

$$\begin{aligned} f_1 &= \dot{x}_1 = x_2 \\ f_2 &= \dot{x}_2 = x_3 \\ &\cdot \\ &\cdot \\ f_n &= \dot{x}_n = -a_1 x_1 - a_2 x_2 - \dots - a_n x_n + a_1 u \\ \\ \dot{a}_1 &= 0 \\ \dot{a}_2 &= 0 \\ &\cdot \\ &\cdot \\ \dot{a}_n &= 0 \end{aligned} \qquad (2.114)$$

The Jacobian matrix from the definition (2.93)

$$
J = \left[\begin{array}{ccccc:cccc}
0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
. & & & & & & & & . \\
. & & & & & & & & . \\
-a_1 & -a_2 & -a_3 & \dots & -a_n & (u-x_1) & -x_2 & \dots & -x_n \\
\hline
 & & & & & & & & \\
 & & 0 & & & & 0 & & \\
 & & & & & & & &
\end{array}\right] \qquad (2.115)
$$

Equation (2.92) can be rewritten

$$
\left[\begin{array}{c}
x_1 \\ \dot{x}_2 \\ . \\ . \\ \dot{x}_n \\ \hline \dot{a}_1 \\ \dot{a}_2 \\ . \\ . \\ \dot{a}_n
\end{array}\right]_{N+1}
=
\left[\begin{array}{ccccc:cccc}
0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
. & & & & & & & & . \\
. & & & & & & & & . \\
-a_1 & -a_2 & -a_3 & \dots & -a_n & (u-x_1) & -x_2 & \dots & -x_n \\
\hline
 & & & & & & & & \\
 & & 0 & & & & 0 & & \\
 & & & & & & & &
\end{array}\right]
\left(
\left[\begin{array}{c}
x_1 \\ x_2 \\ \\ x_n \\ a_1 \\ a_2 \\ \\ a_n
\end{array}\right]_{N+1}
-
\left[\begin{array}{c}
x_1 \\ x_2 \\ \\ x_n \\ a_1 \\ a_2 \\ \\ a_n
\end{array}\right]_{N}
\right)
$$

$$
+ f (x_N)
$$

The particular system can be chosen as below,

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{p}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ -\hat{a}_1 & -\hat{a}_2 & -\hat{a}_3 & \dots & -\hat{a}_n \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \cdot \\ \cdot \\ \cdot \\ p_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ a_1 \end{bmatrix} u\,(t) \qquad (2.116)$$

The initial conditions are

[p(t)] = [x(t)] = [b(t)].

Homogeneous systems are

$$\begin{bmatrix} \dot{h}_1 \\ \dot{h}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{h}_n \end{bmatrix}_1 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ -\hat{a}_1 & -\hat{a}_2 & -\hat{a}_3 & \dots & -\hat{a}_n \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ \cdot \\ h_n \end{bmatrix}_1 + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ u-x_1 \end{bmatrix}$$

$$\begin{bmatrix} \dot{h}_1 \\ \dot{h}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{h}_n \end{bmatrix}_2 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ -\hat{a}_1 & -\hat{a}_2 & -\hat{a}_3 & \dots & -\hat{a}_n \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ \cdot \\ h_n \end{bmatrix}_2 + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ -x_2 \end{bmatrix}$$

$\cdot$

$\cdot$

$$\begin{bmatrix} \dot{h}_1 \\ \dot{h}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{h}_n \end{bmatrix}_N = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ -\hat{a}_1 & -\hat{a}_2 & -\hat{a}_3 & \dots & -\hat{a}_n \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ \cdot \\ h_n \end{bmatrix}_N + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ -x_n \end{bmatrix} \qquad (2.117)$$

The considered system for application was taken as seen below,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} u \qquad (2.118)$$

The state variables are observed, and represented by

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \eta \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} \qquad (2.119)$$

where η is the amplitude of noise and $\xi_{1,2}$ are pseudo-random noises. The particular system can be described and homogeneous systems can be obtained from it as seen below:

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\hat{a}_1 & -\hat{a}_2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{a}_3 \end{bmatrix} \tag{2.120}$$

$$\begin{bmatrix} \dot{h}_{11} \\ \dot{h}_{12} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\hat{a}_1 & -\hat{a}_2 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ -x_1 \end{bmatrix} \tag{2.121}$$

$$\begin{bmatrix} \dot{h}_{21} \\ \dot{h}_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\hat{a}_1 & -\hat{a}_2 \end{bmatrix} \begin{bmatrix} h_{21} \\ h_{22} \end{bmatrix} + \begin{bmatrix} 0 \\ -x_2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{h}_{31} \\ \dot{h}_{32} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\hat{a}_1 & -\hat{a}_2 \end{bmatrix} \begin{bmatrix} h_{31} \\ h_{32} \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix}$$

$$H = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \end{bmatrix} \tag{2.122}$$

Initial conditions are

$$p_1(t_0) = x_1(t_0) = b_1(t_0)$$

$$p_2(t_0) = x_2(t_0) = b_2(t_0)$$

$$h_{ij}(t_0) = 0 \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n+1$$

The particular and homogeneous systems are integrated and replaced in Eq(2.106).

$$\begin{bmatrix} b_1 \\ \\ b_2 \end{bmatrix}_{t=t_i} = \begin{bmatrix} p_1 \\ \\ p_2 \end{bmatrix}_{t=t_i} + \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ \\ h_{12} & h_{22} & h_{32} \end{bmatrix}_{t=t_i} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{2.123}$$

From the rank of the homogeneous matrix, it can easily be seen that it is not satisfactory to solve quantity vector. Quantity values are constant, therefore next-step solution of particular and homogeneous systems and next-step observed value are added to Eq(2.123).

$$\begin{bmatrix} b_1(t_i) \\ b_2(t_i) \\ b_2(t_{i+1}) \end{bmatrix} = \begin{bmatrix} p_1(t_i) \\ p_2(t_i) \\ p_2(t_{i+1}) \end{bmatrix} + \begin{bmatrix} h_{11}(t_i) & h_{21}(t_i) & h_{31}(t_i) \\ h_{12}(t_i) & h_{22}(t_i) & h_{32}(t_i) \\ h_{12}(t_{i+1}) & h_{22}(t_{i+1}) & h_{32}(t_{i+1}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{2.124}$$

The second variables of the next step can be added to Eq(2.123). Because first variables depend on the previous step second variables in the system can be represented with canonical form.

The application results that are illustrated in Fig.2.5. are for the different noise amplitude. Initial particular values were chosen:

$$\hat{a}_1 = \hat{a}_3 = 1,\ \hat{a}_2 = 8$$

2.3.2.1.2. TIME-VARYING SYSTEM

For the identification of time varying systems it is assumed that the parameters are constant during the identification period. Therefore the identification time depends on the rate of change of the system parameters. When many samples can be taken in this time, Eq(2.111) may be used for identification. Otherwise Eq(2.101) must be used. The system used to demonstrate the method was

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_1(t) & -a_2(t) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ a_1(t) \end{bmatrix} u \qquad (2.125)$$

Sample time was 50 milliseconds and control input was unit-step function. The particular system was

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -7 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad (2.126)$$

Results are illustrated in Fig.2.6. with time axis.

2.4. CONCLUSION

Some discrete-time identification methods have been discussed in this chapter. It can easily be seen that all methods are for input-output model and single variable. In practice, systems will have more than one variable. The application of these methods becomes much more difficult, when the system is a multivariable.

The continuous-model algorithm results seem to be more accurate than those of the discrete-time model ,when they are compared as shown by ISERMANN (1974) [24] and SARIDIS (1974) [25]. It seems to take more time because of the integration of the models, although it does produce the actual values of the system parameters in the state-space form.

The transformation algorithm from the discrete-model to continuous-model is only valid for the state-space models. Its results are very good if the identification error is zero. When the discrete-time identification result is carrying 1 or 2 percent error, it reflects more than 25 percent error on the continuous-time models.

The impression that the continuous-time model seems to take more time is not true, because the discrete-time algorithms include many matrix inversion and matrix multiplication. This is true especially for the multivariable system where the algorithm is applied for each iteration and the result converted to the state-space form and at least it is transformed to continuous-time model. The matrix inversion takes the longest time in this operations. The continuous-time algorithm requires only one matrix inversion per iteration. The integration does not require more computation time. It can be suggested that the continuous-time algorithms preferable for on-line identification .

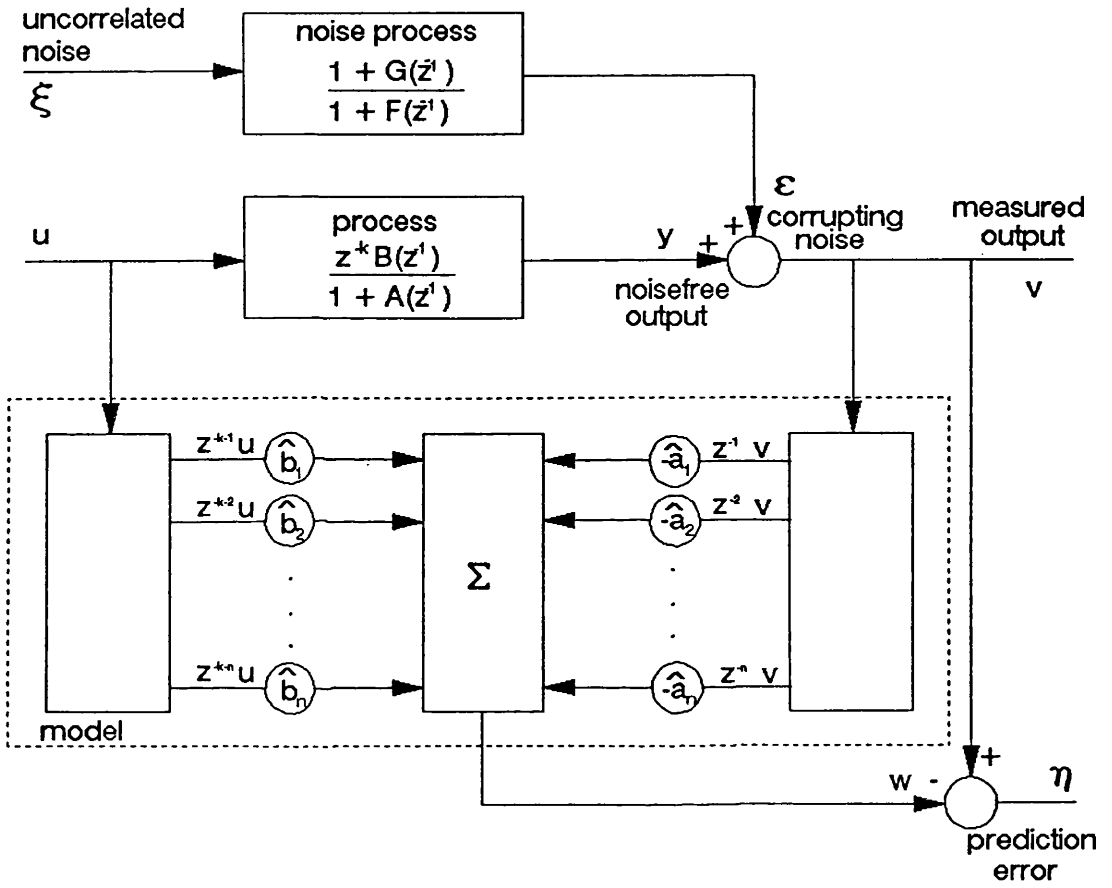


Fig.2.1. The block diagram of Generalized Least Square Method

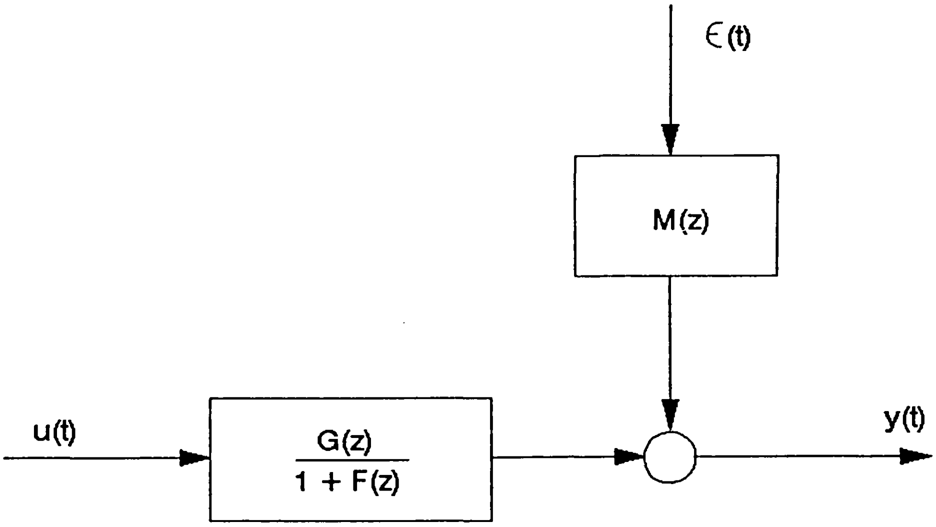


Fig.2.2. The Process Model

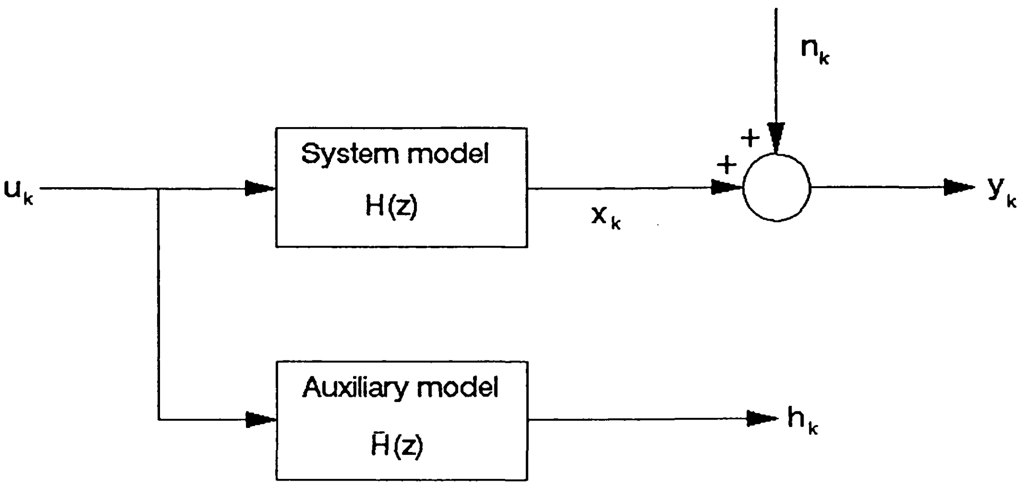


Fig.2.3. Block diagram of Instrumental Variable Method

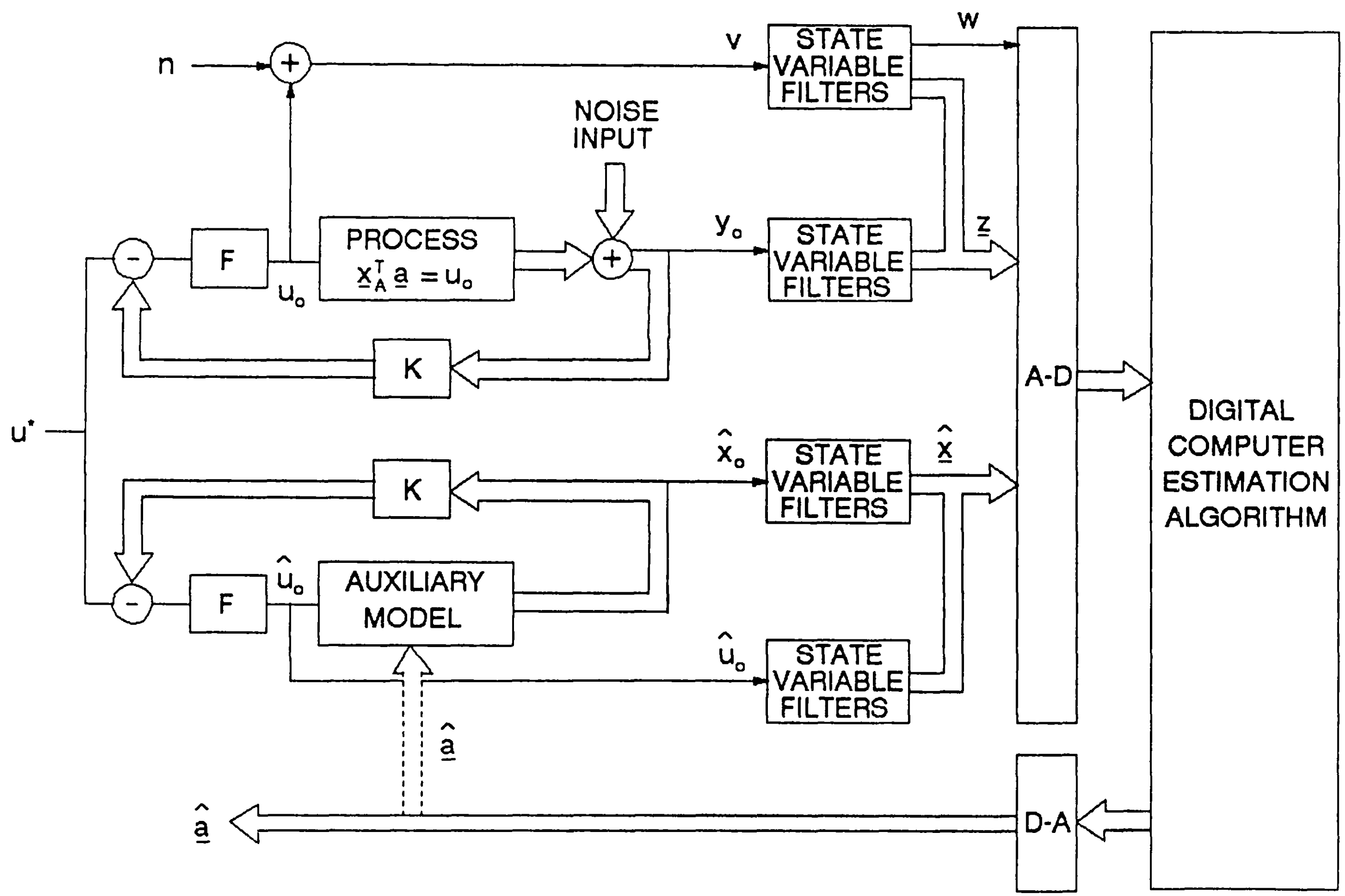


Fig.2.4. Block Diagram of the Istrumental Variable Method

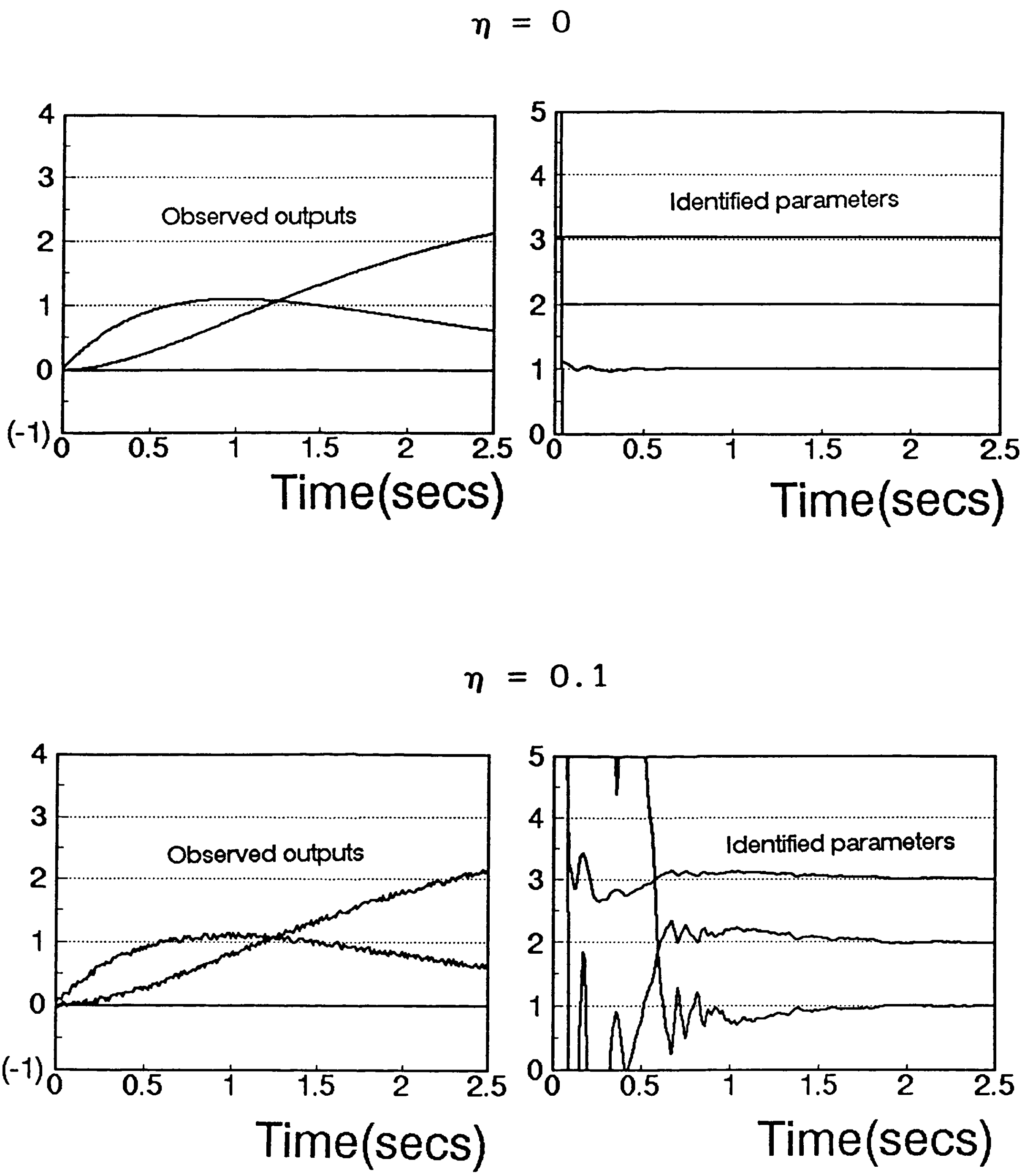
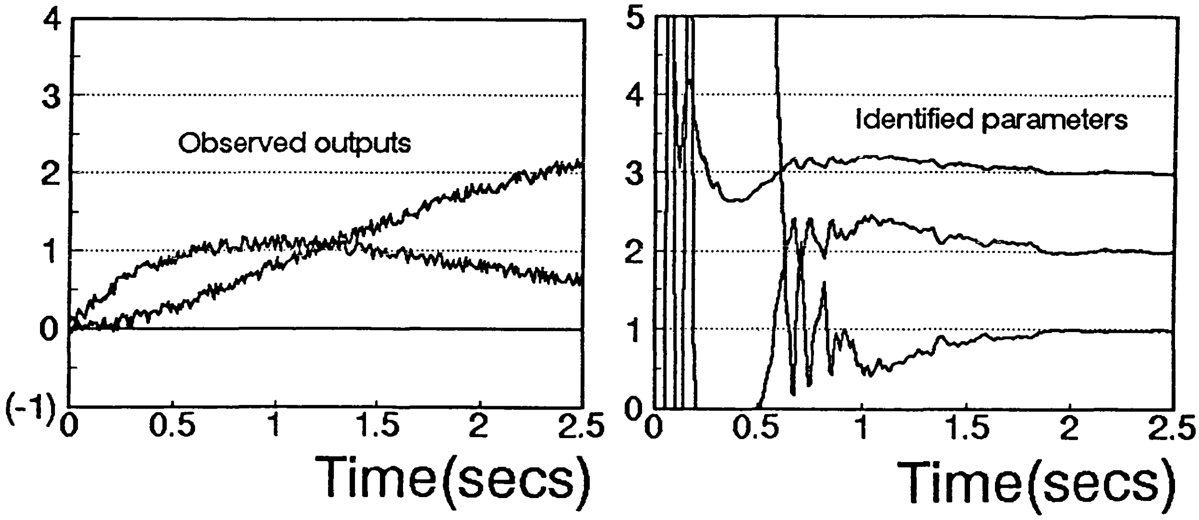


Fig.2.5. The results of the identification for various noise amplitude.

$\eta = 0.2$



$\eta = 0.4$

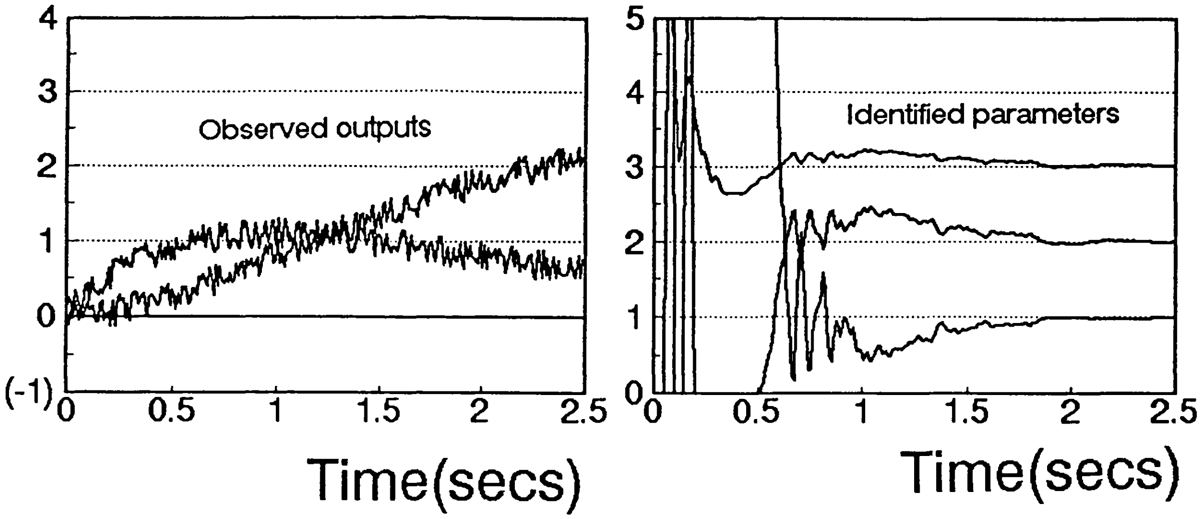


Fig.2.5. (continued)

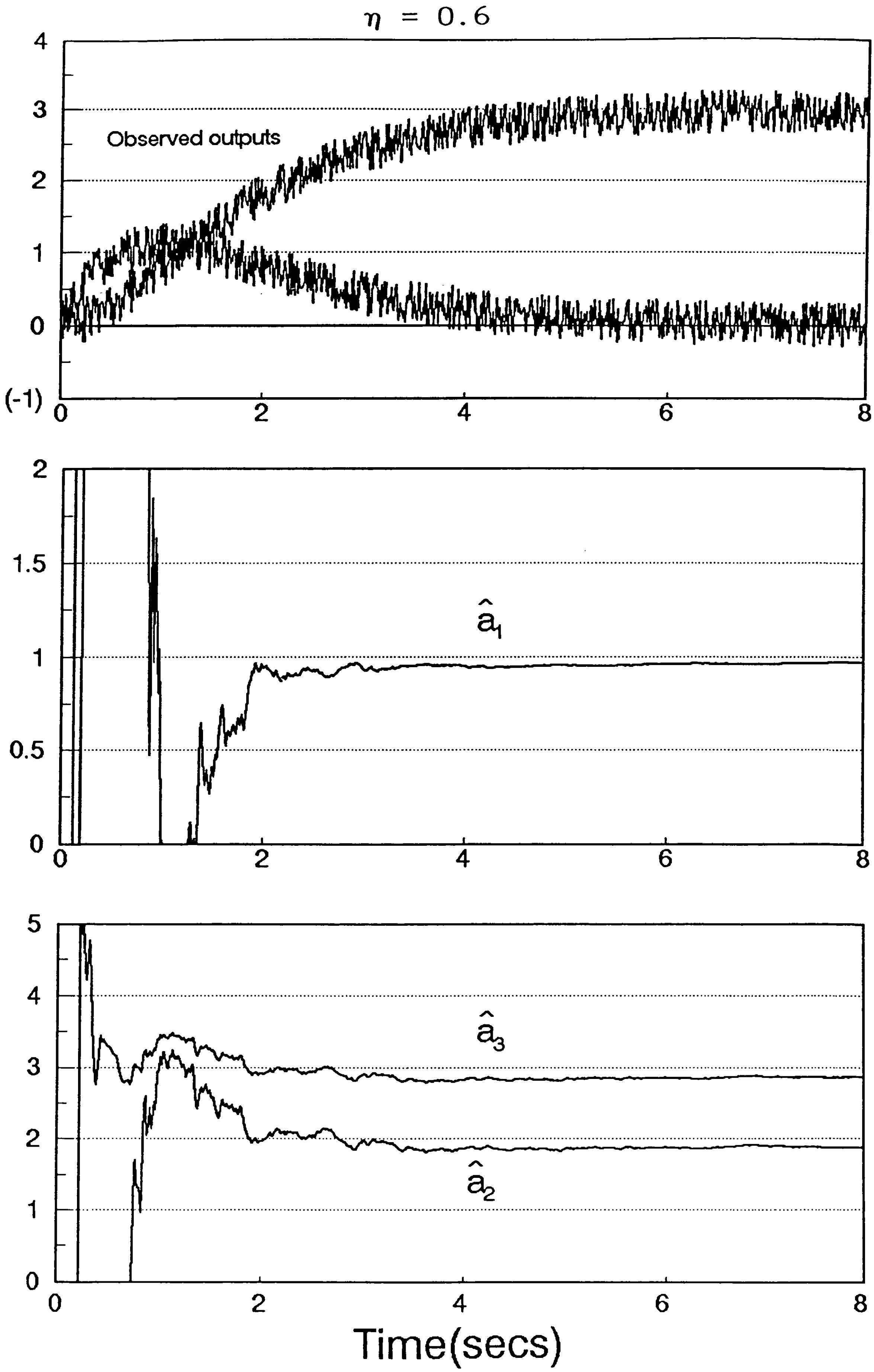


Fig.2.5. (continued)

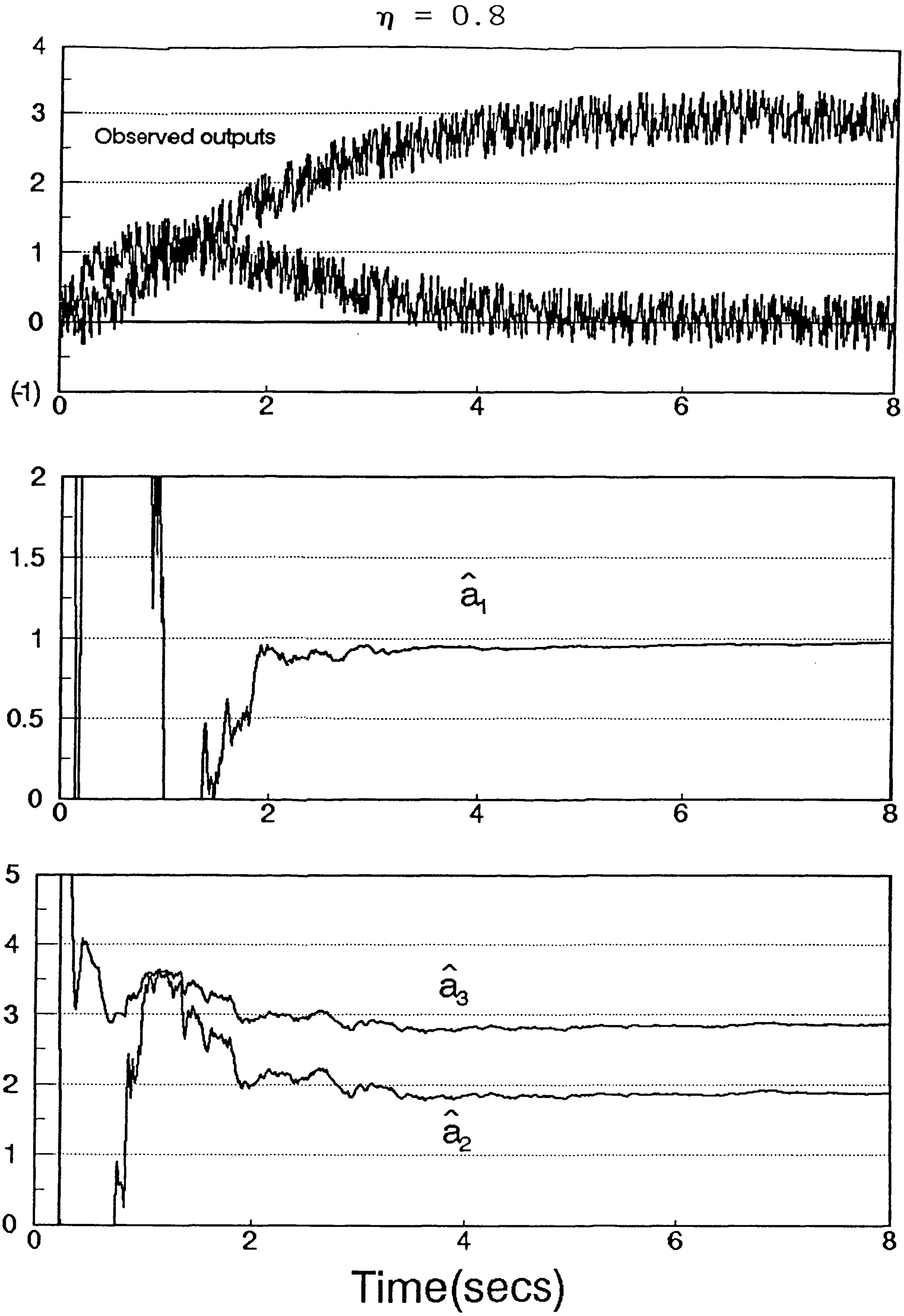


Fig.2.5. (continued)

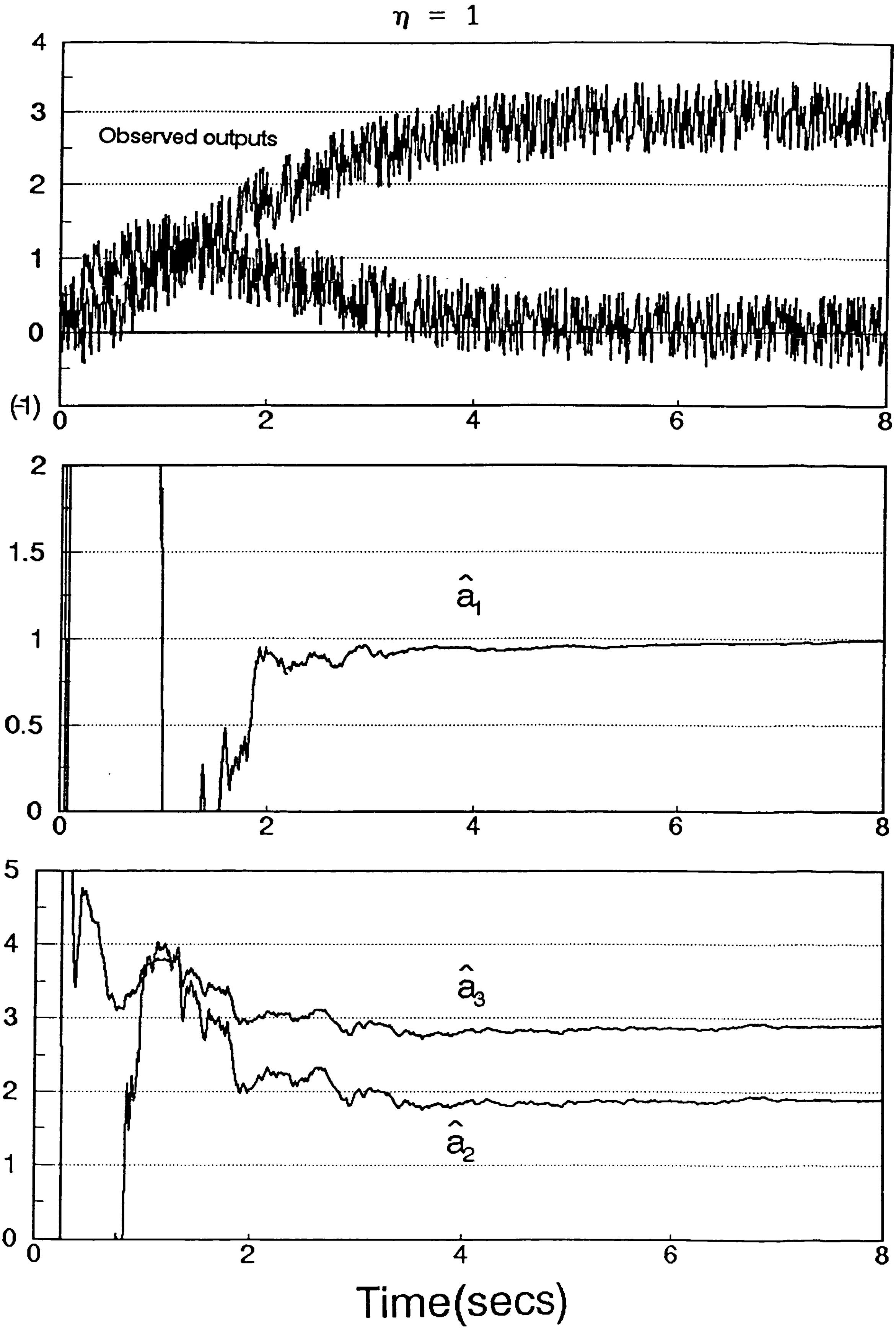


Fig.2.5. (concluded)

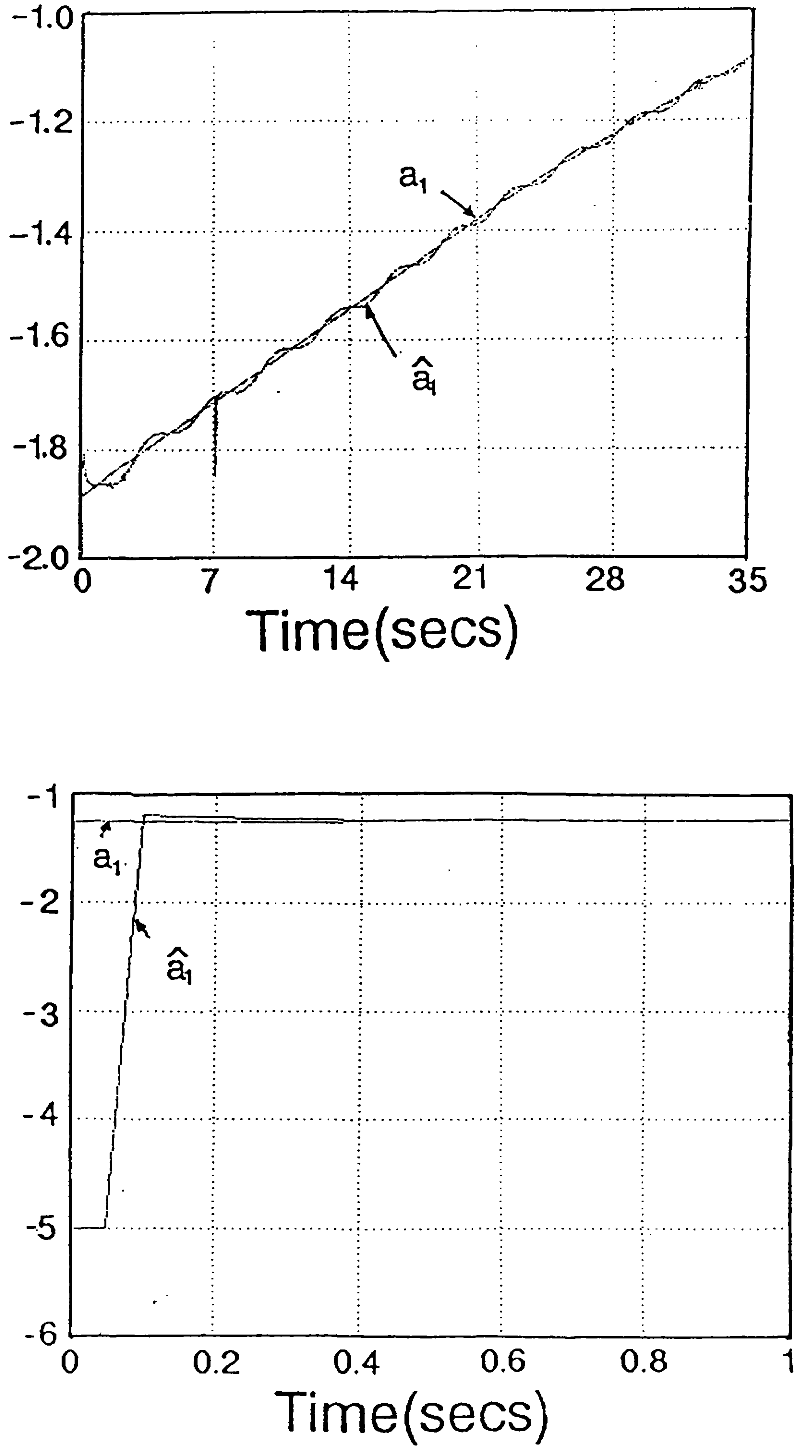


Fig.2.6. The results of the identification of time-varying parameter

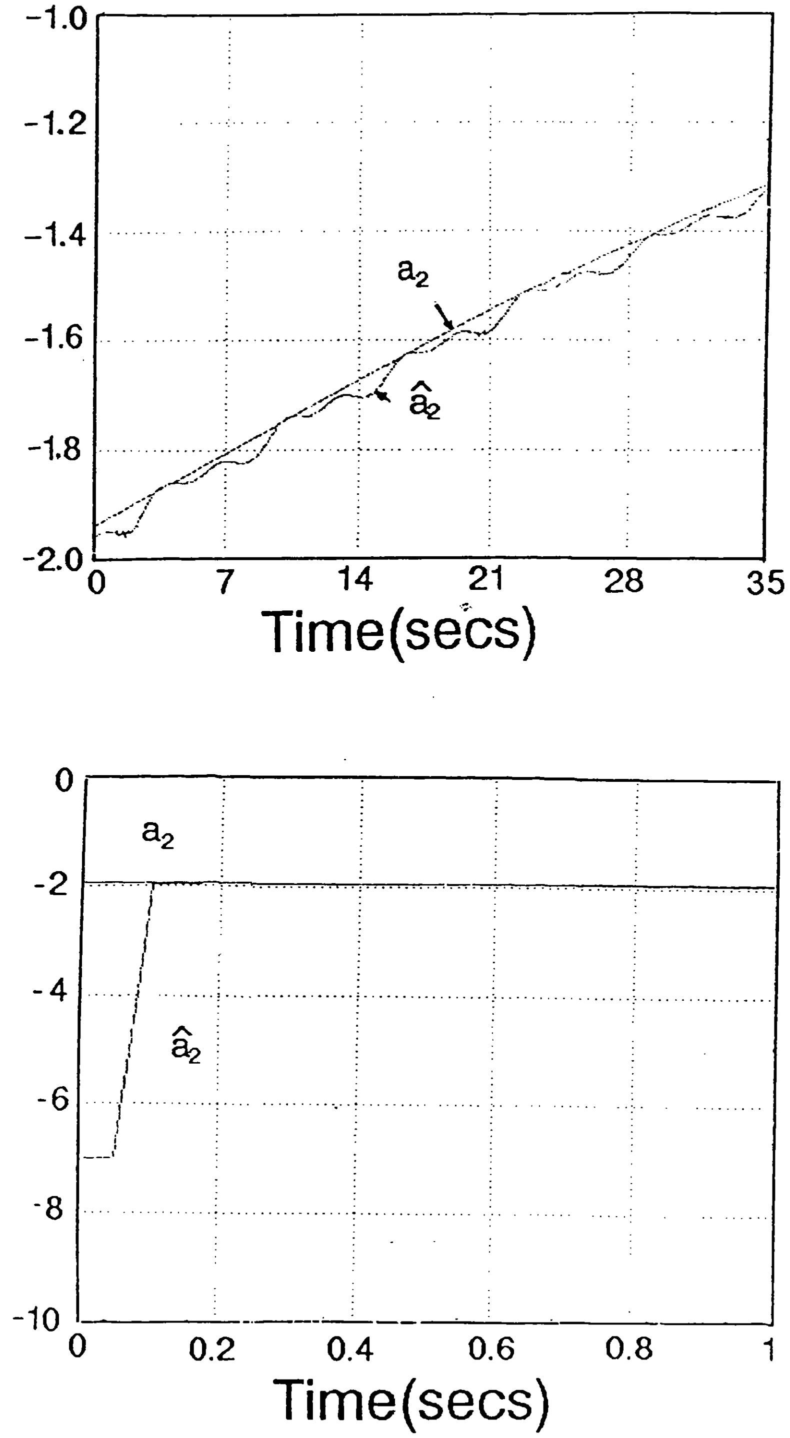


Fig.2.6. The results of the identification of time-varying parameter

CHAPTER 3. AIRCRAFT DYNAMICS

3.1. INTRODUCTION

In this chapter, we will discuss equations of motion of an aircraft, stability of longitudinal dynamics and stability of lateral dynamics. First of all, we will define axes systems. In general, the axes, which are called as "body axes", are fixed in the aircraft and move with the aircraft. This is chosen as a rectangular axes $Oxyz$ where O is the centre of gravity of the aircraft. It is shown Fig.3.1.

The equations of motion of the aircraft will be determined according to the axes system. The stability of dynamics will be investigated.

To discuss all detail about the aircraft dynamics is not possible in one chapter, therefore we will try to give a summary of them and the main idea.

3.2. THE EQUATIONS OF MOTION

The Fig.3.1. is considered to explain forces, moments and motions. In Fig.3.1., U, V, W are velocity components of the centre of gravity Ox, Oy, Oz respectively. p, q, r are the components are angular velocity of the axes frame Oxyz about Ox, Oy, Oz respectively. External forces are defined along Ox, Oy, Oz with respect X, Y, Z . The moments of external forces about Ox, Oy, Oz are L, M, N respectively.

m, I_x , I_y , I_z , I_{xy} , I_{xz} , I_{yz} represent the mass of the aircraft, the moments of inertia of the aircraft about Ox, Oy, Oz, the products of inertia with respect to Oxy, Oxz, Oyz. According to this definition, the motion of the aircraft is defined as given by

Motion parallel to Ox : $m\,(\dot{U} - rV + qW) = X$ (3.1)

parallel to Oy : $m\,(\dot{V} - pW + rU) = Y$ (3.2)

parallel to Oz : $m\,(\dot{W} - qU + pV) = Z$ (3.3)

Angular motion about Ox :

$$L = I_x\dot{p} - (I_y - I_z)qr - I_{yz}(q^2 - r^2) - I_{zx}(\dot{r} + pq) - I_{yx}(\dot{q} - rp) \tag{3.4}$$

$$M = I_y\dot{q} - (I_z - I_x)rp - I_{zx}(r^2 - p^2) - I_{xy}(\dot{p} + qr) - I_{yz}(\dot{r} - pq) \tag{3.5}$$

$$N = I_z\dot{r} - (I_x - I_y)pq - I_{xy}(p^2 - q^2) - I_{yz}(\dot{q} + rp) - I_{zx}(\dot{p} - qr) \tag{3.6}$$

A.W. Babister [26] has shown that the second order terms can be neglected for small disturbance of a symmetric aircraft. He has also rewritten the Eqs (3.1) to (3.6) as seen below,

$$m\,(\dot{u}+qW_e) = X = X_a + X_g \tag{3.7}$$

$$m\,(\dot{v} - pW_e + rU_e) = Y = Y_a + Y_g \tag{3.8}$$

$$m(\dot{w}-qU_e) = Z = Z_a + Z_g \tag{3.9}$$

$$I_{xy} = \sum xy\ \delta m = 0 \tag{3.10}$$

$$I_{yz} = \sum yz\ \delta m = 0 \tag{3.11}$$

$$L_a = I_x\dot{p} - I_{zx}\dot{r} \tag{3.12}$$

$$M_a = I_y\dot{q} \tag{3.13}$$

$$N_a = -I_{zx}\dot{p} + I_z r \tag{3.14}$$

The angle of pitch θ, the angle of yaw ψ and the angle of bank ϕ need to be defined for the explanation of the relationship with p, q and r.

θ, ψ and ϕ can be derived with using Fig.3.2. It assumed that the steady axes are Ox_0, Oy_0, Oz_0. Firstly, we rotate the axes about Oz_0 in a clockwise direction (Fig.3.2.a). ψ is defined the angle of yaw that is between $Ox_0y_0z_0$ axes and $Ox_1y_1z_0$ axes. In the second step, the axes $Ox_1y_1z_0$ is rotated about Oy_1 in a clockwise direction (Fig.3.2.b). θ is called the angle of pitch that is between $Ox_1y_1z_0$ axes and Oxy_1z_2. Finally, Oxy_1z_2 axes is rotated about Ox clockwise. ϕ is named the angle of bank (or roll) that is between Oxy_1z_2 axes and $Oxyz$ axes.

The components of the angular velocity of the axes $Oxyz$ have been defined p, q and r about Ox, Oy, Oz respectively. We can write the relations between the angles and the components of the angular velocity as

$$p = \dot{\phi} - \dot{\psi}\ sin\theta \qquad (3.15)$$

$$q = \dot{\theta}\ cos\phi + \dot{\psi}\ cos\theta\ sin\phi \qquad (3.16)$$

$$r = -\ \dot{\theta}\ sin\phi + \dot{\psi} cos\theta\ cos\phi \qquad (3.17)$$

In reference [26], second order quantities has been shown to neglect for small disturbance. According this neglecting, the Eqs (3.15) to (3.17) are rewritten ;

$$p = \dot{\phi} \quad \text{(rate of roll)} \qquad (3.18)$$

$$q = \dot{\theta} \quad \text{(rate of pitch)} \qquad (3.19)$$

$$r = \dot{\psi} \quad \text{(rate of yaw)} \qquad (3.20)$$

The gravitational forces are determined by using the angle that is between Ox axis and horizontal as seen in Fig.3.3.

$$X_g = mg\ sin\Theta_e = mg_2 \qquad (3.21)$$

$$Z_g = mg\ cos\Theta_e = mg_1 \tag{3.22}$$

The aerodynamic forces and moments cause to change the incidence and velocity components of the aircraft. Therefore, the equations of the response of the aerodynamic forces and moments can be written for small disturbances by using the component of the velocity of the aircraft without second order term as seen below,

$$X_a = \mathring{X}^{\circ}_{ae} + \mathring{X}_u u + \mathring{X}_v v + \mathring{X}_w w + \mathring{X}_{\dot{w}}\dot{w} + \mathring{X}_p p + \mathring{X}_q q + \mathring{X}_r r + \mathring{X}(t) \tag{3.23}$$

$$Y_a = \mathring{Y}_{ae} + \mathring{Y}_u u + \mathring{Y}_v v + \mathring{Y}_w w + \mathring{Y}_{\dot{w}}\dot{w} + \mathring{Y}_p p + \mathring{Y}_q q + \mathring{Y}_r r + \mathring{Y}(t) \tag{3.24}$$

$$Z_a = \mathring{Z}_{ae} + \mathring{Z}_u u + \mathring{Z}_v v + \mathring{Z}_w w + \mathring{Z}_{\dot{w}}\dot{w} + \mathring{Z}_p p + \mathring{Z}_q q + \mathring{Z}_r r + \mathring{Z}(t) \tag{3.25}$$

$$L_a = \mathring{L}_u u + \mathring{L}_v v + \mathring{L}_w w + \mathring{L}_{\dot{w}}\dot{w} + \mathring{L}_p p + \mathring{L}_q q + \mathring{L}_r r + \mathring{L}(t) \tag{3.26}$$

$$M_a = \mathring{M}_u u + \mathring{M}_v v + \mathring{M}_w w + \mathring{M}_{\dot{w}}\dot{w} + \mathring{M}_p p + \mathring{M}_q q + \mathring{M}_r r + \mathring{M}(t) \tag{3.27}$$

$$N_a = \mathring{N}_u u + \mathring{N}_v v + \mathring{N}_w w + \mathring{N}_{\dot{w}}\dot{w} + \mathring{N}_p p + \mathring{N}_q q + \mathring{N}_r r + \mathring{N}(t) \tag{3.28}$$

The coefficient of the components in Eqs(3.23) to (3.28) are called aerodynamic derivatives which are given in the list of symbols. $\mathring{X}_{ae}$, $\mathring{Y}_{ae}$, $\mathring{Z}_{ae}$ are the steady state values of X_a, Y_a, Z_a . In the equations (3.23) to (3.25),

$$\mathring{X}_{ae} = mg\ sin\Theta_e \tag{3.29}$$

$$\mathring{Z}_{ae} = -mg\ cos\Theta_e \tag{3.30}$$

$$\mathring{Y}_{ae} = 0 \tag{3.31}$$

In reference [26], it has been shown that a symmetric disturbance can not cause an asymmetric reaction in the steady rectilinear flight with no roll or yaw. In this condition $\mathring{Y}_u$, $\mathring{Y}_w$, $\mathring{Y}_{\dot{w}}$, $\mathring{Y}_q$, $\mathring{L}_u$, $\mathring{L}_w$, $\mathring{L}_{\dot{w}}$, $\mathring{L}_q$, $\mathring{N}_u$, $\mathring{N}_w$, $\mathring{N}_{\dot{w}}$ and $\mathring{N}_q$ must all be zero. It has also shown that all symmetric forces and moments arising from asymmetric disturbances such as sideslip, rate of roll and rate of yaw are zero, when the second order terms are neglected. Therefore, $\mathring{X}_v$, $\mathring{X}_p$, $\mathring{X}_r$,$\mathring{M}_v$, $\mathring{M}_p$, $\mathring{M}_r$,$\mathring{Z}_v$, $\mathring{Z}_p$ and $\mathring{Z}_r$ are all zero,

We can now rewrite Eqs (3.23) to (3.28),

$$X_a = \mathring{X}^{\circ}_{ae} + \mathring{X}_u u + \mathring{X}_w w + \mathring{X}_{\dot{w}}\dot{w} + \mathring{X}_q q + \mathring{X}(t) \tag{3.32}$$

$$Y_a = \mathring{Y}_{ae} + \mathring{Y}_u u + \mathring{Y}_w w + \mathring{Y}_{\dot{w}}\dot{w} + \mathring{Y}_q q + \mathring{Y}(t) \tag{3.33}$$

$$Z_a = \mathring{Z}_{ae} + \mathring{Z}_u u + \mathring{Z}_w w + \mathring{Z}_{\dot{w}}\dot{w} + \mathring{Z}_q q + \mathring{Z}(t) \tag{3.34}$$

$$L_a = \mathring{L}_v v + \mathring{L}_p p + \mathring{L}_r r + \mathring{L}(t) \tag{3.35}$$

$$M_a = \mathring{M}_v v + \mathring{M}_p p + \mathring{M}_r r + \mathring{M}(t) \tag{3.36}$$

$$N_a = \mathring{N}_v v + \mathring{N}_p p + \mathring{N}_r r + \mathring{N}(t) \tag{3.37}$$

The equations (3.32) to (3.37) can be divided two parts;

a) the symmetric forces and moments that are the Eqs (3.32), (3.33) and (3.36),

b) the asymmetric forces and moments that are the Eqs (3.34), (3.35) and (3.37).

The symmetric forces and moments are derived with respect to the longitudinal plane of symmetry. Therefore they are called The Equations of The Longitudinal Motion. The other equations of them are called The Equations of The Lateral Motion.

a) The equation of the longitudinal motion for small disturbance;

$$m\dot{u} - \mathring{X}_u u - \mathring{X}_w w - \mathring{X}_{\dot{w}} \dot{w} + (mW_e - \mathring{X}_q)q + mg_1 \theta = \mathring{X}(t) \tag{3.38}$$

$$-\mathring{Z}_u u + (m - \mathring{Z}_{\dot{w}})\dot{w} - \mathring{Z}_w w - (mU_e + \mathring{Z}_q)q + mg_2 \theta = \mathring{Z}(t) \tag{3.39}$$

$$-\mathring{M}_u u - \mathring{M}_{\dot{w}} \dot{w} - \mathring{M}_w w + I_y \dot{q} - \mathring{M}_q q = \mathring{M}(t) \tag{3.40}$$

where $q = \theta$. U_e and W_e are steady state values.

b) The equation of the lateral motion for small disturbance;

$$m\dot{v} - \mathring{Y}_v v - (mW_e + \mathring{Y}_p)p + (mU_e - \mathring{Y}_r)r - mg_1 \phi - mg_2 \psi = \mathring{Y}(t) \tag{3.41}$$

$$\mathring{L}_v v + I_x \dot{p} - \mathring{L}_p p - I_{zx} \dot{r} + \mathring{L}_r r = \mathring{L}(t) \tag{3.42}$$

$$-\mathring{N}_v v - I_{zx} \dot{p} - \mathring{N}_p p + I_z \dot{r} - \mathring{N}_r r = \mathring{N}(t) \tag{3.43}$$

where $p = \dot{\phi}$ and $r = \dot{\psi}$.

The equations of motion, which are both the longitudinal and lateral, depend on aerodynamic derivatives. We can change aerodynamic derivatives with aeronormalized non-dimensional derivatives that are given by a list of symbols, which are written according to the notation of the Engineering Data Sheets.

The equation of the longitudinal symmetric motion for small disturbance can be recorded in non-dimensional form by multiplying Eqs(3.38) and (3.39) by $1/\ \frac{1}{2}\rho_e V_e^2 S$ and multiplying Eq(3.47) by $\mu_1/\frac{1}{2}\rho_e V_e^2 S\bar{\bar{c}} i_y$. In addition, some definition, which are included by the list of symbols, are used to simplify as follow,

$$x_u = -X_u\ ,\ x_w = -X_w\ ,\ x_{\dot{w}} = -X_{\dot{w}}/\mu_1\ ,\ x_q = -X_q/\mu_1$$

$$z_u = -Z_u\ ,\ z_w = -Z_w\ ,\ z_{\dot{w}} = -Z_{\dot{w}}/\mu_1\ ,\ z_q = -Z_q/\mu_1$$

$$m_u = -\mu_1 M_u/i_y\ ,\ m_w = -\mu_1 M_w/i_y\ ,\ m_{\dot{w}} = -M_{\dot{w}}/i_y\ ,\ m_q = -M_q/i_y$$

$$x_\eta = -X_\eta\ ,\ z_\eta = -Z_\eta\ ,\ m_\eta = -\mu_1/\ M_n i_y \tag{3.44}$$

Now, we can write Eqs (3.38) to (3.40) with new notation;

$$(\hat{D} + x_u)\hat{u} + (x_{\dot{w}}\hat{D} + x_w)\hat{w} + x_q\hat{q} + \hat{g}_1\theta + x_\eta\eta' = 0 \qquad (3.45)$$

$$z_u\hat{u} + [(1 + z_{\dot{w}})\hat{D} + z_w]\hat{w} + (z_q - 1)\hat{q} + \hat{g}_2\theta + z_\eta\eta' = 0 \qquad (3.46)$$

$$m_u\hat{u} + (m_{\dot{w}}\hat{D} + m_w)\hat{w} + (\hat{D} + m_q)\hat{q} + m_\eta\eta' = 0 \qquad (3.47)$$

where

$$\hat{D} = \frac{\partial}{\partial t} \;,\; \hat{q} = \hat{D}\theta \;,\; \hat{g} = mg \,/\, \tfrac{1}{2}\rho_e V_e^2 S = C_L \sec\Theta_e$$

$$\hat{g}_1 = \hat{g}\cos\Theta_e = C_L$$

$$\hat{g}_2 = \hat{g}\sin\Theta_e = C_L\tan\Theta_e$$

Similarly, the equation of the lateral assymmetric motion for small disturbances can be expressed in non-dimensional form by multiplying Eq (3.41) by $1/\frac{1}{2}\rho_e V_e^2 S$, Eq (3.42) by $\mu_2/\frac{1}{2}\rho_e V_e^2 S b i_x$ and Eq (3.43) by $\mu_2/\frac{1}{2}\rho_e V_e^2 S b i_z$.

We can also use the simplified equations, which are expressed in the list of symbols, as seen below ,

$$y_v = -Y_v \;,\; y_p = -Y_p/\mu_2 \;,\; y_r = -Y_r/\mu_2$$

$$l_v = -\mu_2 L_v/i_x \;,\; l_p = -L_p/i_x \;,\; l_r = -L_r/i_x$$

$$n_v = -\mu_2 N_v / i_x \ , \ n_p = -N_p / i_x \ , \ n_r = -N_r / i_x$$

$$y_\xi = Y_\xi \ , \ l_\xi = -\mu_2 L_\xi / i_x \ , \ n_\xi = -\mu_2 N_\xi / i_z$$

$$y_\zeta = Y_\zeta \ , \ l_\zeta = -\mu_2 L_\zeta / i_x \ , \ n_\zeta = -\mu_2 N_\zeta / i_z \quad (3.48)$$

The equations of the lateral motion are rewritten using the simplified equations.

$$(\hat{D} + y_v)\hat{v} + y_p\hat{p} + (1 + y_r)\hat{r} - \hat{g}_1\phi - \hat{g}_2\psi + y_\xi\xi + y_\zeta\zeta = 0 \quad (3.49)$$

$$l_v\hat{v} + (\hat{D} + l_p)\hat{p} + (e_x\hat{D} + l_r)\hat{r} + l_\xi\xi + l_\zeta\zeta = 0 \quad (3.50)$$

$$n_v\hat{v} + (e_z\hat{D} + n_p)\hat{p} + (\hat{D} + n_r)\hat{r} + n_\xi\xi + n_\zeta\zeta = 0 \quad (3.51)$$

where $\hat{D}$ is differential operator;

$$\hat{D} = \frac{\partial}{\partial t} \ , \ \hat{p} = \hat{D}\phi \ , \ \hat{r} = \hat{D}\psi \ ,$$

$$e_x = -i_{zx} / i_x = I_{zx}/I_x \ , \quad e_z = -i_{zx} / i_z = I_{zx}/I_z \quad (3.52)$$

3.3. LONGITUDINAL DYNAMIC STABILITY

We consider the Eqs(3.45) to (3.47) for the analyses of stability. It is assumed that $\eta'=0$ is taken for stick fixed dynamic stability. Therefore the elevator is kept fixed in trimmed position. We transfer the Eqs (3.45) to (3.47) from the time domain to Laplace domain to review the stability of the motion of the aircraft.

$$\begin{bmatrix} s+x_u & x_{\dot w}s+x_w & x_q s+\hat{g}_1 \\ z_u & [(1+z_{\dot w})s+z_w] & [(z_q-1)s+\hat{g}_2] \\ m & m_{\dot w}s+m & (s^2+m_q s) \end{bmatrix} \begin{bmatrix} \hat{u}(s) \\ \hat{w}(s) \\ \theta(s) \end{bmatrix} = 0 \qquad (3.53)$$

The only non-zero solution of the simultaneous equations requires that the determinant of the coefficient be zero. Thus

$$\begin{vmatrix} s+x_u & x_{\dot w}s+x_w & x_q s+\hat{g}_1 \\ z_u & [(1+z_{\dot w})s+z_w] & [(z_q-1)s+\hat{g}_2] \\ m & m_{\dot w}s+m & (s^2+m_q s) \end{vmatrix} = 0 \qquad (3.54)$$

$$As^4 + Bs^3 + Cs^2 + Ds + E = 0 \qquad (3.55)$$

where

$$A = 1+z_{\dot w} \quad , \quad B = x_u(1+z_{\dot w}) + z_w - x_{\dot w}z_u + (1+z_{\dot w})m_q + (1-z_q)m_{\dot w} \ ,$$

$$C = x_u z_w - x_w z_u + [x_u(1+z_{\dot w})+z_w-x_{\dot w}z_u]m_q + [x_u(1-z_q)+x_q z_u-\hat{g}_2]m_{\dot w} +$$

$$+ (1-z_q)m_w - [x_{\dot w}(1-z_q)+x_q(1+z_{\dot w})]m_u \ ,$$

$$D = (x_u z_w - x_w z_u)m_q + (\hat{g}_1 z_u - \hat{g}_2 x_u)m_{\dot{w}} + [x_u(1-z_q)+x_q z_u - \hat{g}_2]m_w -$$

$$- (x_w(1-z_q)+x_q z_w + \hat{g}_1(1+z_{\dot{w}}) - \hat{g}_2 x_{\dot{w}}]m_u ,$$

$$E = (\hat{g}_1 z_u - \hat{g}_2 x_u)m_w - (\hat{g}_1 z_w - \hat{g}_2 x_w)m_u .$$

The transient response of $\hat{u}$, $\hat{w}$, θ are in the same form because of the Eqs (3.53) . Their amplitudes are not necessary to review the stability. The changing of the transient response is given according to the roots of the Eqs(3.55) in Fig.3.4.

The equation (3.55) can be written with two quadratic factors.

$$As^4 + Bs^3 + Cs^2 + Ds + E = 0 \qquad (3.55)$$

$$(\alpha s^2 + \beta s + \gamma)(s^2 + \sigma s + \xi) \qquad (3.56)$$

$$\left.\begin{array}{l} \alpha = A \\ \beta + \alpha\sigma = B \\ \gamma + \beta\sigma + \alpha\xi = C \\ \beta\xi + \alpha\sigma = D \\ \gamma\xi = E \end{array}\right\} \qquad (3.57)$$

The coefficient D and E are usually small compared with C and C^2. Therefore σ and ξ are small compared β and α . Thus, we may take as a first approximation

$$\beta = B$$

$$\gamma = C$$

$$\sigma = E/\gamma = E/C$$

$$\xi = \frac{\gamma D - \beta E}{\gamma^2} = \frac{CD - BE}{C^2} \tag{3.58}$$

The Eq(3.56) can be rewritten with using the Eq(3.58) ,

$$(As^2 + Bs + C)(s^2 + \frac{CD - BE}{C^2} s + \frac{E}{C}) \tag{3.59}$$

In reference [27], Lin's Method has been used for obtaining the two quadratic factors. This method is done using the following steps ;

i) First trial divisor is found by dividing the coefficients of the last three terms by the coefficient of s^2 term.

ii) The characteristic function is divided by the first trial divisor.

iii) Second trial divisor is taken by dividing the coefficients of the remainder by the coefficient of s^2 term of it.

iv) The characteristic function is divided by the second trial divisor.

v) Each dividend is a quadratic factor.

BLAKELOCK [27] has given an example where the characteristic equation is given by

$$97.5s^4 + 79s^3 + 128.9s^2 + 0.998s + 0.667 = 0 \tag{3.60}$$

Two quadratic factors are found by using Lin's method as seen below,

$$(s^2 + 0.806s^2 + 1.311)(s^2 + 0.00466s + 0.0053) = 0 \quad (3.61)$$

We can apply the Eq(3.59) to Eq(3.60) and find the Eq(3.62).

$$(s^2 + 0.8102s^2 + 1.322)(s^2 + 0.00457s + 0.00525) = 0 \quad (3.62)$$

Eqs(3.61) and (3.62) are nearly same. Therefore we can use both of them.

The characteristic modes for nearly all aircraft in most flight conditions are two oscillations: one of short period with relatively heavy damping, the other of long period with very light damping. Short oscillation is called the "short period mode". It may require autostabilisation. It is the first quadratic factor in the Eq(3.59) . The long period oscillation is called the "phugoid mode". Its period is very long and the pilot can damp the phugoid successfully even if it is divergent or unstable.

In general, the exact values of the roots of the characteristic equation is not necessary to show whether or not the aircraft is stable . We shall use the short period mode in the next chapters, therefore we have shown how we could find the roots of it.

The stability of the characteristic equation can be reviewed with using Routh-Hurwitz method. Firstly the Routh Table is done.

$$As^4 + Bs^3 + Cs^2 + Ds + E = 0$$

$$\begin{array}{c|lll}
s^4 & A & C & E \\
s^3 & K_1 = B & D & \\
s^2 & K_2 = \dfrac{BC-AD}{B} & E & \\
s & K_3 = \left(\dfrac{BC-AD}{B} D - BE\right) \Big/ \left(\dfrac{BC-AD}{B}\right) & & \\
s^0 & K_4 = E & &
\end{array} \qquad (3.63)$$

Secondly, Hurwitz determinants are found,

$$\begin{array}{c|l}
s^4 & H_0 = A \\
s^3 & H_1 = K_1 = B \\
s^2 & H_2 = \dfrac{K_1}{K_2} = BC - AD \\
s & H_3 = \dfrac{K_2}{K_3} = BCD - AD^2 - B^2E \\
s^0 & H_4 = \dfrac{K_4}{K_3} = E\ (\ BCD - AD^2 - B^2E\ )
\end{array} \qquad (3.64)$$

Routh-Hurwitz criteria are

i) it is necessary that all Routh discriminations' (K_i's) sign must be same,

ii) it is efficiency that all Hurwitz determinations (H_i) must be bigger than zero.

In this condition,

$B > 0$,

$K_2 > 0$,

$K_3 > 0$,

$E > 0$,

$H_2 > 0$,

$H_3 > 0$.

B and D are always positive for a conventional aircraft. D is known that is very small compared with C. K_2, K_3, H_2 and H_3 are bigger than zero when C is positive. E value is a critical point of the stability. If E>0, there is no positive real root. Then all roots correspond either to subsidence or to oscillations. E=0 represents a zero root that is a state of natural stability. If E<0, there is minimum one positive real root. It represents the divergence.

3.4. LATERAL DYNAMIC STABILITY

The equations of the lateral motion, which are Eqs(3.49) to (3.51), are transferred from time domain to Laplace domain for the analysis of the stability. Then, they are expressed for the stick fixed lateral dynamics that we assume $\xi=\zeta=0$.

$$\begin{bmatrix} s+y_v & y_p s-\hat{g}_1 & (1+y_r)s-\hat{g}_2 \\ l_v & s^2+l_p s & e_x s^2+l_r s \\ n_v & e_z s^2+ns & s^2+n_r s \end{bmatrix} \begin{bmatrix} \hat{v} \\ \phi \\ \psi \end{bmatrix} = 0 \tag{3.65}$$

The determinant of the coefficient matrix must be zero for the solution of non-zero $\hat{v}$, ϕ and ψ. The determinant value equation is written

$$as^5 + bs^4 + cs^3 + ds^2 + es = 0 \tag{3.66}$$

The order of the Eq(3.66) can be reduced by eliminating s. The new equation is given by

$$as^4 + bs^3 + cs^2 + ds + e = 0 \tag{3.67}$$

where

$$a = 1 - e_x e_z \quad , \quad b = l_p + n_r - e_x n_p - e_z l_r + (1-e_x e_z)y_v \quad ,$$

$$c = l_p n_r - l_r n_p + (l_p+n_r-e_x n_p-e_z l_r)y_v + [e_x(1+y_r)-y_p]l_v -$$
$$- [1+y_r-e_x y_p]n_v \quad ,$$

$$d = (l_p n_r-l_r n_p)y_v + [n_p(1+y_r)-n_r y_p+\hat{g}_1-e_z\hat{g}_2]l_v -$$
$$- [l_p(1+y_r)-l_r y_p-\hat{g}_2+e_x\hat{g}_1]n_v \quad ,$$

$$e = (\hat{g}_1 n_r - \hat{g}_2 n_p) l_v - (\hat{g}_1 l_r - \hat{g}_2 l_p) n_v . \qquad (3.68)$$

The Eq(3.67) has a root zero, which has been eliminated for simplicity. This root represents a neutral stability. The corresponding solution of it is given by

$$\hat{v} = \rho_1 \qquad (3.69)$$

$$\phi = \rho_2 \qquad (3.70)$$

$$\psi = \rho_3 \qquad (3.71)$$

where ρ_1, ρ_2 and ρ_3 are constant.

When we calculate $\hat{v}$, ϕ and ψ from Eq(3.65), they become

$$\hat{v} = 0 \qquad (3.72)$$

$$\hat{g}_1\phi + \hat{g}_2\psi = 0 \qquad (3.73)$$

The simplified equations for the Eqs(3.45) to (3.47) are also used for the Eq(3.73)

$$\phi \cos\Theta_e + \psi \sin\Theta_e = 0 \qquad (3.74)$$

The component of gravity Y_g along Oy is zero, therefore there is no sideslip in the lateral motion, but the angles of roll and yaw are small constant. The lateral aerodynamic forces and moments depend on the velocity of the sideslip and the angular velocities of bank and yaw but not the angles of bank and yaw. The angles of bank and yaw do not depend on the aerodynamic characteristics in the neutral stability.

On the other hand, if Eq(3.67) is considered by the relations between the other roots and the coefficient. For conventional aircraft, the coefficient 'a' is approximately unity, the coefficient 'b' is much bigger than 'a'. 'e' is much smaller than 'd'. In this situation, we can use some approximations for to find the roots values according to reference [26].

First approximation :

The characteristic equation (3.67) has a large negative root because of the coefficient b.

$$s \cong -b \tag{3.75}$$

Second approximation :

The Eq(3.67) has also a very small root according to small 'e' and much bigger 'd' compared with 'e'.

$$s \cong -e/d \tag{3.76}$$

Third approximation :

The other roots are a pair of complex roots. Its oscillation increases or decreases due to the sign of the real part of complex roots that is positive or negative.

$$s = u \mp iv \tag{3.77}$$

We use these three approximations and write the characteristic equation,

$$(s + b)(s + \frac{e}{d})(s + r + iv)(s + r - iv) = 0 \tag{3.78}$$

The motions in the Eq(3.78) are called Dutch roll or lateral oscillation, the rolling subsidence and the slow spiral motion in respect of the Eq(3,75), the Eq(3.76) and the Eq(3.77).

The Routh-Hurwitz criteria is applied as well as the equation of the longitudinal motion. Tables are the same but only small letters are used instead of capital letters.

The Routh discrimination are,

$$\begin{array}{c|lll}
s^4 & a & c & e \\
s^3 & K_1 = b & d & \\
s^2 & K_2 = \dfrac{bc-ad}{b} & e & \\
s & K_3 = \left(\dfrac{bc-ad}{b} d - be\right) \Big/ \left(\dfrac{bc-ad}{b}\right) & & \\
s^0 & K_4 = e & &
\end{array} \tag{3.79}$$

The Hurwitz determinants are

$$
\begin{array}{c|l}
s^4 & H_0 = a \\[2ex]
s^3 & H_1 = K_1 = b \\[2ex]
s^2 & H_2 = \dfrac{K_1}{K_2} = bc - ad \\[2ex]
s & H_3 = \dfrac{K_2}{K_3} = bcd - ad^2 - b^2e \\[2ex]
s^0 & H_4 = \dfrac{K_4}{K_3} = e\,(\,bcd - ad^2 - b^2e)
\end{array}
\tag{3.80}
$$

The critical points are H_4 and K_4. K_4 depends on only e. H_4 depends on e and $R=bcd-ad^2-b^2e$. If $e<0$, the slow spiral motion is divergence. If $R<0$, Dutch roll is an increasing oscillation. If $R>0$, Dutch roll is a damped oscillation.

In reference [27], another example has been given for the lateral motion and its characteristic equation is shown below,

$$s^5 + 2.44s^4 + 2.51s^3 + 3.68s^2 - 0.0152s = 0 \tag{3.81}$$

Its solution has also been given by

$$s\,(s^2 + 0.380s + 1.813)\,(s + 2.09)\,(s - 0.004) = 0 \tag{3.82}$$

where the slow spiral is $(s-0.004)$ and roll subsidence $(s+2.09)$. We can find the the slow spiral and the roll subsidence with by the approximation.

The slow spiral : $(s + \frac{e}{d}) = (s - 0.0041)$ (3.83)

The roll subsidence : $(s + b) = (s - 2.44)$ (3.84)

3.5. CONCLUSION

An attempt has been made to explain the aircraft dynamics and stability by using the equation of the aircraft motion. Some terms have been neglected according to the references. The characteristic equation has been derived for the purpose of the stability. Therefore each aerodynamic derivative was not discussed in detail.

The longitudinal dynamic motion was divided into two modes ; the short time mode and phugoid mode. The short time mode is important for the control because it may not be controlled by the pilot without autostabilization. However the phugoid mode can be controlled by the pilot, even if it is divergent or unstable.

The lateral motion has also been reviewed for the stability. The slow spiral mode is not very important, because this root value is very small and controllable by the pilot. The roll subsidence is not dangerous for the stability that it is recognized from its name. Dutch roll must be considered for the lateral motion. It can be a divergence oscillation or a subsidence oscillation.

The approximations, which are used for the analysis of the stability of the longitudinal and the lateral motion, have been supported the numerical examples.

The longitudinal short period mode and the Dutch roll are most important for pilot handling and it is the identification of these parameters which is required for adaptive or optimal control.

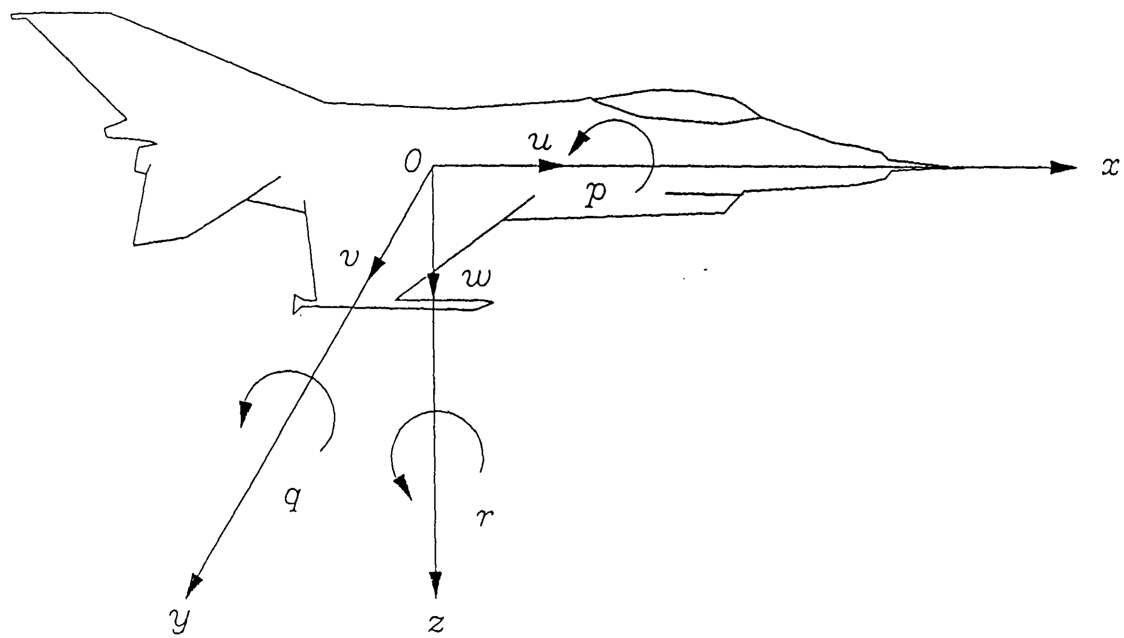


Fig.3.1. Body axes of an aircraft.

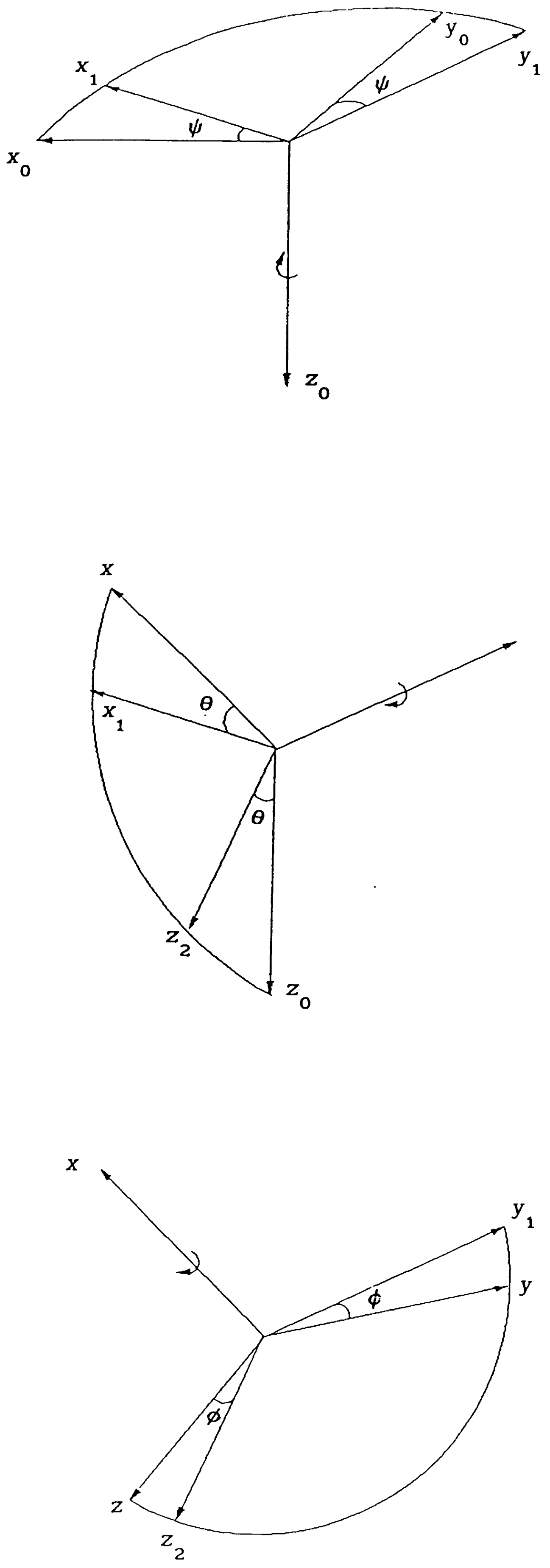


Fig.3.2. The determination of angles.

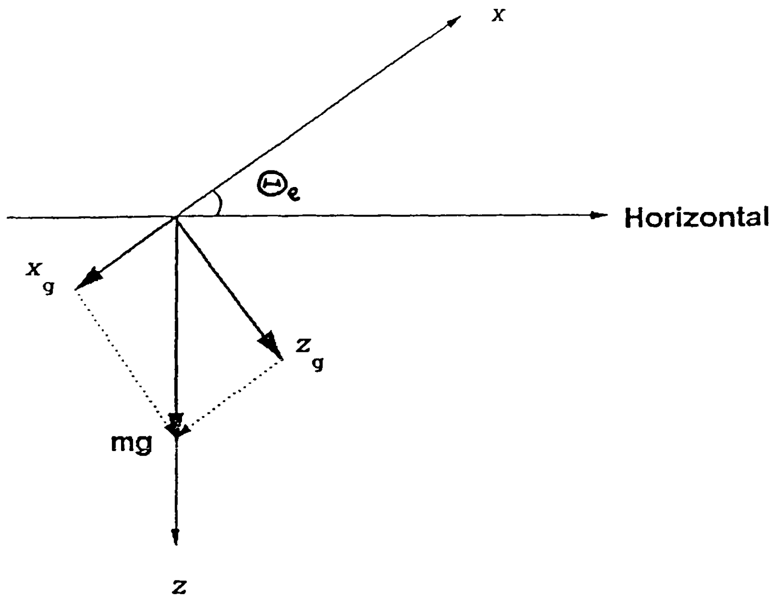


Fig.3.3. The gravitational forces.

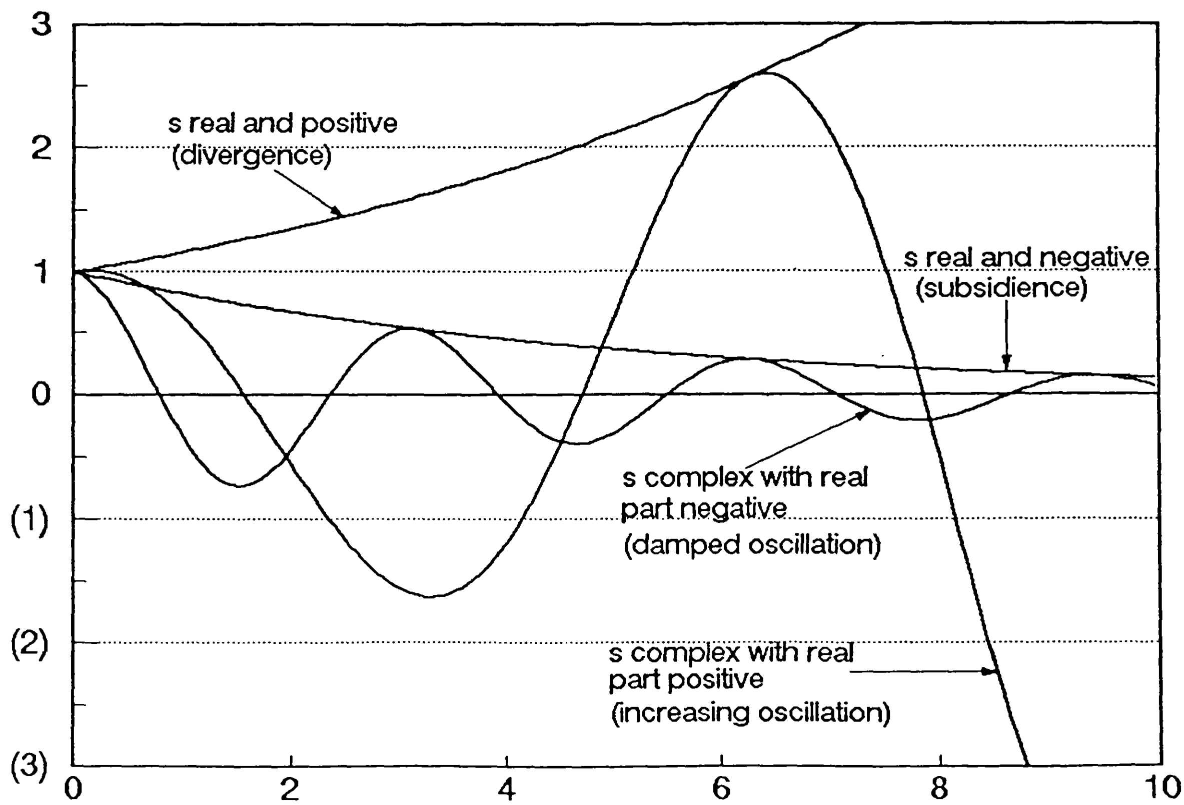


Fig.3.4. The responses of characteristic equation

CHAPTER 4. OPTIMAL ADAPTIVE CONTROL

4.1. INTRODUCTION

Adaptive control can be defined as a kind of control where the system performance approximates to the desired system performance by the application of the suitable control signals. The aim is to approximate the process output to the desired output in the minimum time. In this chapter, how the adaptive systems are realized will be discussed. Two main approaches will be discussed: model reference adaptive systems and self-tuning controllers.

The model reference systems control strategy, its development, its disadvantage and its stability will be discussed briefly. They will be given by the equations and the block diagrams.

The self-tuning regulators will be the bases on the adjustment mechanism of Model Reference Adaptive System (MRAS). Self-tuning controllers are studied in two methods namely, explicit and implicit. They will be explained in detail. The optimal controller methods will be studied by Linear Quadratic (LQG) methods which requires either the solution of the Riccati equation or analysis of some special spectral factorization methods.

4.2. MODEL REFERENCE ADAPTIVE SYSTEMS

This method was originally developed by Whitaker and his colleagues in 1958 [1]. This original Model Reference Adaptive System (MRAS) is given by Fig.4.1.. The MRAS consists of a process, a reference model, a regulator and adjustment mechanism , which adjusts the parameters of the regulator according to the difference between the model outputs and the process outputs. The model is chosen to satisfy the desired performance of the process. The main problem in MRAS is the design of the regulator and the adjustment mechanism. The regulator tries to approximate the process output to the model output by changing its structure, which contains the adjustable parameters. This regulator operation is called the model-following. Åström [28] has used the pole placement design to solve the model-following problem.

4.2.1. Model-following Design

Single input , single output system transfer function is given below as:

$$y = \frac{B}{A} u \tag{4.1}$$

where y is the output signal, u is the control input and A, B are polynomials which are written in the Laplace domain. On the other hand, the reference model can be written as Eq(4.2)

$$y_m = \frac{B_m}{A_m} u_c \tag{4.2}$$

where u_c is the common reference input signal, A_m and B_m are polynomials. In reference [28], the regulator has been described as seen

$$R\,u = T\,u_c - Sy \tag{4.3}$$

where R, T and S are polynomials. The control signal of the process can be rewritten from Eq(4.3) ;

$$u = \frac{T}{R} u_c - \frac{S}{R} y \tag{4.4}$$

when u is eliminated in the Eq(4.1) the closed loop system can be found as:

$$(AR + BS)\ y = BT\ u_c \tag{4.5}$$

$$y = \frac{BT}{AR + BS} u_c \tag{4.6}$$

Since eq(4.6) must be equal to Eq(4.2)

$$\frac{B_m}{A_m} = \frac{BT}{AR + BS} \tag{4.7}$$

There are many approaches to choose the polynomials R, S and T. In reference [28], Åström has suggested the pole-zero cancellation for perfect model-following. The open-loop process zeros, given by B=0, will also be the closed-loop zeros unless they are cancelled by corresponding closed-loop poles. Only, the stable zeros of B may be cancelled, the polynomial is factored as

$$B = B^{-} B^{+} \tag{4.8}$$

where B^{+} includes zeros that can be cancelled, and B^{-} includes the remaining factor of B. On the other hand, B^{-} must be factors of B_m otherwise the solution is not possible. Hence,

$$B_m = B^{-} B_m' \tag{4.9}$$

Since B^{+} is a factor of $AR + BS$, it follows that it is also a factor of R.

$$R = B^{+} R' \tag{4.10}$$

Eq(4.7) can be rewritten as

$$\frac{B^{+}B^{-}T}{AB^{+} + B^{+}B^{-}S} = \frac{B^{-}B_m'}{A_m}$$

$$\frac{B^{+}B^{-}T}{B^{+}(AR'+B^{-}S)} = \frac{B^{-}B_m'}{A_m}$$

$$\frac{T}{AR'+B^{-}S} = \frac{B_m'}{A_m} \tag{4.11}$$

From Eq(4.11) one can conclude that A_m is a factor of $AR' + B^{-}S$. The observer polynomial A_o, which may be used to cancel the desired stable process poles, is also a factor of $AR' + B^{-}S$. Thus, the new equation is obtained as :

$$AR' + B^{-}S = A_oA_m \tag{4.12}$$

$$T = B_m'A_o \tag{4.13}$$

The closed-loop characteristic equation becomes

$$AR + BS = B^{+}A_oA_m \tag{4.14}$$

In reference [28], Åström has proposed that there exist a solution to Eq(4.14) which gives a continuous-time or discrete-time control law :

$$\deg A_o \geq 2 \deg A - \deg A_m - \deg B^{+} - 1 \tag{4.15}$$

$$\deg A_m - \deg B_m \geq \deg A - \deg B \qquad (4.16)$$

4.2.2. ADJUSTMENT MECHANISM

Many different systems have been used for the adjustment mechanism. We will discuss only the gradient approach because many of MRAS have been developed using the gradient approach.

The Gradient Approach :

This approach was used in the original MRAS by Whitaker and his colleagues. It is also called the M.I.T. rule because it was done at Massachusetts Institute of Technology. Let e denote the error between the process output and the model output and θ the parameters. The loss function is defined as

$$J\ (\ \theta\ ) = \frac{1}{2}\ e^2 \qquad (4.17)$$

To reduce the loss it is reasonable to change the parameters in the direction of the negative gradient of J, therefore

$$\frac{d\theta}{dt} = -\ k\ \frac{\partial J}{\partial \theta} = -\ k\ e\ \frac{\partial e}{\partial \theta} \qquad (4.18)$$

The Eq(4.18) can be written, if the parameters change much more slowly than the other variables. The M.I.T. rule performs well when the parameter k is small. Otherwise, the M.I.T. rule can give an unstable closed-loop. But the M.I.T. rule can become more convenient by using some modified adjustment rules, which have been obtained from stability theory. Some approaches are used to prevent instability . For example, one approach is to make a normalization and replace the M.I.T. rule by [5]

$$\frac{d\theta}{dt} = -k \frac{e\dfrac{\partial e}{\partial \theta}}{\alpha + \left[\dfrac{\partial e}{\partial \theta}\right]^{T}\left[\dfrac{\partial e}{\partial \theta}\right]} \tag{4.19}$$

where $\alpha > 0$ is replaced to avoid a possible division by zero. In addition, a limit is determined to guarantee stability.

$$\frac{d\theta}{dt} = -k \,\mathrm{sat}\left[\frac{e\dfrac{\partial e}{\partial \theta}}{\alpha + \left[\dfrac{\partial e}{\partial \theta}\right]^{T}\left[\dfrac{\partial e}{\partial \theta}\right]},\ \beta\right] \tag{4.20}$$

$$\mathrm{sat}\,(x, \beta) = \left[\begin{array}{ll} -\beta & x < \beta \\ x & |x| \le \beta \\ \beta & x > \beta \end{array}\right.$$

Parks [29] has reviewed the stability of the M.I.T. design by using a Lyapunov function. He has also developed a stable approach as

$$\frac{d\theta}{dt} = - k \, u_c e \qquad (4.21)$$

Fig.4.2. and Fig.4.3. show the M.I.T. rule and Lyapunov approach systems. The MRAS, which is given by Fig.4.3. needs a compensator for which the closed loop transfer function is strictly positive real. Its block diagram can be given as in Fig.4.4.. According to the passivity theorem, the compensator will include derivatives [30] when the degree of the closed loop polynomial is more than 1. The augmented error model has been developed to avoid the derivation of the signal [31]. An augmented error signal, which is obtained using the error and the compensator, is used instead of the error signal as in Fig.4.5.. An adjustment law can be obtained from the general transfer function. The transfer function can be factored as:

$$G = G_1 G_2 \qquad (4.22)$$

where G_1 is strictly a positive real function and G_2 is the remaining factor of G. The parameter adjustment rule is then

$$\frac{d\theta}{dt} = - k \, \varepsilon \, (G_2 u_c) \qquad (4.23)$$

$$\varepsilon = e + G_1 (\theta \, G_2 u_c) - G (\theta \, u_c) \qquad (4.24)$$

The block diagram of this adjustment law is shown in Fig.4.6..

4.3. SELF TUNING REGULATOR

The purpose of design of a self-tuning regulator (STR) is same as the MRAS. But the method is different. In the STR, the regulator automatically adjust itself due to the output of the process. Fig.4.7. helps to explain the system. The regulator structure depends on the regulator design algorithm, which determines the regulator parameter to control the desired output of the process. The design algorithm requires the process parameters. When the process parameters are known, the regulator design algorithm specifies a set of desired controller parameters. Therefore, the unknown parameters are estimated on-line by using a recursive estimation method. Many different estimation schemes can be used as we discussed in Chapter 2. The design algorithm simply accepts the results of the estimation and ignores the uncertainties of the estimation. This is called the *certainty equivalence principle* [6].

The STR are classified according to the estimation of the regulator and the process parameters. If the process parameters are estimated and the regulator parameters are calculated by the design algorithm, this system is called *explicit.* If the controller parameters are estimated directly this is called *implicit* self-tuning control. Its block diagram is given in the Fig.4.8.. Fig.4.7. represents the explicit self-tuning control system.

4.3.1. Explicit Self-Tuning Regulators

4.3.1.1. Pole-Placement Method

An explicit self-tuning controller design algorithm is used for pole-placement by using the estimation result. The regulator structure is exactly the same as in section 4.2.1.. The process is described by the single input, single output system as:

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{4.25}$$

where y is the output, and u is the input of the process. A and B are polynomials in the forward shift operator q. The design algorithm is obtained in the following form [5]

Data : The desired system transfer operator $\frac{B_m}{A_m}$ and a desired observer polynomial A_o are given

Step 1 : The coefficients of transfer polynomials A, B and C are estimated by the on-line method.

Step 2 : The polynomials A and B are replaced and solve Eq(4.12) to obtain R' and S. R and T are calculated by the Equations (4.10) and (4.13).

Step 3 : The control signal of the process is calculated from the Eq(4.4).

The Steps 1, 2, and 3 are repeated for each sampling period.

This method is not suitable for the non-minimum phase system, because the control input goes to infinite value. But this method can still be used without zero cancellation. The Eq(4.7) is rewritten as

$$\frac{y}{u_c} = \frac{BT}{AR + BS}$$

The open-loop system zeros, which are determined by the dead-time, are remained on the closed-loop system by selecting T=1. "$AR + BS$" consists of the closed-loop system poles that can be done to be equal to the desired system poles.

$$AR + BS = A_o A_m$$

$$R = \frac{A_o A_m - BS}{A} \qquad (4.26)$$

An appropriate S brings all poles of R and all zeros of R on the left hand side of the complex plane (inside the unit circle in discrete-time). In addition, the steady state of the closed-loop system output can be made close to the steady state of the desired system output by recalculating T.

$$T = \frac{\lim_{t \to \infty} \frac{B_m}{A_m}}{\lim_{t \to \infty} \frac{B}{A_m}}$$

$$T = \frac{\prod zeros(B_m)}{\prod zeros(B)} \tag{4.27}$$

The pole placement algorithm for the minimum phase can be used for the non-minimum phase system by using the Eqs(4.26) and (4.27). The Eqs(4.26) and (4.27) are also valid for unstable system with an appropriate S.

Example 4.1.

The aircraft attack angle function is modelled as

$$\alpha = \frac{b_o s - b_1}{s^2 + a_1 s + a_o}\, \eta$$

where α is the angle of attack and η is the angular displacement of the elevator. The desired model is given as Appendix 1.14

$$\alpha_m = \frac{1.17\text{x}10^{-3} s - 0.42}{s^2 + 1.938\, s + 1.203}\, \eta$$

where η is assumed to be unit step function. The steady state values of the process output and desired output can be determined as follows:

$$\alpha_{fin} = \lim_{t \to \infty} \alpha(t) = \lim_{s \to 0} s\, \alpha(s) = \lim_{s \to 0} s \frac{b_o s - b_1}{s^2 + a_1 s + a_o} \frac{1}{s}$$

$$\alpha_{fin} = -\frac{b_1}{a_1}$$

$$\alpha_{mfin} = \lim_{t \to \infty} \alpha_m(t) = \lim_{s \to 0} s\, \alpha_m(s)$$

$$= \lim_{s \to 0} s \frac{1.17 \times 10^{-3} s - 0.42}{s^2 + 1.938\, s + 1.203} \frac{1}{s} = -0.357$$

$$T = \frac{0.357\, a_o}{b_1}$$

The value of b_1 may not be the same as the model value but it is not very different from the model value. However, taking $S = 1$ can guarantee that the Eq(4.26) will not have any zeros or poles in the right half side of complex plane. R is calculated from the Eq(4.26) and the outputs and the control input are shown in Fig.4.9. This example has been used and resulted in an unstable system result as given Fig.4.10.

4.3.2. Implicit Self-Tuning Controller

4.3.2.1. Pole-Placement Method

The design of the regulator parameters in the explicit method consumes an additional time for the identification . The regulator parameter can be estimated by using the equation where the system is described by S and R. This equation can be obtained by multiplying both sides of Eq(4.12) by the output y.

$$
\begin{aligned}
y\,(A_o A_m) &= y\,(AR' + B^- S) \\
&= R'Ay + B^- Sy \\
&= R'Bu + B^- Sy + R'Ce \\
&= B^- (Ru + Sy) + R'Ce \qquad (4.28) \\
&= \bar{R}u + \bar{S}y + R'Ce \qquad (4.29)
\end{aligned}
$$

$\bar{R} = B^- R \quad$ and $\quad \bar{S} = B^- S$

where all variables and polynomials were explained in section 4.2.1.. Direct or implicit self-tuning regulator algorithm can be given by the following steps,

Step 1 : The parameters of the polynomials $\bar{B}$ and $\bar{S}$ are identified from the Eq(4.29).

Step 2 : R and S are obtained from $\bar{B}$ and $\bar{S}$.

Step 3 : The control signal is calculated from Eq(4.4)

Steps 1, 2, and 3 are repeated at each sampling period.

This algorithm can be applied to both of the Eqs(4.28) and (4.29). The identification of the parameters by using Eq(4.29) involves more calculation than using Eq(4.28). The Eq(4.28) represents a nonlinear model unless B^- is a constant [5]. This algorithm avoids the nonlinear estimation problems. Hence, B^- is assumed to be constant

$(B^{-}=b_o)$. In this case the estimation of the eq(4.28) can be simplified as below

$$y(t)(A_oA_m) = b_o(Ru(t) + Sy(t)) + R'Ce(t)$$

$$y(t) = \frac{b_o}{A_oA_m}(Ru(t) + Sy(t)) + \frac{R'}{A_oA_m}Ce(t) \qquad (4.28)$$

$$y(t) = R^{*}u(t) + S^{*}y(t) + \frac{R'C}{A_oA_m}e(t) \qquad (4.30)$$

on the other hand, the Eq(4.6) can be rewritten as:

$$\frac{y}{u_c} = \frac{BT}{AR + BS}$$

$$= \frac{b_oB^{+}T}{AR + BS} = \frac{B_m}{A_oA_m} = \frac{y_m}{u_c}$$

where B^{+} includes all zeros that can be cancelled. In this case ;

$$A_oA_my = b_oTu_c$$

$$y = \frac{b_oT}{A_oA_m}u_c = T^{*}u_c \qquad (4.31)$$

The following algorithm can be used for the simplified equations.

Step 1 : The parameters of the polynomials R^{*}, S^{*} and T^{*} are identified from the Eqs(4.30) and (4.31).

Step 2 : The control signal is calculated from Eq(4.4)

Steps 1 and 2 are repeated at each sampling period.

These direct adaptive pole placement models which are given by Eq(4.28), Eqs(4.30) and (4.31) are suitable only for minimum phase and stable system. The pole placement method for the non-minimum phase systems has been developed by Elliot [32]. He has studied the system with a partial state and the non-adaptive linear control strategy. The transfer function is determined by the Eq(4.1) as:

$$A(s)y(t) = B(s)u(t) \tag{4.32}$$

The following *priori* information is assumed about the process.

i) $A(s)$ is a monic polynomial of degree n.

ii) deg $B \le$ deg A

iii) $A(s)$ and $B(s)$ are relatively prime so that the Eq(4.32) is a minimal realization.

This system can be represented by using the partial state z as seen below

$$A(s)z(t) = u(t) \tag{4.33}$$

$$y(t) = B(s)z(t) \tag{4.34}$$

Initially, a nonadaptive linear control law which can be used to assign the poles of the Eq(4.32) that $A(s)$ and $B(s)$ are assumed to know is derived. Consequently it will be converted to an adaptive strategy assuming $A(s)$ and $B(s)$ to be unknown. An observable polynomial A_o of degree n is defined and the considered compensator for the Eq(4.32) is characterized by the equations

$$A_o(s)G(t) = S(s)y(s) + R(s)u(t) \tag{4.35}$$

$$u(t) = G(t) + v(t) \tag{4.36}$$

where $v(t)$ is an external reference input, and $S(s)$ and $R(s)$ are polynomials of degree n-1 of the form

$$S(s) = \sum_{i=0}^{n-1} S_i s^i \tag{4.37}$$

$$R(s) = \sum_{i=0}^{n-1} R_i s^i \tag{4.38}$$

Putting Eqs(4.33) to (4.36) in to Eq(4.32)

$$[\ A_o(s)A(s) - S(s)B(s) - R(s)A(s)\ ]\ z(t) = A_o(s)v(t) \tag{4.39}$$

$$y(t) = B(s)z(t) \tag{4.40}$$

The polynomial $A_m(s)$ represents the desired closed-loop poles. These can be assigned in order that $S(s)$ and $R(s)$ satisfy

$$S(s)B(s) + R(s)A(s) = A_o(s) \; [\; A(s) - A_m(s) \;] \tag{4.41}$$

When Eq(4.41) holds Eq(4.40) simplifies to

$$A_m(s)A_o(s)z(t) = A_o(s)v(t) \tag{4.42}$$

$$y(t) = B(s)z(t) \tag{4.43}$$

In this case the transfer function is

$$\frac{y(t)}{v(t)} = \frac{B(s)}{A_m(s)} \tag{4.44}$$

This scheme can be slightly simplified if the process is known to be characterized by a strictly proper transfer function that is deg $B \le$ deg $A - 1$. In this condition, $A_o(s)$ can be chosen to be of degree n-1 and $R(s)$ can be chosen to be of degree n-2. When the plant is unknown, the following adaptive version of the Eqs(4.35) and (4.36) can be implemented as follows:

$$A_o(s)\tilde{y}_i = s^i y(t), \qquad 0 \le i \le n-1 \tag{4.45}$$

$$A_o(s)\tilde{u}_i = s^i u(t), \qquad 0 \le i \le n-1 \tag{4.46}$$

$$u(t) = \sum_{i=0}^{n-1} (\; S_i^*(t)\tilde{y}(t) + R_i^*(t)\tilde{u}(t) \;) + v(t) \tag{4.47}$$

For the resulting closed-loop system to converge to Eq(4.39) thee adjustable gains S_i^*, R_i^* must satisfy

$$\lim_{t\to\infty} S^*_i(t) = S_i \qquad 0 \le i \le n-1 \tag{4.48}$$

$$\lim_{t\to\infty} R^*_i(t) = R_i \qquad 0 \le i \le n-1 \tag{4.49}$$

where S_i, R_i satisfy Eqs(4.38) and (4.39). The *Bezout* identity is defined by as:

$$P(s)B(s) + Q(s)A(s) = 1 \tag{4.50}$$

where $P(s)$ and $Q(s)$ are the filter polynomials of the output and the input.

$$P(s) = \sum_{i=0}^{n-1} P_i s^i \quad , \qquad Q(s) = \sum_{i=0}^{n-1} Q_i s^i \tag{4.51}$$

When Eq(4.50) is substituted into Eq(4.41)

$$S(s)B(s) + R(s)A(s) = A_o(s)\ \{\ A(s) - [P(s)B(s) + Q(s)A(s)]A_m\ \} \tag{4.52}$$

The degree of the right side of Eq(4.52) is greater than the left side to satisfy the Eqs(4.41) and (4.42). The coefficients of $R(s)$, $S(s)$, $P(s)$ and $Q(s)$ are estimated to establish the adaptive control which is obtained by Eq(4.41). The coefficients are identified by the filtering signals that depend on the known and the measurable values. The relationships between the filter function and the filtered signals are defined as:

$$\bar{u}(t) = \frac{1}{F(s)} u(t)$$

$$\bar{y}(t) = \frac{1}{F(s)} y(t)$$

$$\bar{z}(t) = \frac{1}{F(s)} z(t)$$

$$\bar{u}_i(t) = s^i \, \bar{u}(t), \qquad \bar{y}_i(t) = s^i \, \bar{y}(t) \, , \qquad 0 \le i \le n-1$$

$\bar{z}$ is not a real system state therefore it is eliminated by multiplying Eq(4.52) and using Eqs(4.33) and (4.34). We have

$$\begin{aligned} S(s)\bar{y}(t) + R(s)\bar{u}(t) = A_o(s)\bar{u}(t) & - P(s)A_o(s)A_m(s)\bar{y}(t) \\ & - Q(s)A_o(s)A_m(s)\bar{u}(t) + \gamma_1 \end{aligned} \qquad (4.53)$$

where γ_1 is an exponentially decaying signal for the nonzero initial conditions. The equation (4.53) can be written in the estimation form

$$\theta\bar{x}(t) = A_o(s)\bar{u}(s) + \gamma_1(t)$$

$$\theta = [\ S_1 \ \dots \ S_{n-1},\ R_1 \ \dots R_{n-1},\ P_1 \ \dots P_{n-1},\ Q_1 \ \dots Q_{n-1} \ ]$$

$$\begin{aligned} \bar{x}(t) = [\ & \bar{y}_o(t),\ \dots,\ \bar{y}_{n-1}(t),\ \bar{u}_o(t),\ \dots,\ \bar{u}_{n-1}(t),\ A_o(s)A_m(s)\bar{y}_o(t), \\ & \dots, A_o(s)A_m(s)\bar{y}_{n-1}(t),\ A_o(s)A_m(s)\bar{u}_o(t),\ \dots,\ A_o(s)A_m(s)\bar{u}_{n-1}(t) \ ] \end{aligned}$$

(4.54)

The direct adaptive algorithm for non-minimum phase systems can be given as follows:

Step 1) The coefficients of the *S*, *R*, *P* and *Q* polynomials are estimated via the Eq(4.54).

Step 2) The control input u is found from the Eq(4.47).

Step 1 and step 2 are repeated for each sampling period.

4.3.2.2. Minimum Variance

The minimum variance method has been designed for a stochastic system whose output must follow the e. It is assumed that the reference input $u_c = 0$ and also the optimal observer polynomial $A_o = C$. The minimum variance algorithm for the minimum phase system is given as:

$$R^{*}u(t) + S^{*}y(t) = 0$$

$$u(t) = \frac{S^{*}}{R^{*}}\ y(t) \tag{4.55}$$

R^{*} and S^{*} have been defined in the Eq(4.30). In reference [2] , the identity diophantine equation has been described by

$$C = AR + q^{-k}S \tag{4.56}$$

with k = deg A - deg B . Multiplying both side of Eq(4.56) by the output term $y(t)$ we obtain the direct adaptive estimation equation.

$$Cy(t) = ARy(t) + q^{-k}Sy(t)$$

$$= R\ (\ Bu(t-k) + Ce(t)\ ) + q^{-k}Sy(t)$$

$$y(t) = \frac{1}{C^*}(\ R^*u(t-k) + S^*y(t-k)\ ) + R_1^*e(t) \qquad (4.57)$$

$$C = BC^*, \qquad R = BR^*, \qquad S = BS^*$$

The condition k = deg A - deg B must be satisfied to cancel all zeros If the process is non-minimum phase, no zeros are cancelled and k is equal to deg A. This controller can be called a *Moving Average Controller.* The minimum variance or the moving average controller algorithm can be applied in the following form,

Step 1) The coefficients of the polynomials R^* and S^* are identified from the Eq(4.57).

Step 2) The control signal is calculated from the Eq(4.55)

Repeat the step 1 and step 2 for each sample period.

If the system delay k_o is smaller than half of the sample time, the minimum variance controller can work with a minimum phase system [5].

4.3.2.3. Generalized Minimum Variance

The minimum variance method has been extended by using the filtered signal. The filtered output has been defined by

$$y_f = \frac{1}{P} y(t)$$

where P is an arbitrary stable polynomial. In addition,, the reference input u_c and a user-specified transfer function Q has been introduced by Clarke and Gawthrop [33].

$$u(t) = [\ u_c - (\ R^* u(t) + S^* y(t)\ )\ ]\ /\ Q(q^{-1}) \qquad (4.58)$$

One objective of the use of Q is to reduce the excessive control activity associated with minimum variance control with small sample time, for if $u_1(t)$ is the required to achieve exact model-following the Eq(4.58) can be shown to produce a control [34]

$$u_1(t) = (\ R^* + \frac{Q}{r_o}\ )\ u(t) \qquad (4.59)$$

When the Eq(4.58) is asserted, the closed-loop satisfies

$$(\ PB + QA\ )y(t) = Bu_c(t-k) + (\ QA_o + R^* B\ )\ e(t)\ /\ A_o$$

$$(\ PB + QA\ )u(t) = Au_c(t) - S^* e(t)\ /\ A_o \qquad (4.60)$$

If $P = 1$, $Q = 0$ and u_c=0 are assumed, it can easily be seen that the generalized minimum variance becomes the minimum variance. The loss function of the generalized minimum variance is :

$$I = E \{ P^2 (y - u_c)^2 + Q^2 u^2 \} \tag{4.61}$$

Clarke and Gawthrop used $P = 1$ and $Q = \lambda$. In this case the model and its loss function becomes

$$(B + \lambda A)y(t) = Bu_c(t-k) + (\lambda C + R^* B) e(t)$$

$$(B + \lambda A)u(t) = Au_c(t) - S^* e(t) \tag{4.62}$$

$$I = E \{ (y - u_c)^2 + \lambda r_o u^2 \}$$

The stabilization of the non-minimum phase system can be achieved by the appropriate choice of P and Q which are determined by the polynomial of ($PB + QA$). The unstable system can also be stabilized with a suitable choice of P and Q.

4.4.4. Linear Quadratic Self-Tuning Controllers

This method has been developed to obtain the optimal controller via the cost function of the adaptive control system. The cost function can be given by

$$J = \sum_{i=t}^{t+N} (y(i) - y_m(i))^2 + \lambda u^2(i) \tag{4.63}$$

The first optimal regulator has been introduced with state-space realization. The system has been considered by

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k) \tag{4.64}$$

It can be transferred in the state-space observable form as [35]

$$\begin{aligned} x(t+1) &= A_I x(t) + B_I u(t) + Ke(t) \\ y(t) &= C_I x(t) + e(t) \end{aligned} \tag{4.65}$$

where

$$A_I = \begin{bmatrix} -a_1 & & \\ -a_2 & I_{n+k-1} & \\ \cdot & & \\ -a_n & & \\ \cdot & & \\ \cdot & & \\ 0 \;\cdots & & 0 \end{bmatrix}$$

$$B_I = [\; \underbrace{0 \;\; \dots \;\; 0}_{k-1} \;\; b_0 \;\; b_1 \;\; \dots \;\; b_n \;]^T$$

$$K = [\; c_1 - a_1 \;\; c_2 - a_2 \;\dots\; c_n - a_n \;\; 0 \;\; \dots \;\; 0 \;]^T$$

$$C_I = [\; 1 \;\; 0 \;\; \dots \;\; 0 \;]^T$$

$$n = \max(\deg A, \deg B, \deg C)$$

The model (4.65) is called *innovation model* and K is the optimal steady-state gain in a Kalman. The cost function in state-space form is obtained as;

$$J = x^{T}(t+N)Sx(t+N) + \sum_{i=t}^{t+N} x^{T}(i)Qx(i) + \lambda u^{2}(i)$$

$$S = Q = C_{I}^{T}C_{I} \tag{4.66}$$

The optimal controller is given by

$$u(t) = -L(t)x(t)$$

where $L(t)$ is Kalman control gain which can be obtained from the iterative solution of the Riccati equation :

$$L(t) = (\lambda + B_{I}^{T}P(t+1)B_{I})^{-1}B_{I}P(t+1)A_{I}$$

$$P(t) = Q + A_{I}^{T}P(t+1)A_{I} - A_{I}^{T}P(t+1)B_{I}(\lambda + B_{I}^{T}P(t+1)B_{I})^{-1}B_{I}P(t+1)A_{I} \tag{4.67}$$

$$P(t+N) = S$$

The Riccati equation is iterated backwards to converge at every sampling instant by starting from terminal conditions $P(t+N)=S$. In this case, the stage number N must be chosen greater than time-delay k, otherwise cost minimization is meaningless. It has been shown [5] that a limiting controller is

$$\bar{L} = \lim_{t \to \infty} L(t)$$

The closed-loop characteristic equation is

$$P(q) = \det\ (\ q - A_I + B\bar{L}\ ) = 0 \tag{4.69}$$

The LQG method via the solution of the Riccati equation can be applied to the adaptive control by the given specific λ value and the following steps,

Step 1) The coefficients of the polynomials A, B and C are estimated from the eq(4.64).

Step 2) The Riccati equation is solved and $\bar{L}$ is obtained.

Step 3) The control signal is calculated from $u(t) = -L^T(t)\ x(t)$.

These steps are repeated for each sampling period.

Different approaches were used to solve the Riccati equation by some researchers. On the other hand, Åström and Wittenmark have developed the spectral factorization method for the design of an LQG self-tuner. They have considered the Eq(4.25) when $A(q)$ and $C(q)$ have degree n. $C(q)$ is assumed to be strictly positive and it has no common factor with $A(q)$ and $B(q)$. The polynomial $A_2(q)$ is the greatest common divisor of $A(q)$ and $B(q)$ and A_2^- of deg m which is the factor of A_2 has all its zeros outside of the unit circle of complex discrete time space.

$$A(q) = A_1(q)A_2(q)$$

$$B(q) = B_1(q)A_2(q)$$

The cost function for $\lambda>0$ is minimized by the control law :

$$R(q)u(t) = -S(q)y(t) + T(q)y_m(t) \tag{4.70}$$

where the polynomials R and S satisfy the diophantine equation

$$A(q)R(q) + B(q)S(q) = P(q)C(q) \tag{4.71}$$

where

$$\deg R(q) = \deg S(q) = m + n$$

$$A_2 \text{ divides } R(q)$$

$$\deg S^*(q) < n$$

The desired polynomial $P(q)$ is given by

$$P(q) = q^m P_1(q)_2A(q) \tag{4.72}$$

$$P_1(q)P_1(q^{-1}) = \lambda A_1(q)A_1(q^{-1}) + B(q)B(q^{-1}) \tag{4.73}$$

$$T(q) = t_o q^m C(q)$$

$$t_o = P_1(1) / B(1)$$

The application of this method for a specific value of λ is shown below:

Step 1) The coefficients of the polynomials A, B and C are estimated from the eq(4.25).

Step 2) The polynomial $P(q)$ is obtained from the eq(4.73).

Step 3) The diophantine equation (4.71) is solved.

Step 4) The control signal is calculated from the eq(4.70)

This operation is continued for each sampling period.

Another spectral factorization method based on LQG control law has been developed by Grimble [37],[38]. He has considered the system model in the form :

$$y(t) = \frac{B(z^{-1})}{A(z^{-1})}u(t) + \frac{C(z^{-1})}{A_d(z^{-1})}\zeta(t) + \frac{D(z^{-1})}{A_l(z^{-1})}v(t) \qquad (4.74)$$

where $v(t)$ is measurable load disturbance. The measurable disturbance $v(t)$ and the reference signal $r(t)$ are generated by the linear systems :

$$r(t) = \frac{E(z^{-1})}{A_e(z^{-1})}\varepsilon(t) \qquad (4.75)$$

$$v(t) = \frac{F(z^{-1})}{A_f(z^{-1})}\xi(t) \qquad (4.76)$$

The control input is described by

$$u(t) = -\frac{C_{1n}}{C_{1d}}y(t) + \frac{C_{2n}}{C_{2d}}r(t) - \frac{C_{3n}}{C_{3d}}v(t) \tag{4.77}$$

Tracking error is defined by

$$e(t) \triangleq r(t) - y(t) \tag{4.78}$$

It is assumed that the following conditions are satisfied :

i) ζ, ξ and ε are mutually uncorrelated stochastic squences

ii) The process transfer function $\frac{B}{A}$ does not include unstable hidden modes. The reference and disturbance subsystems ($\frac{C}{A_d}$, $\frac{D}{A_l}$, $\frac{E}{A_e}$, $\frac{F}{A_f}$) are asymptotically stable.

iii) The reference and disturbance models can without loss of generality be assumed inverse stable.

iiii) All system polynomials without B and D are monic.

The cost function is to be minimized by the following equation :

$$J = \frac{1}{2\pi j} \oint_{|z|=1} (Q_c(z^{-1})\Phi_{ee}(z^{-1}) + R_c(z^{-1})\Phi_{uu}(z^{-1}))\frac{dz}{z} \tag{4.79}$$

where $\Phi_{ee}(z^{-1})$ and $\Phi_{uu}(z^{-1})$ represent the spectral densities of the signals $e(t)$ and $u(t)$, respectively. Q_c and R_c are the weighting elements that are strictly positive on the unit circle.

$$Q_c \triangleq \frac{Q_n}{A_q^* A_q} , \qquad R_c \triangleq \frac{R_n}{A_r^* A_r} \tag{4.80}$$

where A_{qc} and A_{rc} are strictly monic polynomials.

The coefficients polynomials of $y(t)$, $r(t)$ and $v(t)$ in (4.77) can be defined as follows

$$C_{1n} = G\ A_r , \qquad C_{1d} = H\ A_q \tag{4.81}$$

$$C_{2n} = M\ A_r , \qquad C_{2d} = A_q HE \tag{4.82}$$

$$C_{3n} = (\ XD_f - FDG\ )\ A_r, \qquad C_{3d} = A_q HFA_1 \tag{4.83}$$

The polynomials G, H are the minimal degree solutions with respect to Z of the coupled diophantine equations

$$D_c^* Gz^{-g} + ZAA_q A_d = B^* A_r^* Q_n D_f z^{-g} \tag{4.85}$$

$$D_c^* Hz^{-g} - ZBA_r A_d = A^* A_q^* R_n D_f z^{-g} \tag{4.86}$$

where $g \triangleq \max\ (n_D,\ n_B + n_{A_r} + k,\ n_A + n_{A_q}\ )$.

The minimal degree solution of the polynomial M with respect to N can be found from the following diophantine equation :

$$D_c^* z^{-g1} M + NA_q A_e = B^* A_r^* Q_n Ez^{-g1} \tag{4.87}$$

where $g1 \triangleq \max\,(n_{D_c},\ n_B + n_{A_r} + k\ )$.

The minimal degree solution of the polynomial X with respect to Y can be found via the diophantine equation :

$$D_c^{*}z^{-g2}X + YA_qA_fA_l = B^{*}A_r^{*}Q_nFDz^{-g2} \tag{4.88}$$

where $g2 \triangleq \max\,(n_{D_c},\ n_B + n_{A_r} + k\ )$

The polynomials D_c and D_f which appear in the above diophantine equations are defined by

$$D_c^{*}D_c = B^{*}A_r^{*}Q_nA_rB + A^{*}A_q^{*}R_nA_q^{*}A \tag{4.89}$$

$$D_f^{*}D_f = AA^{*}CC^{*} \tag{4.90}$$

The Z is eliminated between equations (4.85) and (4.86), and the new diophantine equation appears as:

$$AA_qH + A_rBG = D_fD_c \tag{4.91}$$

The eq(4.91) is the characteristic equation of the closed-loop system. The stability of the closed-loop system is guaranteed because of the definitions of D_c and D_f. All controllers, which are

C_1, C_2 and C_3, can adjust their zeros due to the A_r weighting term and their poles due to the A_q weighting term. All these theories and properties are to be found in the references [37] and [38]. The estimation equation is given by

$$\hat{y}(t) = \hat{\Theta}^T(t-1)\Phi(t) \tag{4.92}$$

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{4.93}$$

where

$$\hat{\Theta}^T(t) = [a_1 \dots a_{n_A} ; b_0 \dots b_{n_B} ; d_{f_1} \dots d_{f_{nDf}} ; a_{d_1} \dots a_{d_{nAd}} ; \\ d_0 \dots d_{n_D} ; a_{l_1} \dots a_{l_{nAl}}] \tag{4.94}$$

$$\Phi^T(t) = [-r(t-1) \dots -r(t-n_A) ; u(t-k) \dots u(t-k-n_B) ; \\ \bar{\varepsilon}(t-1) \dots \bar{\varepsilon}(t-n_{Df}) ; -s(t-1) \dots -s(t-n_{Ad}) ; \\ v(t-k_D) \dots v(t-k_D-n_D) ; -q(t-1) \dots -q(t-n_{Al})] \tag{4.95}$$

where

$$q(t) = \frac{\hat{D}(z^{-1})}{\hat{A}_l(z^{-1})} v(t), \tag{4.96}$$

$$r(t) = y(t) - q(t) \tag{4.97}$$

$$s(t) = \hat{A}(z^{-1})r(t) - \hat{B}(z^{-1})u(t) \tag{4.98}$$

$$\bar{\varepsilon}(t) = \frac{\hat{A}_d(z^{-1})}{\hat{D}_f(z^{-1})} s(t) \tag{4.99}$$

The application of this method to the adaptive control for a chosen cost function weight can be given by the following steps,

Step 1) The coefficients of the polynomials A, B, A_d, A_1, D and D_f are identified with a suitable recursive estimation method.

Step 2) The spectral factor D_c is calculated.

Step 3) The controller polynomials the eqs(4.81) to (4.83) are calculated.

Step 4) The control signal is calculated from the eq(4.77)

The steps 1 to 4 are repeated each sampling period.

4.5. CONCLUSION

In this chapter, the MRAS and the STR systems have been discussed. Because we want to review the adaptive control of non-minimum phase and unstable systems. MRAS has only been mentioned briefly. Many MRAS algorithm involve pole-zero cancellation, therefore the non-minimum phase plant causes instability. When the stability of the MRAS is considered, one of the assumptions for stability is that the system is minimum phase [39]. The M.I.T. rule is valid for the small difference between the process and the desired process. The Lyapunov redesign method gives strictly positive real closed-loop but it needs some derivatives for high order system. However, the augmented error model avoids the derivation problem.

The self-tuning method has been explained with two categories, explicit and implicit. Both of them are used for non-minimum phase or the unstable plants, but the implicit method may not distinguish whether the system is non-minimum phase or not. When the system is known to be non-minimum phase, it can be controlled by the implicit method as well as the explicit method.

The LQG method can give an optimal controller. The LQG method is stable for every condition of the process. However, the self-tuning controller cost function value is not very different from an optimal controller, which has been shown for some systems in reference [5]. The LQG method is an optimal method but its application is more difficult than STR. Hence, the STR can be used for this application.

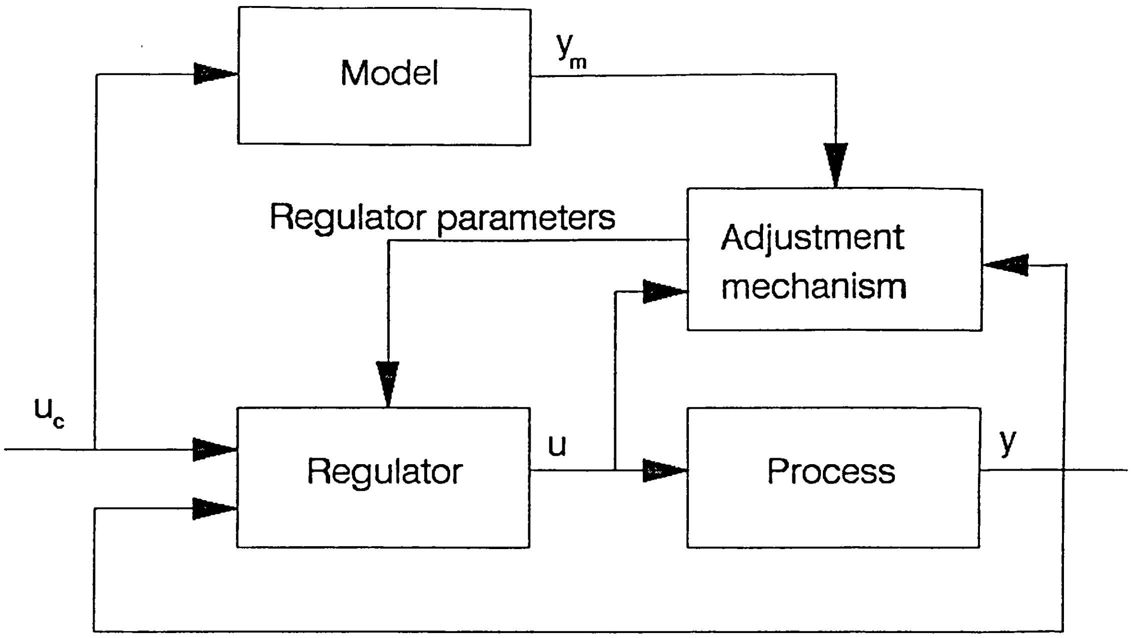


Fig.4.1. The original Model Reference Adaptive System

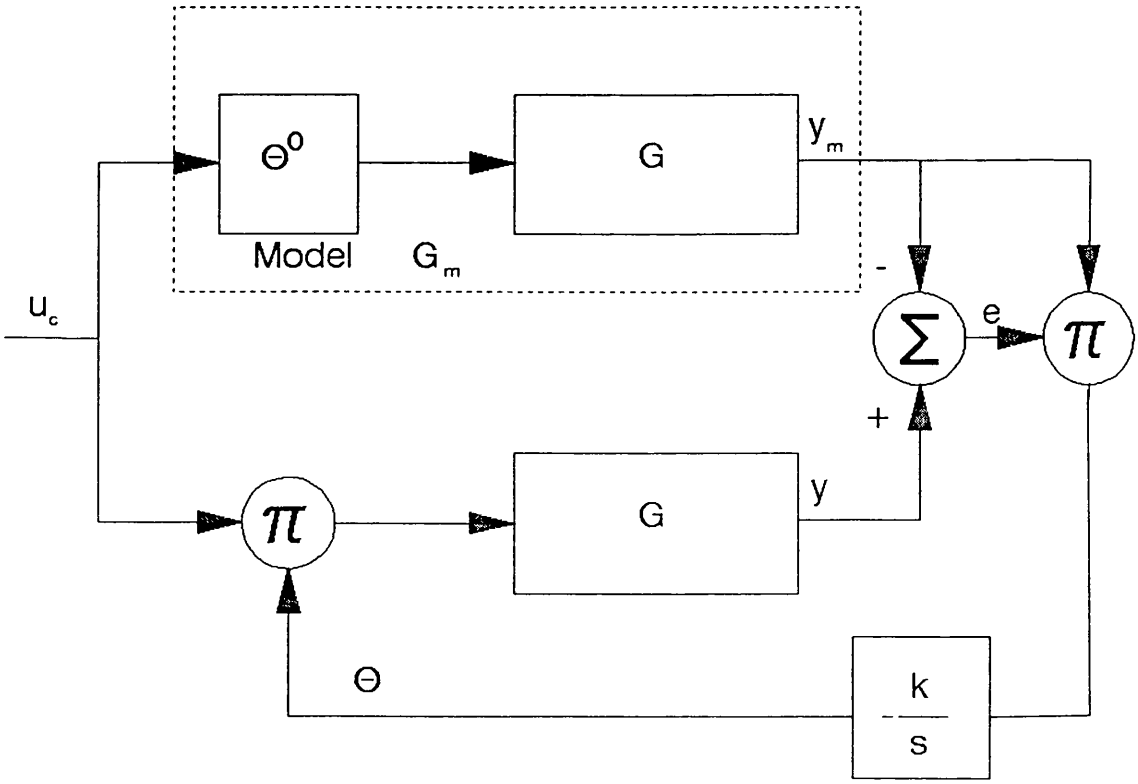


Fig.4.2. The M.I.T. Design Adaptive Controller

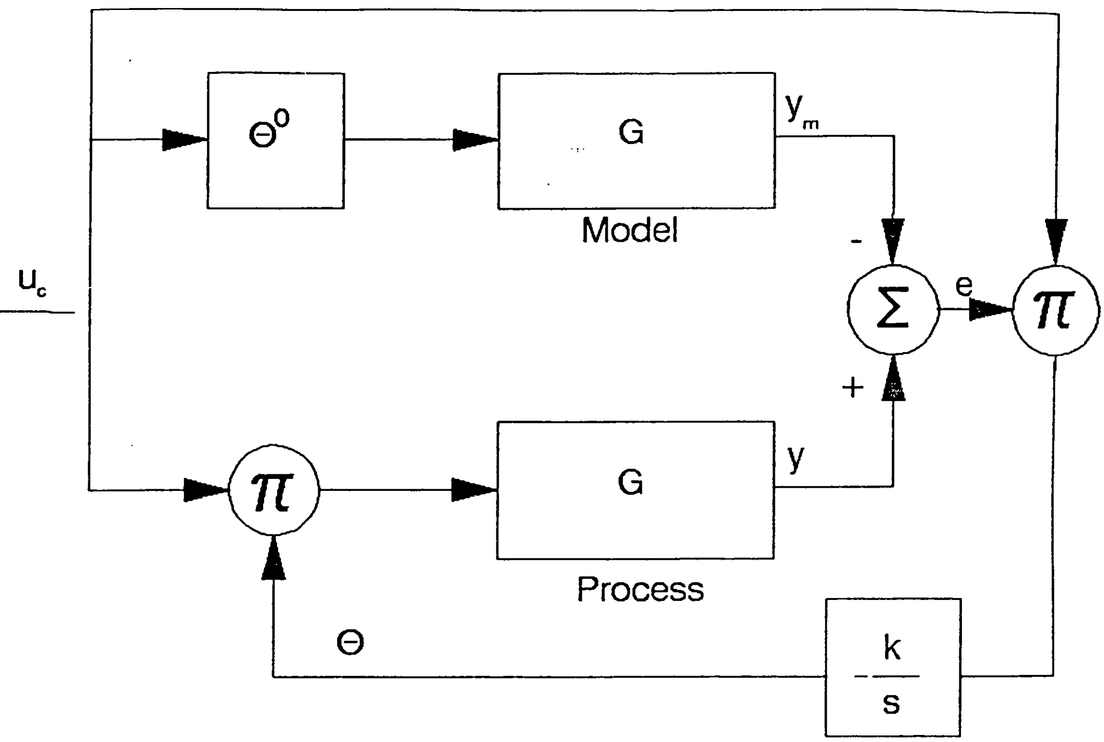


Fig.4.3. The Lyapunov Redesign MRAS

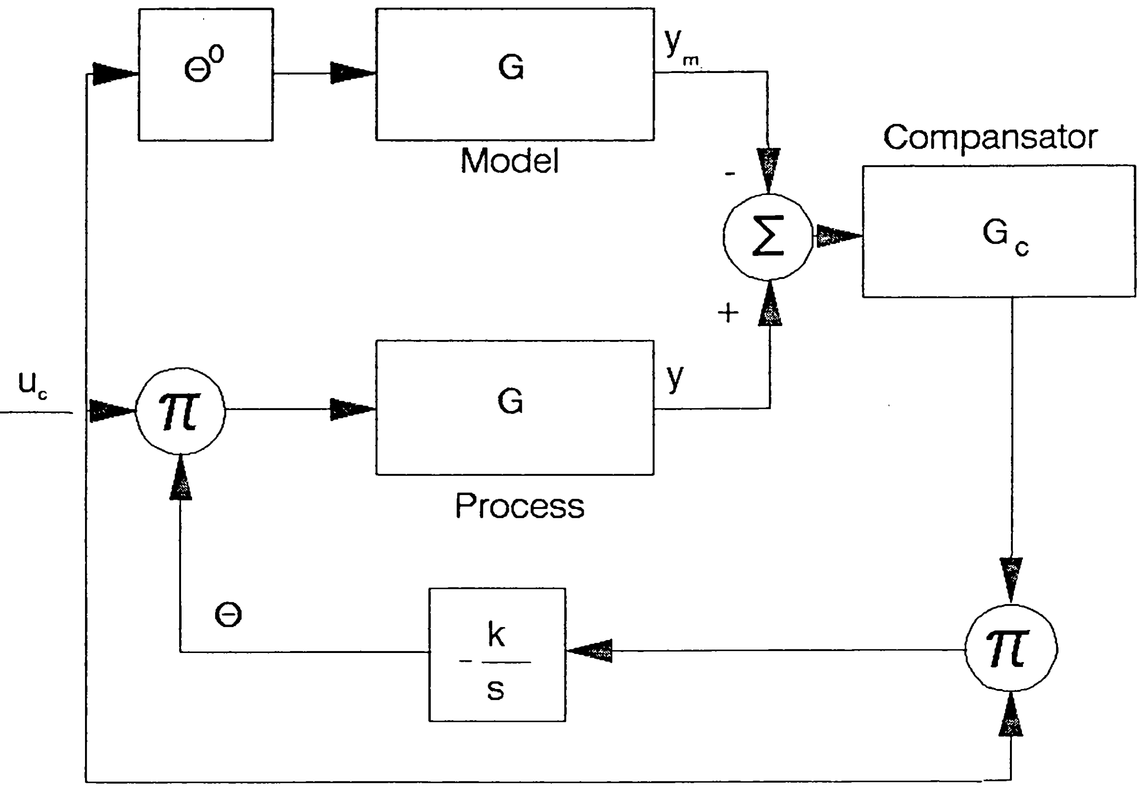


Fig.4.4. The strickly positive real MRAS

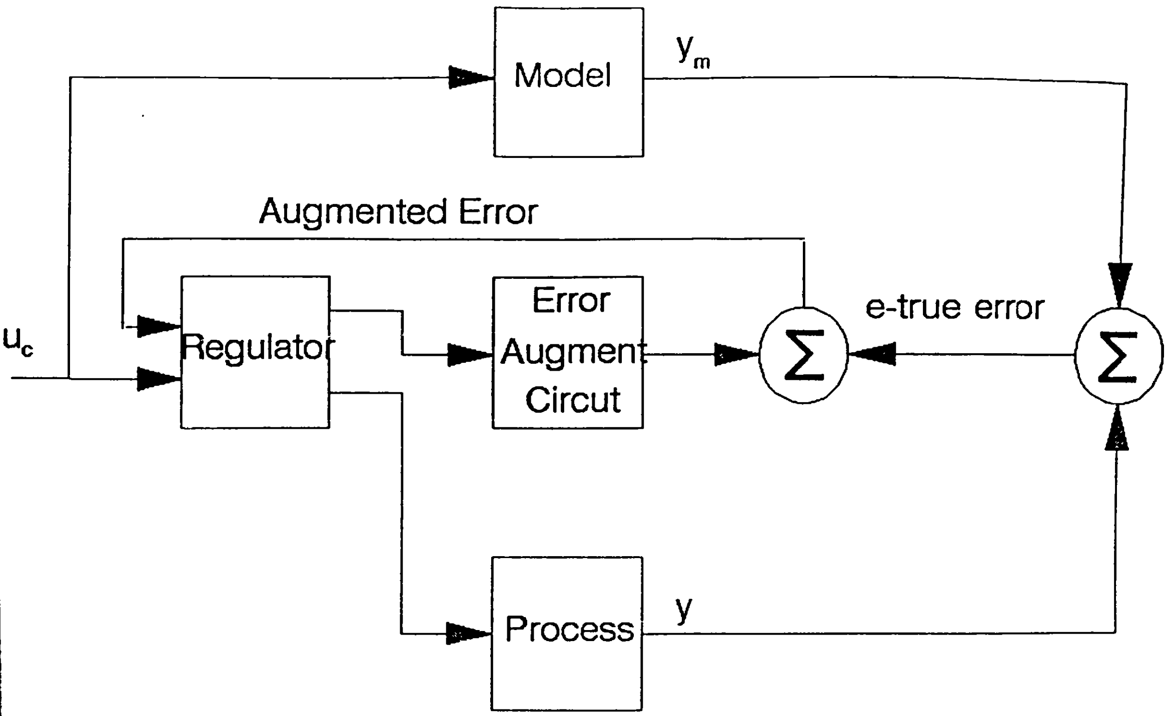


Fig.4.5. The MRAS with an Augment Error Signal

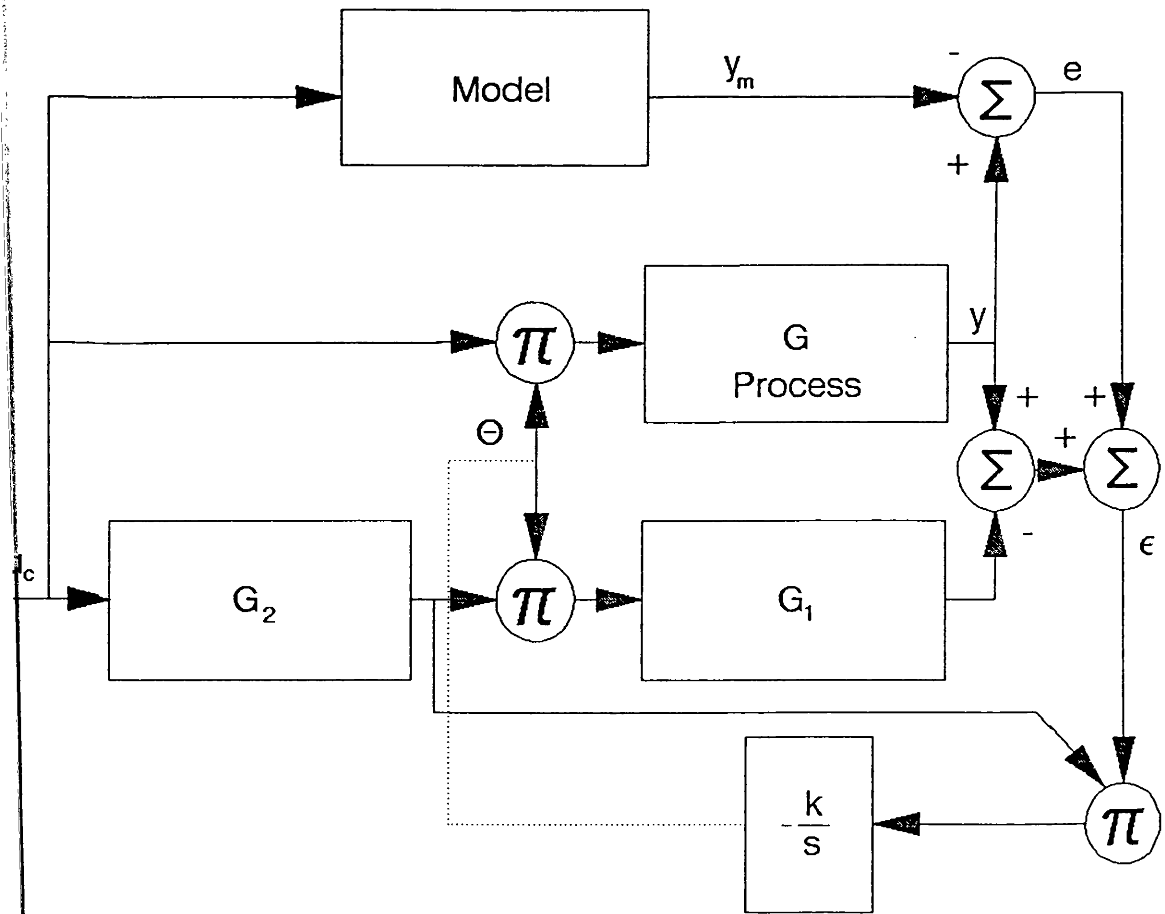


Fig.4.6. An application of the augment signal on MRAS

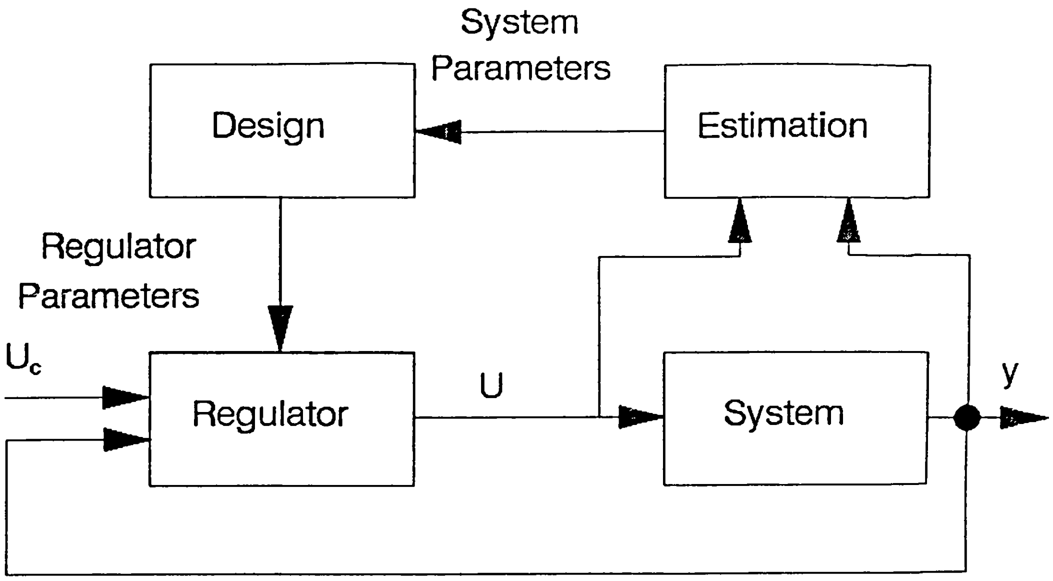


Fig.4.7. Explicit Self-tuning Regulator

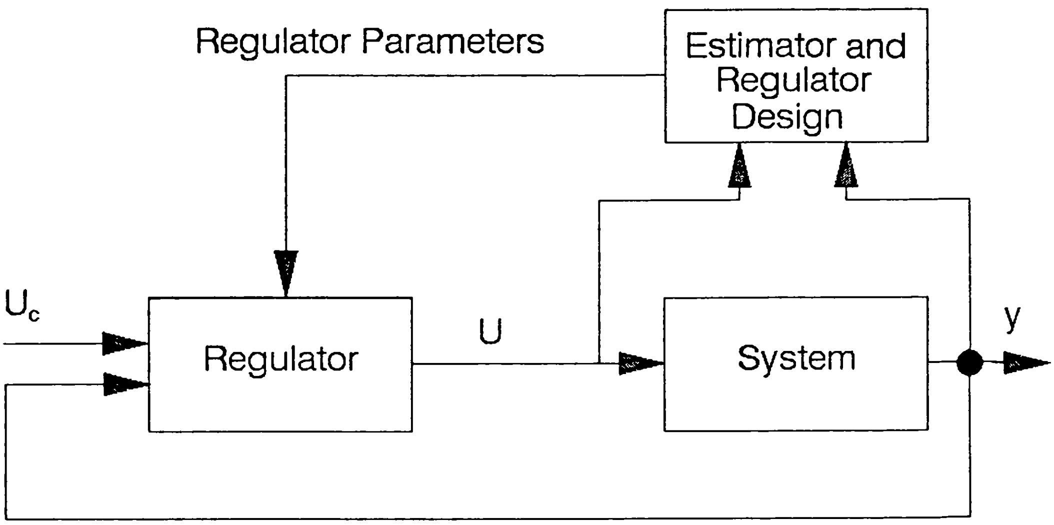


Fig.4.8. Implicit Self-tuning Regulator

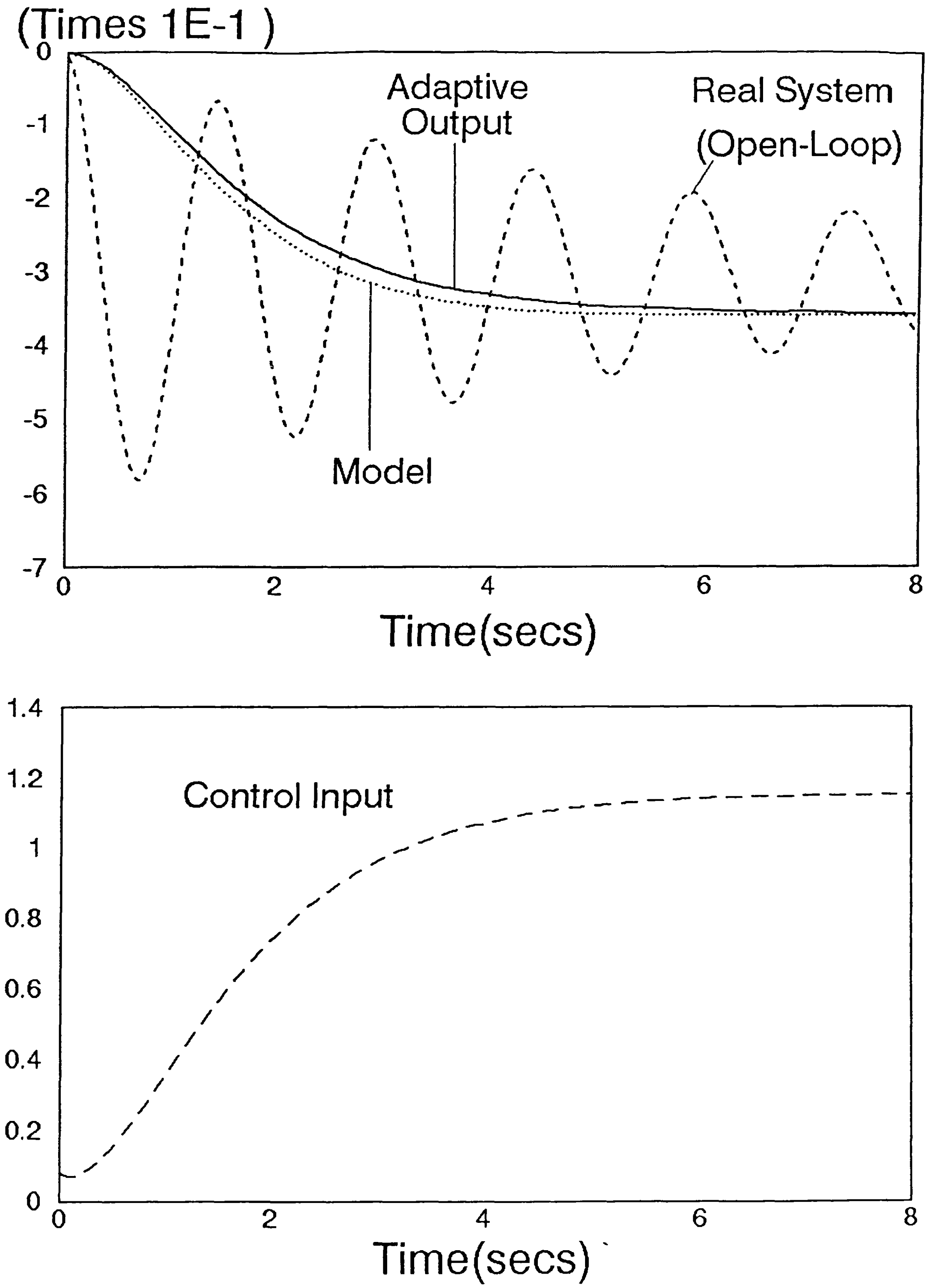


Fig.4.9. The outputs before adaptive control, adaptive control and model and control input in example 4.1.

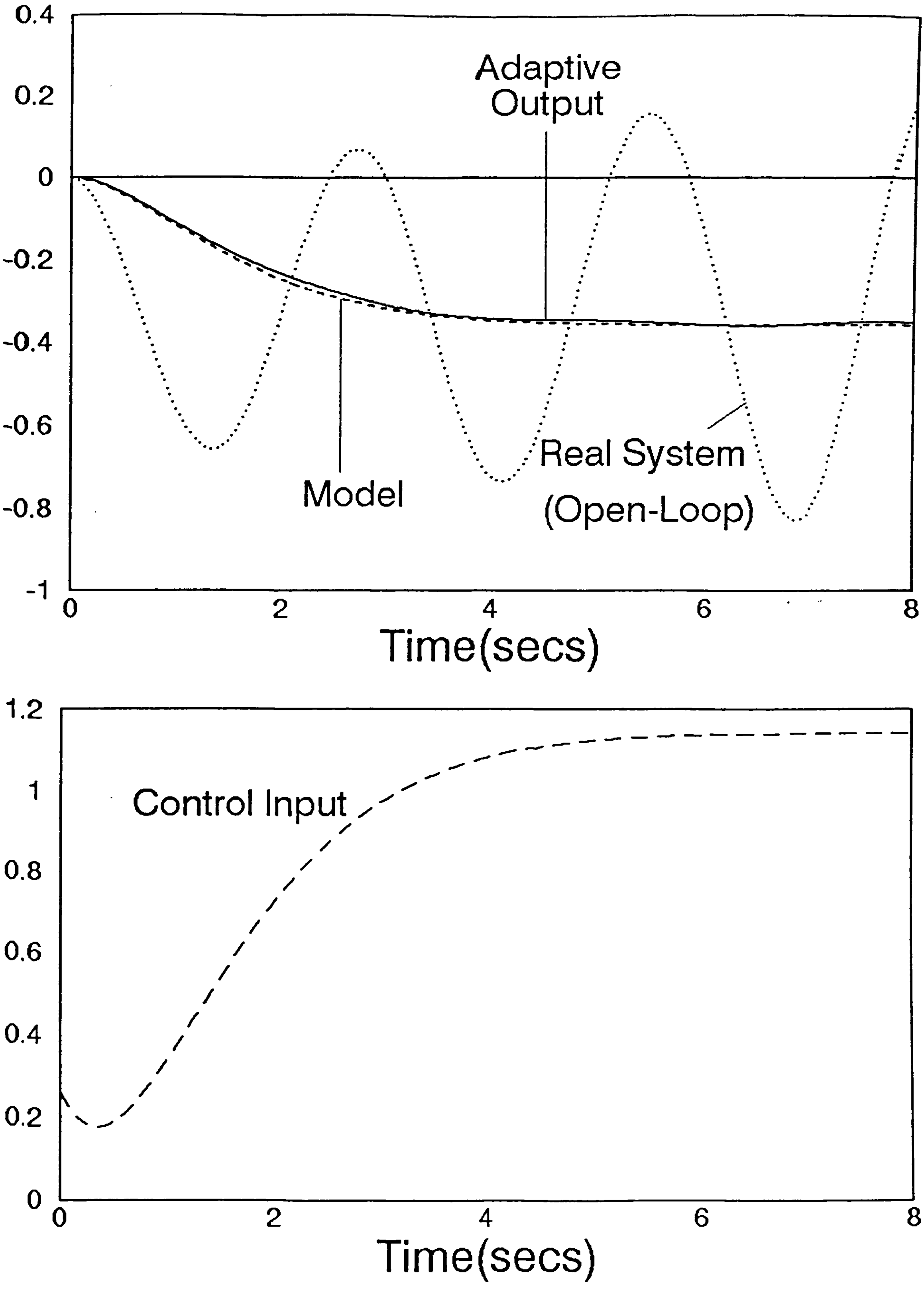


Fig.4.10. The outputs for the unstable system in example 4.1.

CHAPTER 5. IMPLEMENTATION OF IDENTIFICATION

5.1. INTRODUCTION

A solution of an aircraft identification problem is presented in this chapter. Firstly, the essential equations of the aircraft motion are introduced and simplified. Subsequently, this motion is simulated with a suitable model as a real aircraft. The model system has been identified in real time and the block diagram of the realization is given by Fig.5.1.

The longitudinal equations of the aircraft motion and the lateral equations of the aircraft motion were defined in chapter 3. The longitudinal equations of motion are again reviewed to include atmospheric turbulence.

The results of the parameter estimation have calculated for the different parameter structure and for different environmental conditions including noises and gusts etc.

5.2. FLIGHT MODELLING

The aircraft is assumed to be in a trimmed position with a constant speed . Therefore the angular displacements of elevator, ailerons and rudder are zero. It has been shown in chapter 3 that the longitudinal motion only depends on the elevator. On the other hand, the lateral motion is independent from the elevator movements. Thus the flight modelling during landing can be represented by the equation of the longitudinal motion. Eqs(3.45) to (3.47) are rewritten as:

$$(\hat{D} + x_u)\hat{u} + (x_{\dot{w}}\hat{D} + x_w)\hat{w} + x_q q + \hat{g}_1\theta + x_\eta \eta' = 0 \tag{5.1}$$

$$z_u\hat{u} + [(1 + z_{\dot{w}})\hat{D} + z_w]\hat{w} + (z_q - 1)\hat{q} + \hat{g}_2\theta + z_\eta \eta' = 0 \tag{5.2}$$

$$m_u\hat{u} + (m_{\dot{w}}\hat{D} + m_w)\hat{w} + (\hat{D} + m_q)\hat{q} + m_\eta \eta' = 0 \tag{5.3}$$

$\hat{u}$ is defined by $\frac{u}{V_e}$ where u represents the difference between the disturbed flight velocity and the steady flight velocity along Ox and V_e is the aircraft speed in steady flight. According the initial condition assumption $\hat{u}$ can be omitted on the equations (5.1) to (5.2) [26]. The term $\hat{g}_2\theta$ can also be neglected in comparison other term. In this case, Eqs(5.2) and (5.3) are rewritten as:

$$[(1 + z_{\dot{w}})\hat{D} + z_w]\hat{w} + (z_q - 1)\hat{q} + z_\eta \eta' = 0 \tag{5.4}$$

$$(m_{\dot{w}}\hat{D} + m_w)\hat{w} + (\hat{D} + m_q)\hat{q} + m_\eta \eta' = 0 \tag{5.5}$$

where $\hat{D}$ is the differential operator ($\hat{D} = \frac{\mathrm{d}}{\mathrm{dt}}$), $\hat{w}$ is the attack angle, which is equal to $\frac{w}{V_e}$, $\hat{q}$ is the aircraft angular velocity in pitch, η' is the increment in elevator angle from trimmed position. Other coefficients are aerodynamic derivatives of the aircraft that can change with flight condition. Therefore the aircraft dynamics change because of the change in coefficients. During the landing time, the parameters of the dynamic can be assumed to be constant because the landing time is small in comparison with the parameter varying time. Although, we reviewed to identification for time-varying parameters. The considered aircraft model is given as Eqs(5.4) and (5.5):

$$(ar_{11}\frac{\mathrm{d}}{\mathrm{dt}} + ar_{12})\alpha - 0.9892\ ar_{11}q = br_1\eta' \qquad (5.6)$$

$$(ar_{21}\frac{\mathrm{d}}{\mathrm{dt}} + ar_{22})\alpha + (\ ar_{23}\frac{\mathrm{d}}{\mathrm{dt}} + ar_{24})q = br_2\eta' \qquad (5.7)$$

where α is equal to w. This system has been identified by direct continuous method where the quasilinearization method has been used as discussed in chapter 2. For this purpose, Eqs(5.6) and (5.7) were written the state space form as:

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{bmatrix} \alpha \\ q \end{bmatrix} = \begin{bmatrix} a_{11} & 0.9892 \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\eta' \qquad (5.8)$$

where η' includes a first order lag which was given 0.1 sec by the manufacturer. This model was simulated by using the fourth order

Runge-Kutta integration method on a personal computer. The step time is equal 40 msec which is determined according to the telemetry of the aircraft. The desired performance coefficients values which is given as Appendix 1.14

$$\frac{d}{dt}\begin{bmatrix} \alpha \\ q \end{bmatrix} = \begin{bmatrix} -0.0142 & 0.9892 \\ -1.244 & -1.924 \end{bmatrix}\begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} 1.17e\text{-}03 \\ -.434 \end{bmatrix} \eta' \qquad (5.9)$$

Experiences on the simulation of Eq(5.8) were done for different kind of parameters and noises which will be given next sections.

5.3. THE DESIGN OF THE IDENTIFICATION ROUTINE

The quasilinearization method was used to identify the parameters. This method was given in chapter 2 but it is briefly given here again. The identified system has been considered as :

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t) \qquad (5.10)$$

$$\dot{x}(t) = f(x, u, a, b, t)$$

where A and B are the parameter matrices, whose element can be time varying, x is state variable matrix and u is control input matrix. The state observable values is defined by

$$s_i(t) = p_i(t) + \sum_{k=1}^{n} c_k h_{ki} \qquad (5.11)$$

where s is the vector of the observable values of the process state variable, p is the vector of the state variables of the estimation model, h_k is the vector of the sum of the homogeneous variables and n is the number of the unknown parameters. c_k will be used to modify initial estimation of unknown parameters. Firstly, an estimation model and homogeneous models are described respectively. The estimation model was chosen to have exactly the same structure as the flight model but all coefficients were assumed to be equal to 1 as :

$$\frac{d}{dt} p(t) = \hat{A}p(t) + \hat{B}\eta' \qquad (5.12)$$

$$\frac{d}{dt}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}}_{\hat{A}} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\hat{B}} \eta' \qquad (5.13)$$

The homogeneous systems are defined for each unknown parameter which are given as :

$$\frac{d}{dt} h_k(t) = \hat{A}h_k(t) + \frac{\partial f}{\partial a_k} \qquad (5.14)$$

When $\dfrac{\partial f}{\partial a_k}$ are calculated, the homogeneous systems are found as:

$$\frac{d}{dt}\begin{bmatrix} h_{11} \\ h_{12} \end{bmatrix} = \begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} h_{11} \\ h_{12} \end{bmatrix} + \begin{bmatrix} s_1 \\ 0 \end{bmatrix} \tag{5.15}$$

$$\frac{d}{dt}\begin{bmatrix} h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} h_{21} \\ h_{22} \end{bmatrix} + \begin{bmatrix} 0 \\ s_1 \end{bmatrix} \tag{5.16}$$

$$\frac{d}{dt}\begin{bmatrix} h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} h_{31} \\ h_{32} \end{bmatrix} + \begin{bmatrix} 0 \\ s_2 \end{bmatrix} \tag{5.17}$$

$$\frac{d}{dt}\begin{bmatrix} h_{41} \\ h_{42} \end{bmatrix} = \begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} h_{41} \\ h_{42} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \eta' \tag{5.18}$$

$$\frac{d}{dt}\begin{bmatrix} h_{51} \\ h_{52} \end{bmatrix} = \begin{bmatrix} 1 & 0.9892 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} h_{51} \\ h_{52} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \eta' \tag{5.19}$$

The control input vectors of $h_4(t)$ and $h_5(t)$ were found according the derivation of $\frac{\partial f}{\partial b_i}$. Eq(5.11) must be written in matrix form to solve the elements of the c matrix. In this problem, the number of variables is 2, the number of the unknown parameters (c_k) is 5, therefore we need to add enough observations In this case, Eq(5.11) can be rewritten in the matrix form as :

$$
\begin{bmatrix} s_1(t) \\ s_2(t) \\ s_1(t+1) \\ s_2(t+1) \\ s_2(t+2) \end{bmatrix} = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_1(t+1) \\ p_2(t+1) \\ p_2(t+2) \end{bmatrix} +
$$

$$
+ \begin{bmatrix} h_{11}(t) & h_{21}(t) & h_{31}(t) & h_{41}(t) & h_{51}(t) \\ h_{21}(t) & h_{22}(t) & h_{32}(t) & h_{42}(t) & h_{52}(t) \\ h_{11}(t+1) & h_{21}(t+1) & h_{31}(t+1) & h_{41}(t+1) & h_{51}(t+1) \\ h_{21}(t+1) & h_{22}(t+1) & h_{32}(t+1) & h_{42}(t+1) & h_{52}(t+1) \\ h_{21}(t+2) & h_{22}(t+2) & h_{32}(t+2) & h_{42}(t+2) & h_{52}(t+2) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} \qquad (5.20)
$$

The c matrix can be solved from Eq(5.20). The c matrix is corresponded with the unknown parameter as :

$$
\hat{a}_{11}(k+1) = c_1 + \hat{a}_{11}(k), \quad \hat{a}_{21}(k+1) = c_2 + \hat{a}_{21}(k),
$$

$$
\hat{a}_{22}(k+1) = c_3 + \hat{a}_{22}(k)
$$

$$
\hat{b}_1(k+1) = c_4 + \hat{b}_1(k) \ , \quad \hat{b}_2(k+1) = c_5 + \hat{b}_2(k) \qquad (5.21)
$$

All of the identification operations were done with TMS320C30 Digital Signal Processor. The lag on the elevator response was also considered in the identification operation. The essential software and hardware will be discussed in the next sections.

5.4. DESIGN OF THE INTERFACE

The identification board clock frequency was different from the clock frequency of the personal computer which was used for the flight modelling. Therefore a synchro-communication between PC and the identification board was not possible. Hence an interface was inserted between them in order to realize the asynchro-communication. This interface provided parallel communication. Its detail will also be given in a subsequent chapter.

5.5. THE IDENTIFICATION RESULT

5.5.1. Time Invariant System Without Noise

The noise-free time invariant system can be represented as

$$\frac{d}{dt} x(t) = Ax(t) + Bu(t) \tag{5.22}$$

$$s(t) = Cx(t)$$

where C is unit matrix, because the identification method uses the state variables directly. It has been seen from Eq(5.20) that the identification operation needs three step results after initial condition. In addition one step is necessary because of the dynamics of the elevator response. Therefore first result is obtained at the fourth step after the initial condition. The results of the identification are given in Fig.5.2. and Fig.5.3. The Fig.5.2. and Fig.5.3. illustrate the parameters which were identified between the initial condition zero and the last observable values. The Fig.5.5. and Fig.5.6. show the parameters which were identified between the last observable values and previous three steps values. It can be shown that the identification is possible to each boundaries before the steady state of the outputs. The unit step function was applied to the process (flight modelling) and its outputs, which were used in the parameter estimation, are shown in Fig.5.4.

5.5.2. The Time-variable Parameters

This estimation method can be used for the time-invariant system. When the observer time is chosen as short as possible then the parameters can be assumed to be constant during identification interval and the identification can be possible. Three observation steps are good enough to calculate the unknown parameters in our study. This time is acceptable and the coefficients are constant. The parameters can be found as Eq(2.139):

$$\hat{a}_i(N+1) = \hat{a}_i(N) + c_i(N+1) \tag{5.23}$$

$$c(N) = [\ h(N)\ ]^{-1}\ [\ s(N) - p(N)\ ] \tag{5.24}$$

where h, s and p matrices include $(N+1)$th and $(N+2)$th steps' results. Eq(2.138) was not used because the system was assumed to be free of noise. The results are shown in Figs.5.7. to 5.9.. The identified parameters of $\hat{a}_{22}$, $\hat{b}_1$ and $\hat{b}_2$ are slightly fluctuated, because their rate of change is not very small according to step time.

5.5.3. The Identification of Noisy System

The system representation Eq(5.22) is rewritten for the measurement with noise as;

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = Ax(t) + Bu(t) \tag{5.25}$$

$$s(t) = Cx(t) + \xi(t)$$

Where $\xi(t)$ is zero-mean random function, representing the noise. Eq(2.140) can be used for identification with noise. It can be rewritten as ;

$$[\, c \,]_{k+1} = \left(\sum_{i=1}^{N} [\, h(t_i) \,]^T [\, h(t_i) \,] \right)^{-1} * \sum_{i=1}^{N} [\, h(t_i) \,]^T \{ [\, s(t_i) \,] - [\, x(t_i)]\} \tag{5.26}$$

It will be shown that the requirements of step numbers increase due to the amplitude of noise. The identifications were performance for different amplitudes of the random function. The identification results are given by Fig.5.12 to Fig.5.40. For N=30, the identification board was used to collect the data during the first 30 steps. Therefore the first estimation results are available after 30 samples time. The identification board can calculate the parameters with two iterations in one sample time. Hence, each step identification can be done recursively after 30 steps. This must be considered when choosing the initial estimation, because the particular system outputs may cause the processor to overflow in 30 samples time. However the initial estimation values were taken to be the same as the first estimation value when the control input was changed completely. The system outputs, from which we tried to identify the parameters in this project, are not measured with a very good resolution. The maximum measurement error is around 0.007 which can be represented as a random signal amplitude. When N=30, the identification is possible with a small error which is less than 20 percent. When the noise amplitude is bigger than 0.01, estimation error increases more than 50 percent. Therefore the number of N must be increased. The identification is possible until the noise amplitude is 0.1 with N=60. The noise amplitude 0.1 means that the noise is bigger than 30 percent of the signal, which can be seen in Fig.5.38..

5.6. The Random Turbulence Effect On The Identification

In chapter 3 and the previous sections of this chapter, the atmosphere was assumed to be uniform. It is necessary to recognize the existence of atmospheric wind or gust and its effect. We discuss only the gust effect on the pitching. The magnitude of the vertical gust can be represented with w_g. The effect of the gust is to change the angle of attack of the aircraft; and since an updraft causes a positive change in the angle of attack, $\alpha_g = -w_g/U$, where w_g is considered negative for an updraft [27]. The longitudinal equations can be obtained by adding $z_w\alpha_g$ and $m_w\alpha_g$ to the right-hand sides of Eqs(5.4) and (5.5). Then they become [2]

$$[(1 + z_{\dot{w}})\hat{D} + z_w]\hat{w} + (z_q - 1)\hat{q} + z_\eta\eta' = -z_w\alpha_g \tag{5.27}$$

$$(m_{\dot{w}}\hat{D} + m_w)\hat{w} + (\hat{D} + m_q)\hat{q} + m_\eta\eta' = -m_w\alpha_g \tag{5.28}$$

α_g has been defined according to the vertical gust. A gust or an atmospheric disturbance is represented with a scalar reference intensity $\bar{\sigma}$. Turbulence intensity is often described qualitatively as light, severe, etc.; such term are here in related to specific values of the reference intensity $\bar{\sigma}$ according to Table 1 [40]:

In reference [40], the reference intensity ($\bar{\sigma}$) has been shown to vary smoothly with altitude above 75 m (250 ft) and remains constant below 75 m. The rms intensities of the *u*, *v* and *w* components of turbulence are related to $\bar{\sigma}$. Von Karman and Dryden have described differently as:

Von Karman:

$$\frac{\sigma_{u_g}}{(L_u)^{1/3}} = \frac{\sigma_{v_g}}{(L_v)^{1/3}} = \frac{\sigma_{w_g}}{(L_w)^{1/3}} = \frac{\bar{\sigma}}{(750)^{1/3}} \qquad (5.29)$$

Dryden:

$$\frac{\sigma_{u_g}}{(L_u)^{1/2}} = \frac{\sigma_{v_g}}{(L_v)^{1/2}} = \frac{\sigma_{w_g}}{(L_w)^{1/2}} = \frac{\bar{\sigma}}{(750)^{1/2}} \qquad (5.30)$$

Above 750 m (2500 ft), these simplify to

$$\sigma_{u_g} = \sigma_{v_g} = \sigma_{w_g} = \bar{\sigma} \qquad (5.31)$$

Below 750 m, the changing of the rms intensities of turbulence velocity are given both in reference [40], [41]. They imply that the rms intensities approach the reference intensities by decreasing the measurement time.

The flight model under the gust was realized by the Eqs(5.27) and (5.28). The σ_{α_g} was calculated by using Table 1 and Eq(5.31) and Fig.5.41. The speed U was given 160 Knots in the system. For example, for the moderate gust,

$$\sigma_W \cong \bar{\sigma} = 1.8 \text{ m/s} = 6 \text{ ft/s}$$

$$\sigma_{\alpha_g} = -\frac{\sigma_W}{U} \cong -0.022$$

α_g was represented with intensities and zero mean random function as given Fig.5.42. The identification was performed different gust intensities. Eqs(5.27) and (5.28) are rewritten in a matrix form as Eq(5.8)

$$\frac{d}{dt}\begin{bmatrix} \alpha \\ q \end{bmatrix} = \begin{bmatrix} a_{11} & 0.9892 \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\eta' - \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}\alpha_g \qquad (5.32)$$

where $z_w \cong a_{11}$, $m_w \cong a_{21}$ are assumed. It can be seen from Eq(5.32) that gust affects the control vector parameters. The system with the different atmospheric turbulences, where the intensities have been chosen from Table 1, have been identified and the results are given in Fig.5.42 to Fig.5.67.

5.7. CONCLUSION

Even if the parameters are time-variable, noise free identification can be done accurately. The identification accuracy with noise mixed measurement depends on the noise amplitude. The identification board can identify the system within the transient response time without using a special input. The same algorithm with the noise mixed measurement can be used for the gust effect.

Only the a_{11} value can not be identified from noise mixed observations. This is because its effect on the observation value is smaller than the noise every time. When the atmospheric effect is assumed to be constant, only control vector parameters are affected. If the atmospheric turbulence is modelled by its intensity and a random function, all parameters are changing. However the identified parameter can represent the system very well as seen in Fig.5.67.

Grades of Turbulence and

Reference Intensities

Nominal grade of turbulance	Values of reference intensity, $\bar{\sigma}$	
	m/s	ft/s
Light	0.9	3
Moderate	1.8	6
Severe	3.7	12
Extreme	7.3	24

Table 5.1.

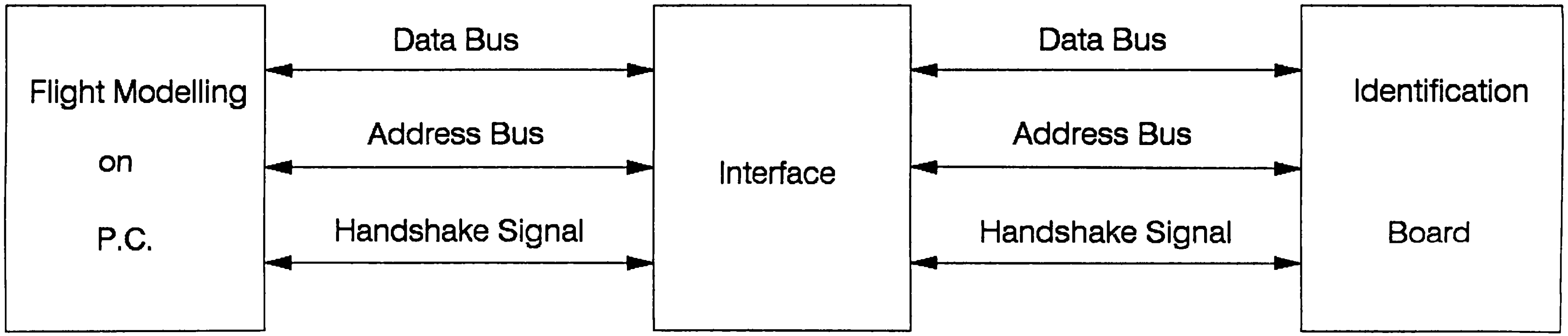


Fig.5.1. Block diagram of the identification on real-time simulation

The following figures 5.2 and 5.3 show how the identification is achieved in a short time for the noise free system. The calculations are done between initial condition and the present time. Figures 5.5. and 5.6 show identification parameter values calculated using the input and output values from the last three steps. Fig. 5.5 and 5.6 indicate that when the derivative of the signals approach zero the errors of the identification increase. These cause small fluctuations in the parameter values but they can be ignored in the transient response time.

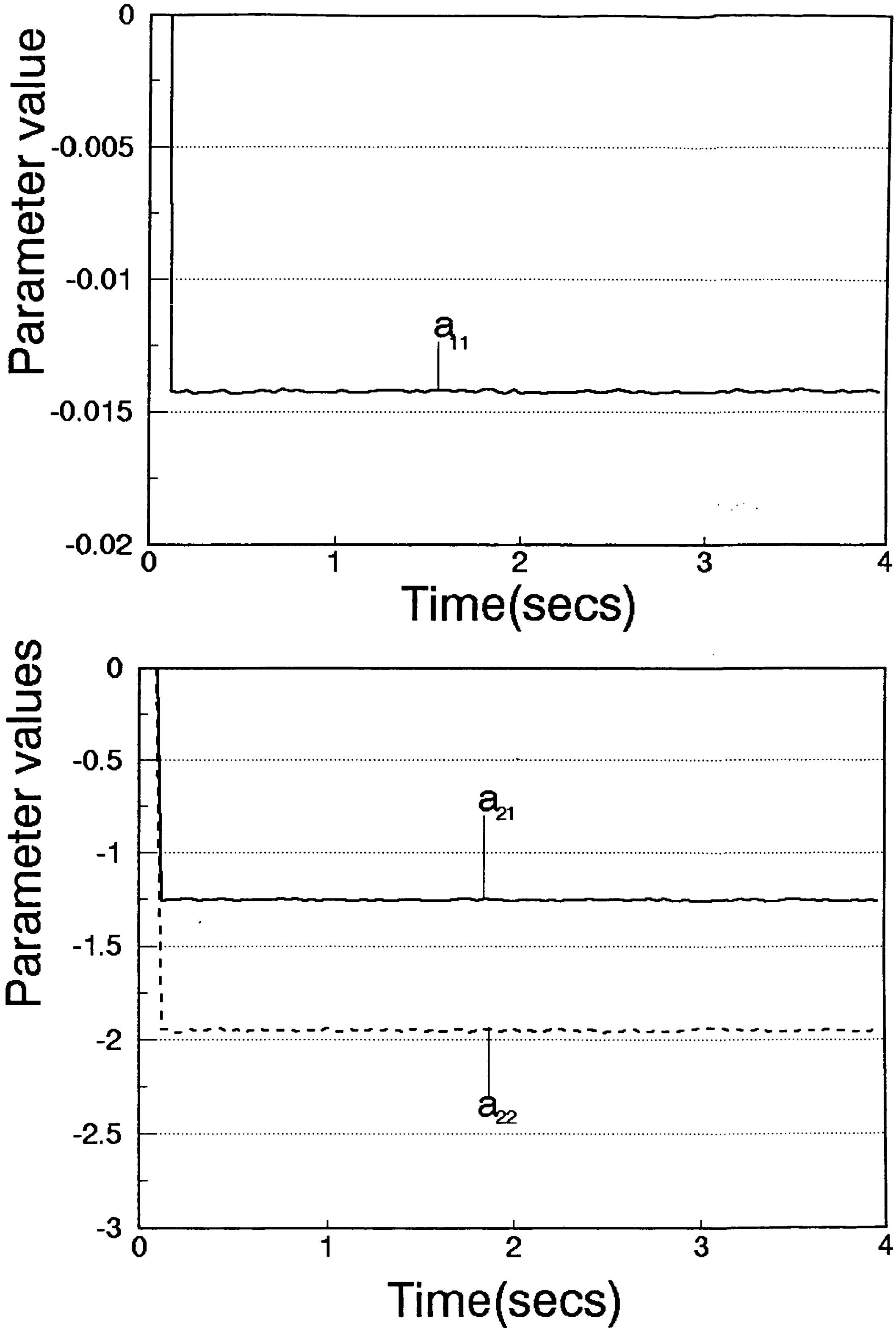


Fig.5.2. The identification results for time invariant system are shown for the noise-free observation. The parameter values can approach to the actual values in a few steps.

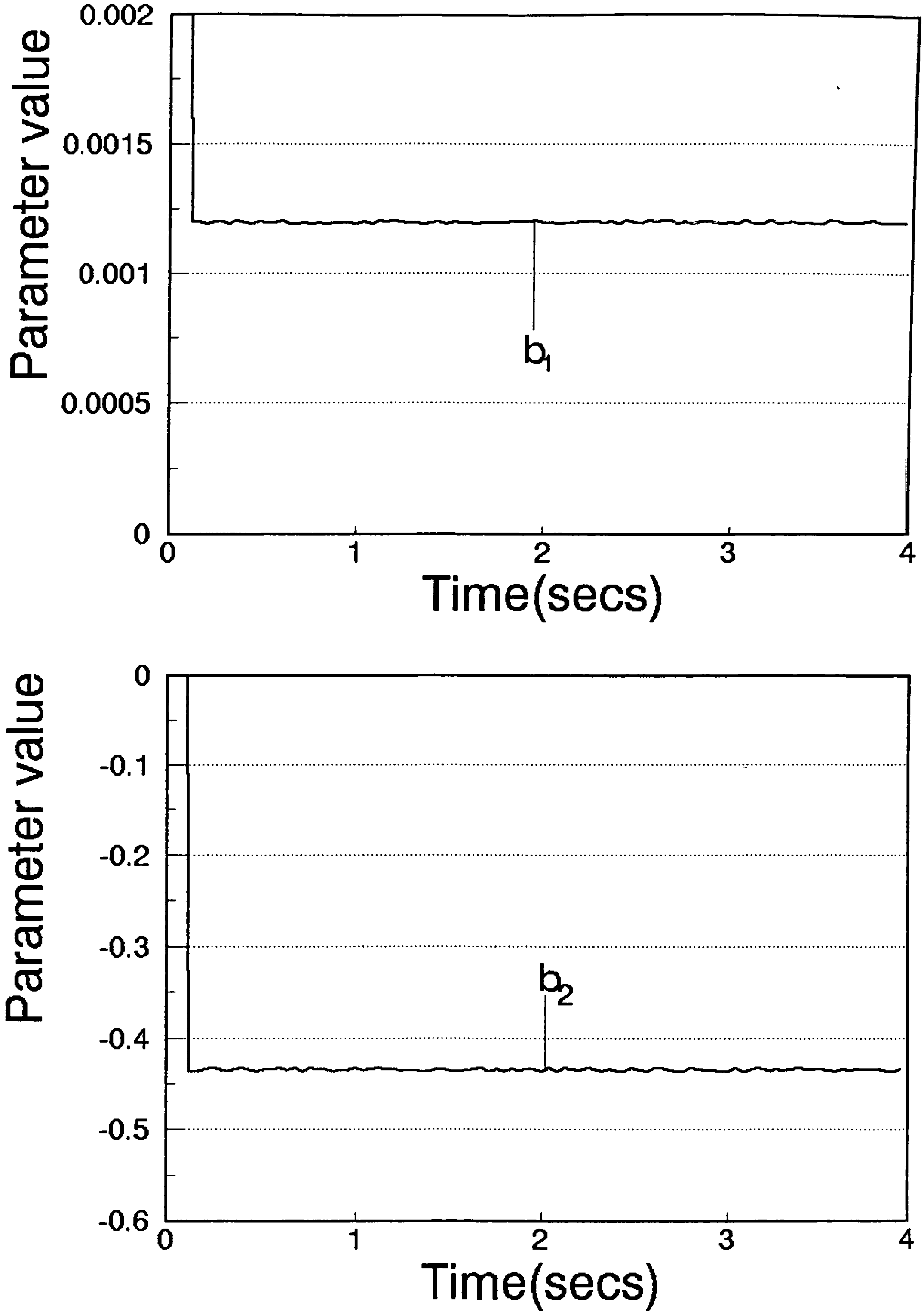


Fig.5.3. The identification result of the control parameter for time invariant system are shown for noise-free observation.

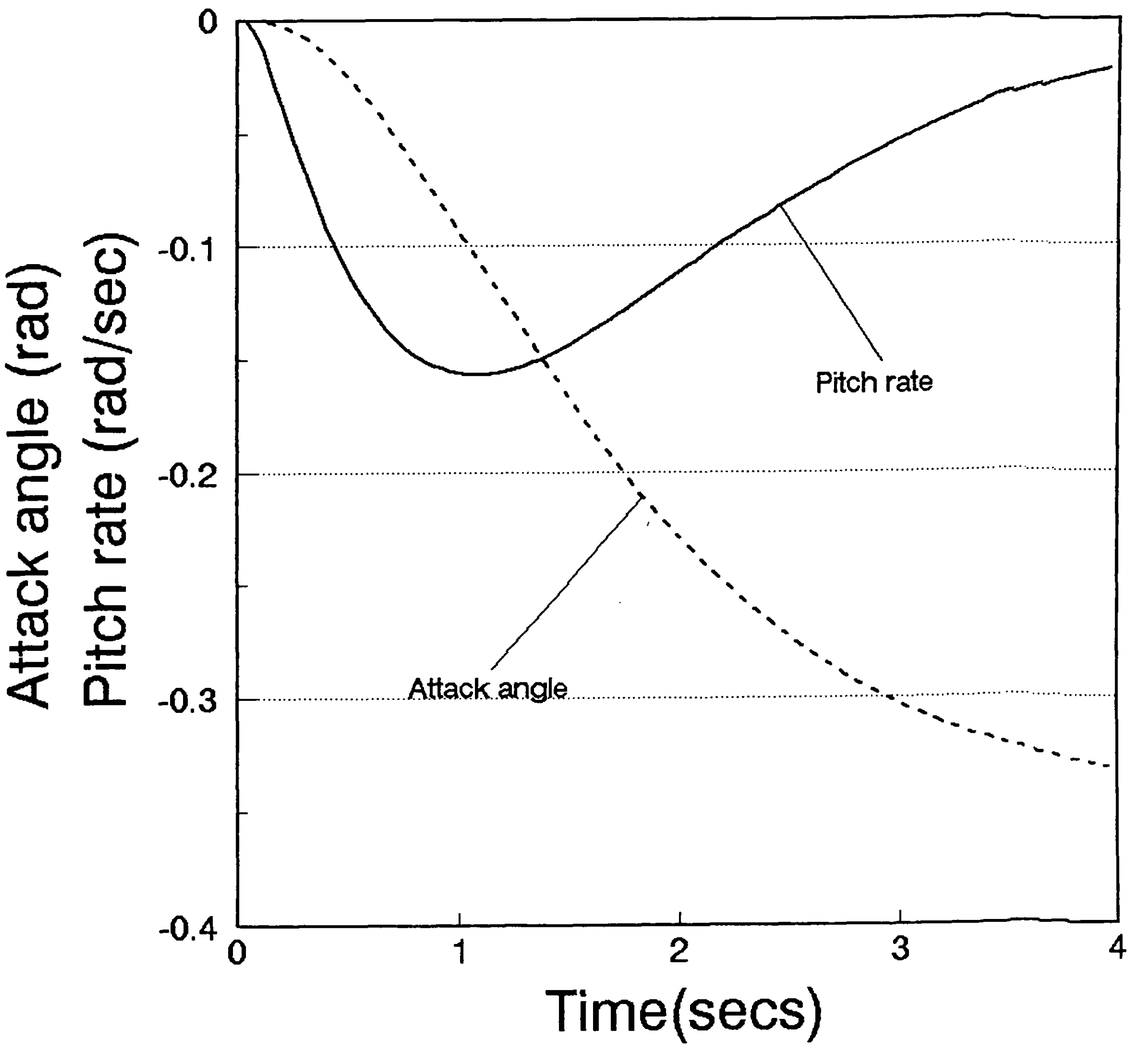


Fig.5.4. The outputs of the process

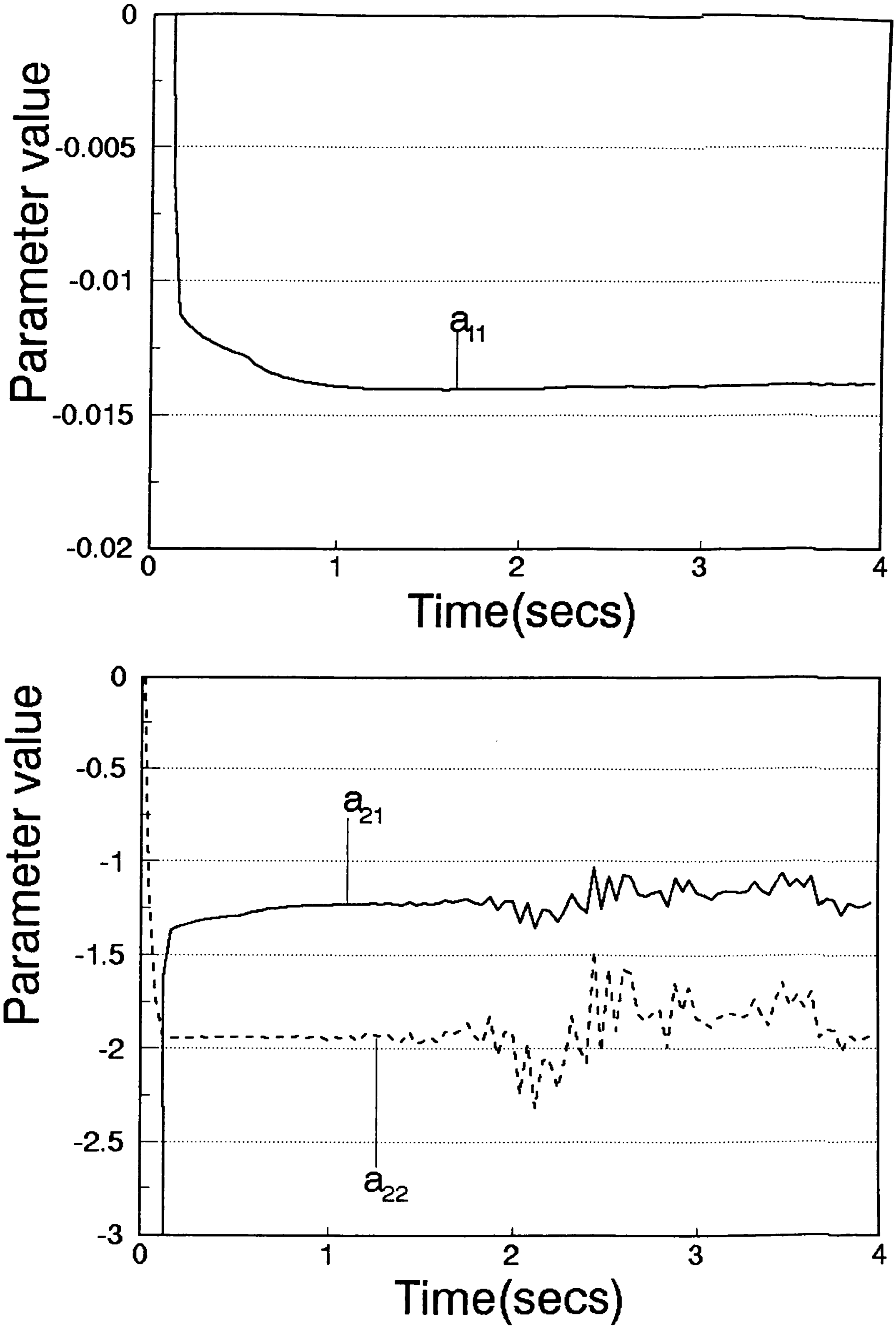


Fig.5.5. The identification results with each three steps.

Identification error incerase with aproaching to the steady state.

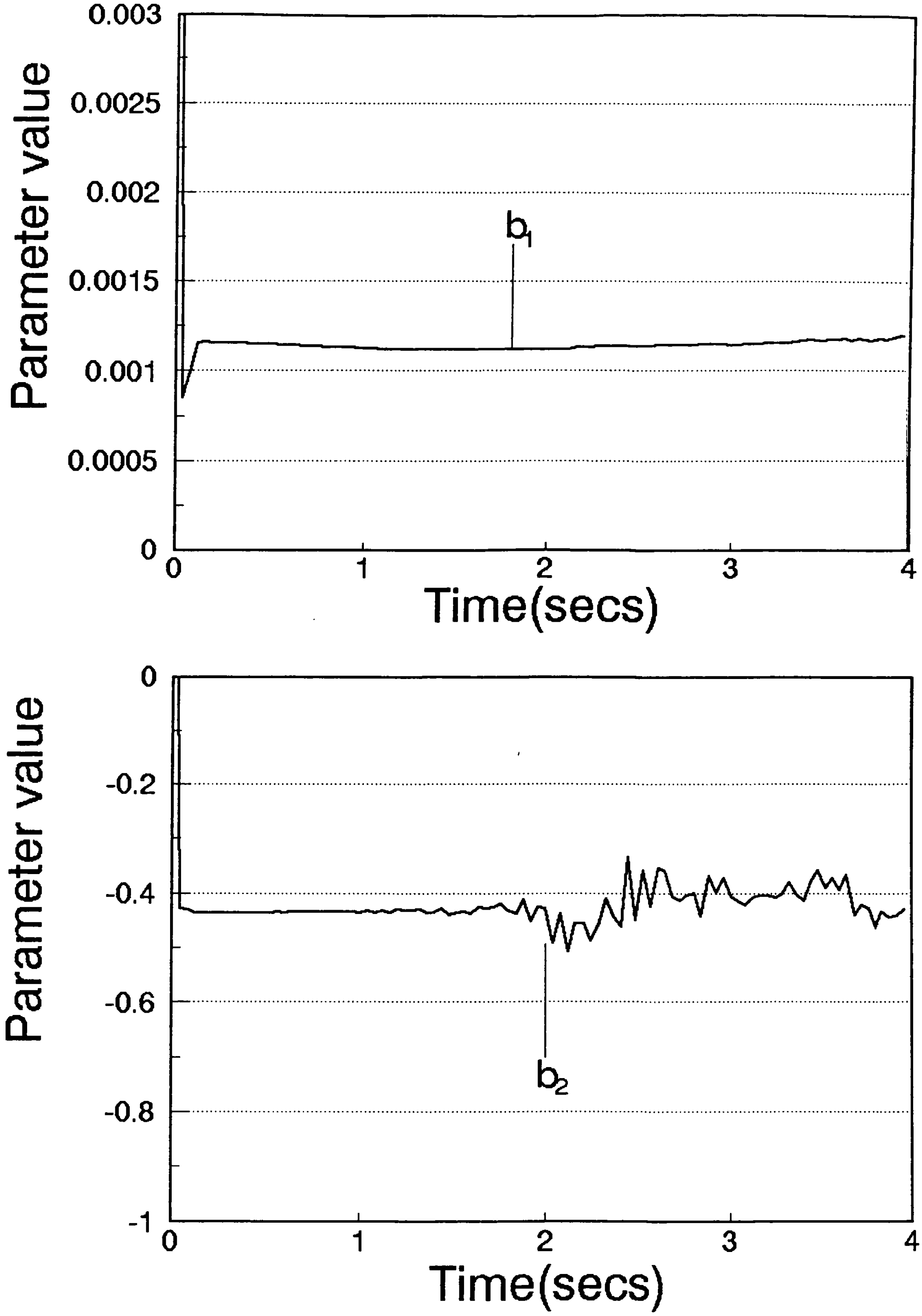
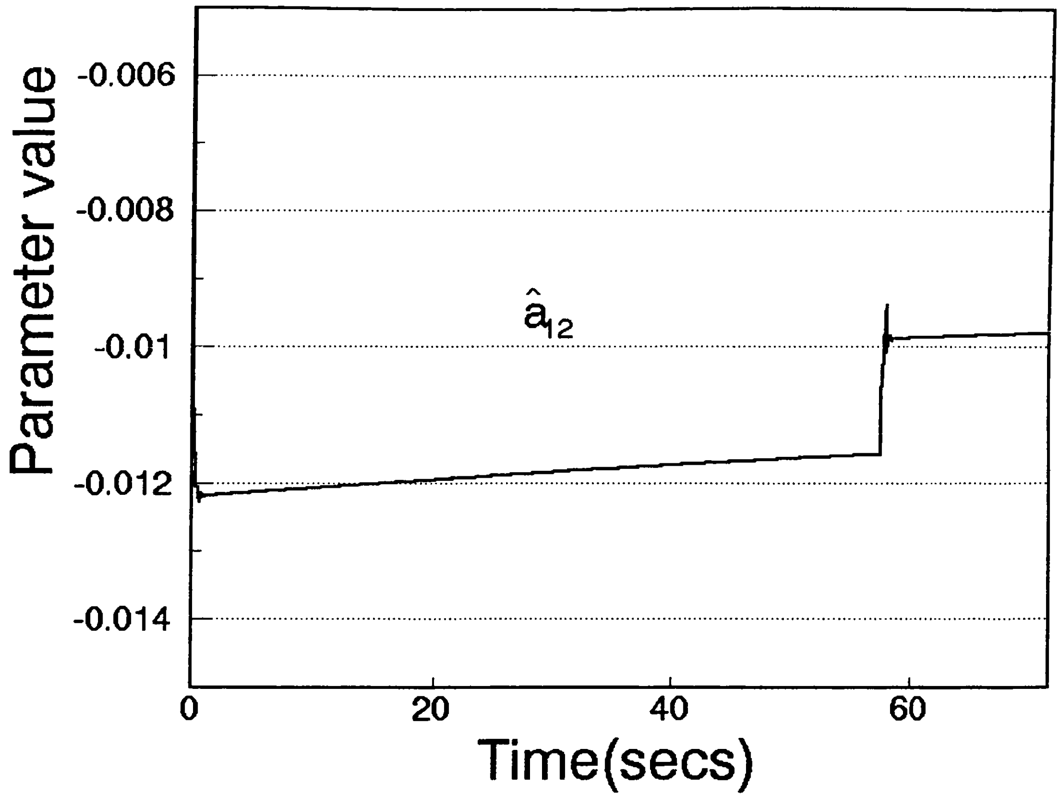


Fig.5.6. The identification of the control parameters with each three steps. Identification error incease with approaching to the steady state.



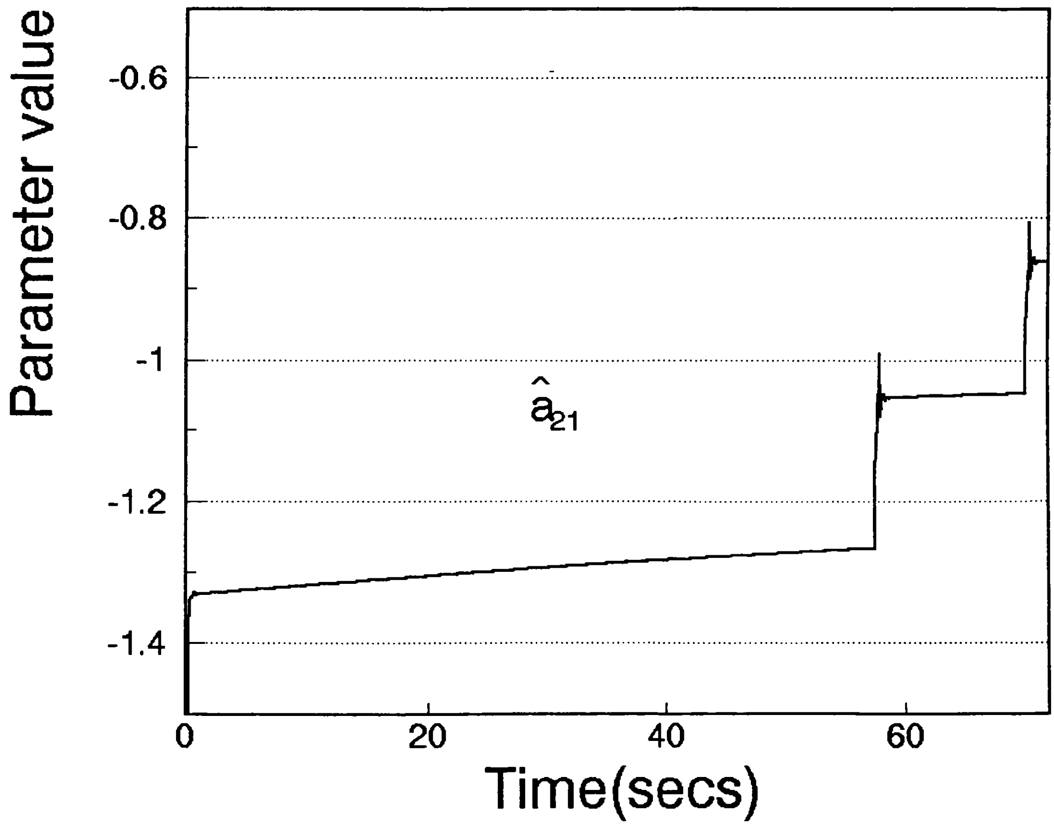


Fig.5.7. The identification of the time-varying parameters

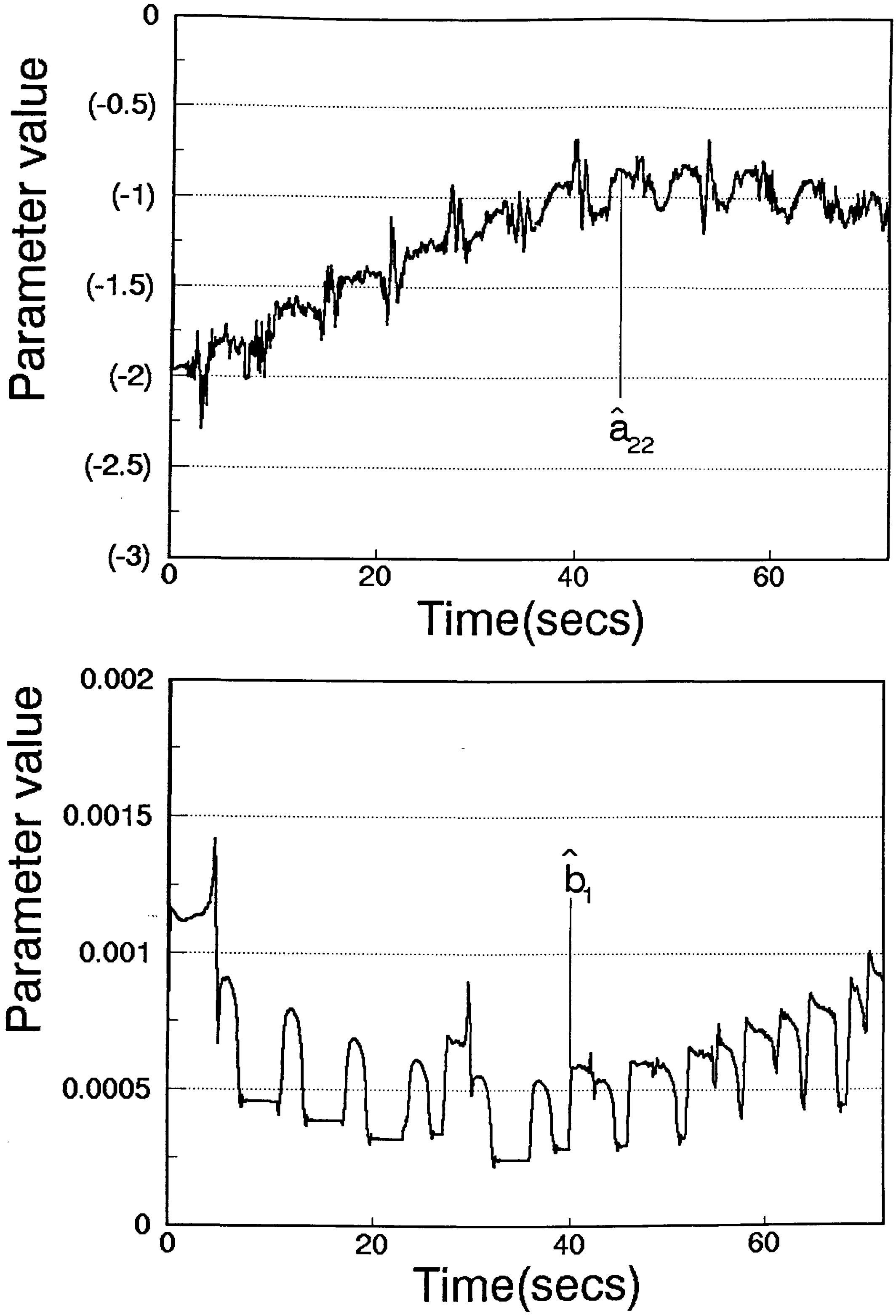


Fig.5.8. The identification of time-varying parameters. The reason of fluctuation of the parameter values are explained in section 5.5.2..

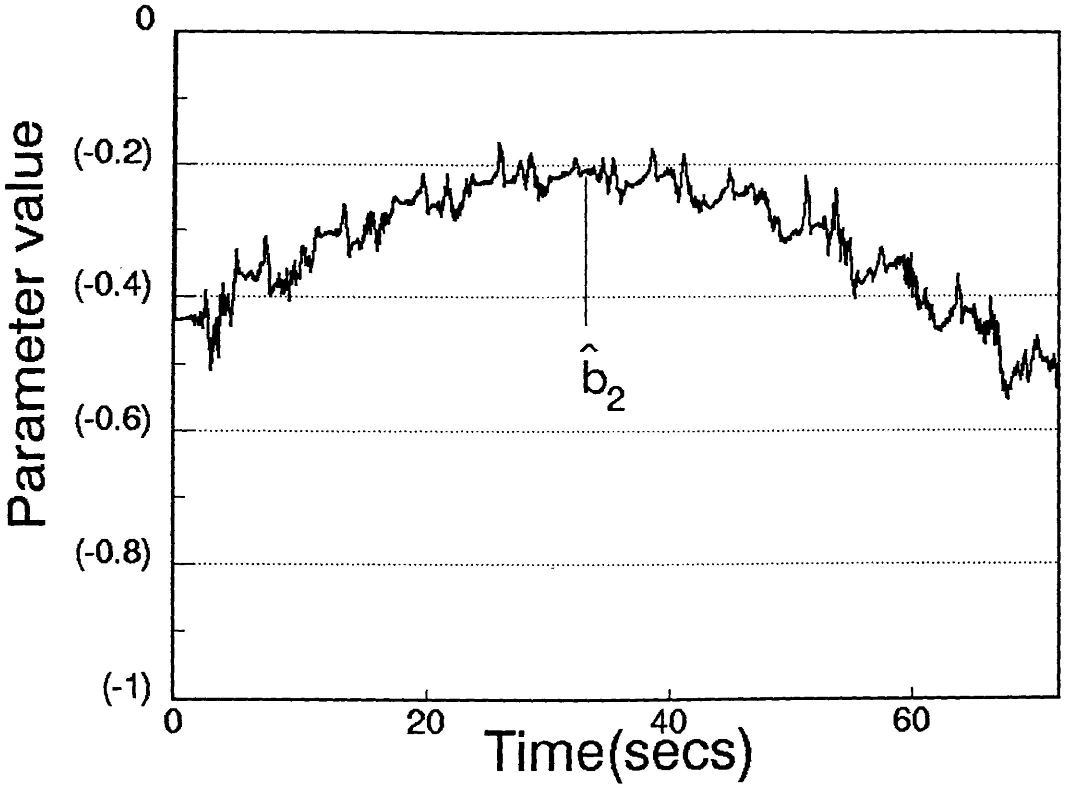


Fig.5.9. The identification of time-varying parameters

The following figures 5.10 to 5.24 help to give an idea about the relation between the identification feasibility and the noise. It can easily be seen from the figures that increasing noise causes the error to increase. But the identification is still feasible until the noise level reaches to 0.01 . Only a_{11} identification is impossible, because its effect on the system is much more smaller than the other. The identification routine is based on noise elimination with time averaging, but the noise effect can not be made smaller than the a_{11} effect in 30 steps.

The figures 5.25 to 5.40 show that the step number increasing from 30 tO 60 is enough to elimination the noise effect. The a_{11} effect on the system output is so small that even 60 steps noise elimination is not enough to sense it.

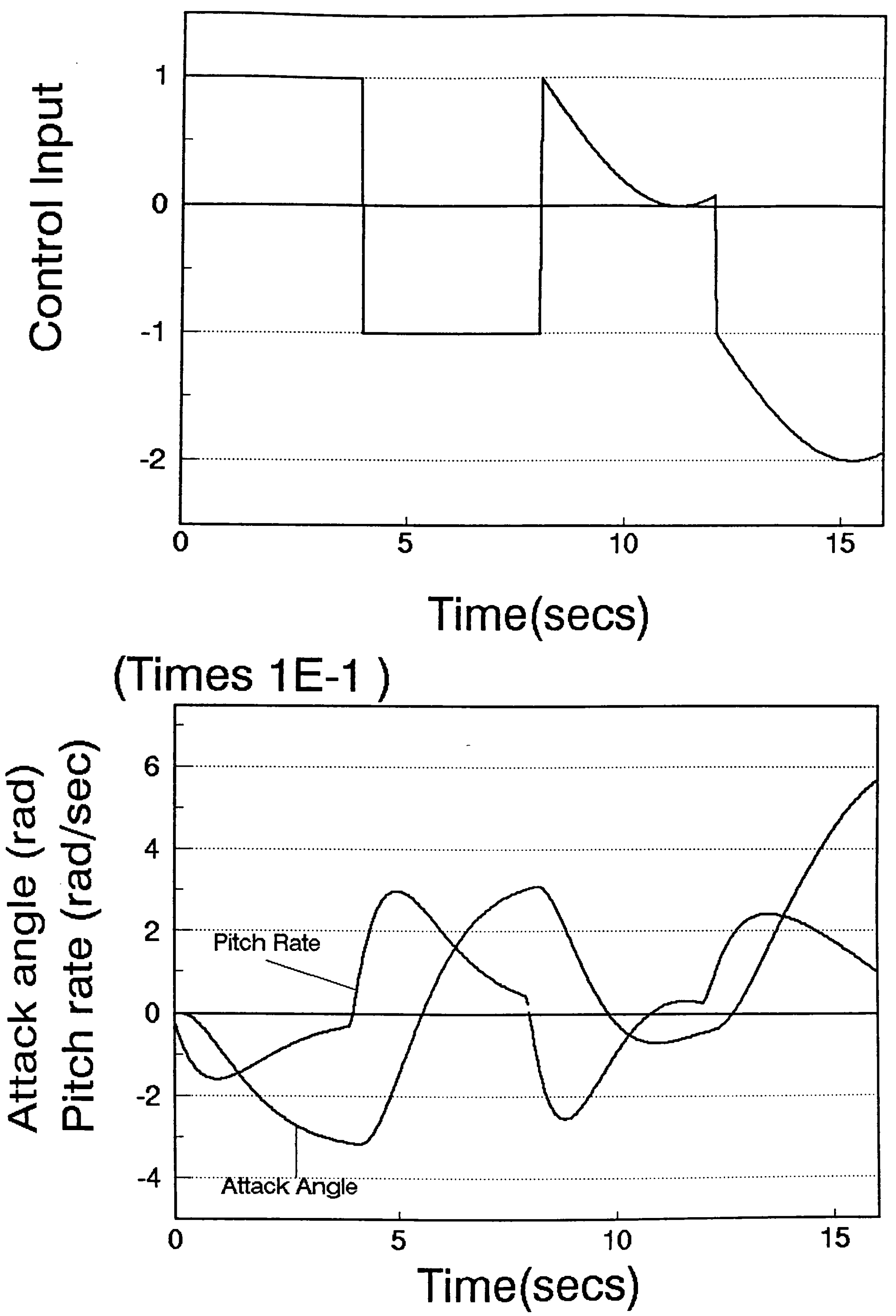


Fig.5.10. The control input and output observation for the noise free system.

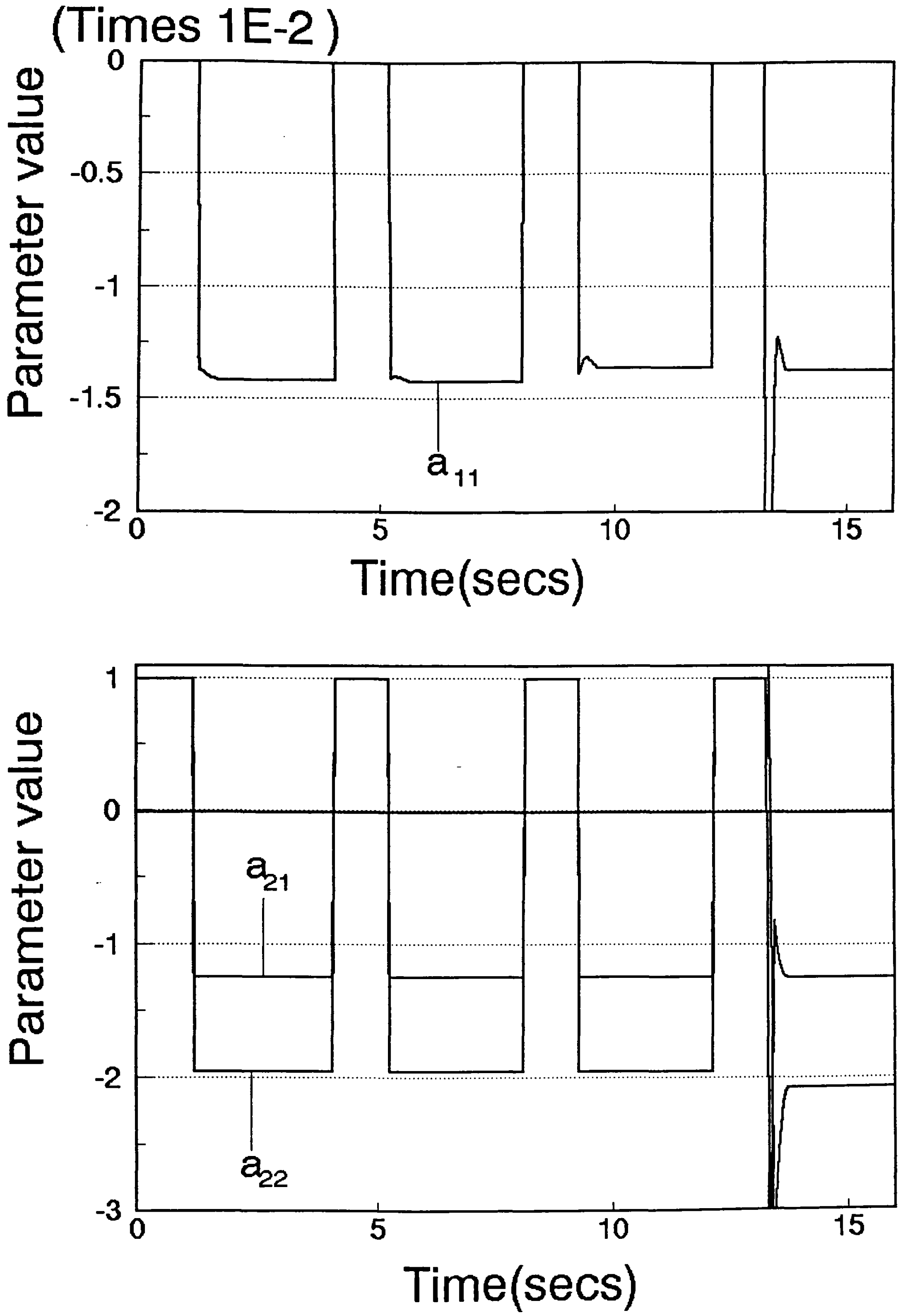


Fig.5.11. The identification of parameters for noise-free observation. Parameter values can be identified after 30th step, correctly. This operation has been done with data of Fig.5.10.

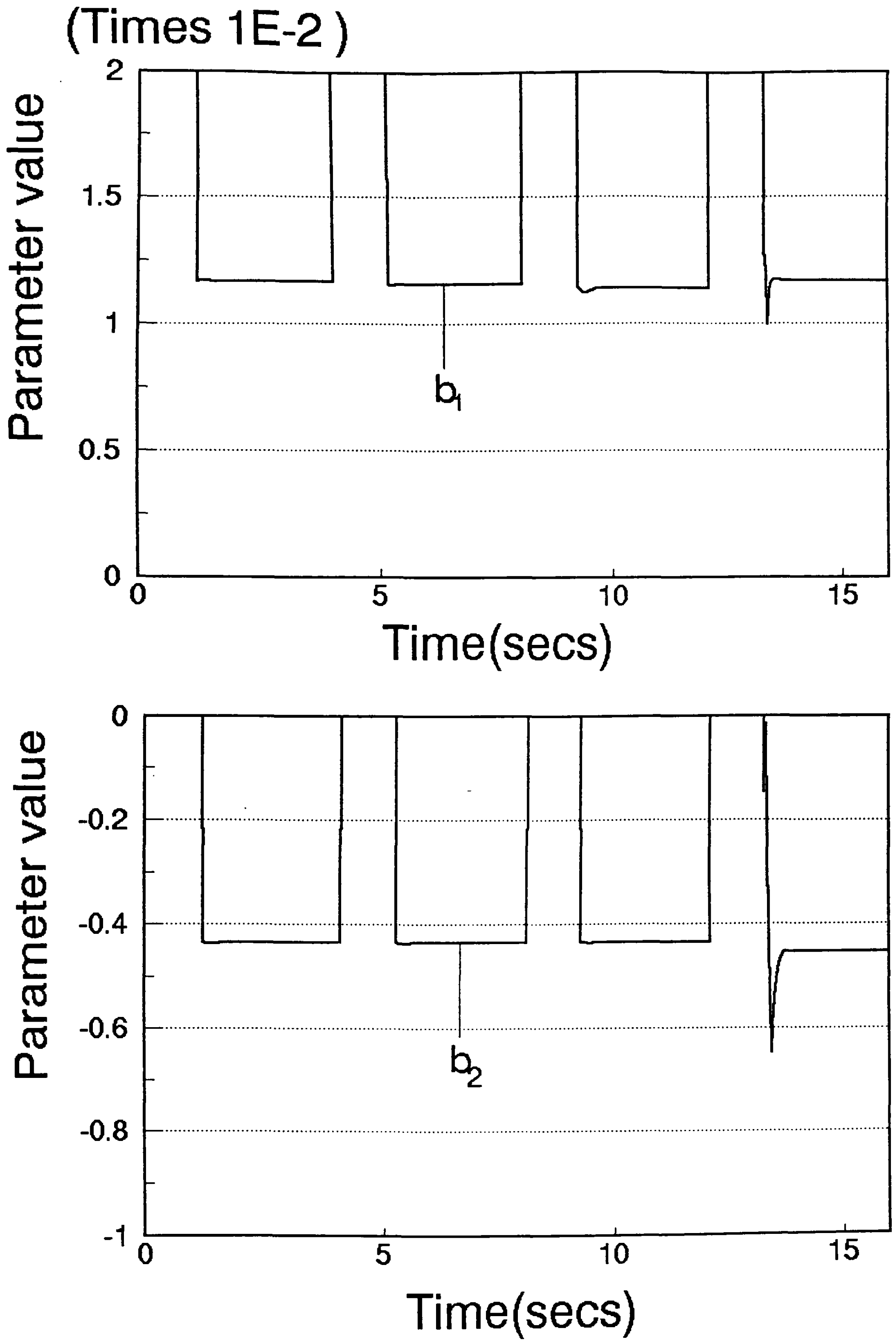


Fig.5.12. The identification of the control parameters for noise-free observation. Data are the same with Fig.5.11.

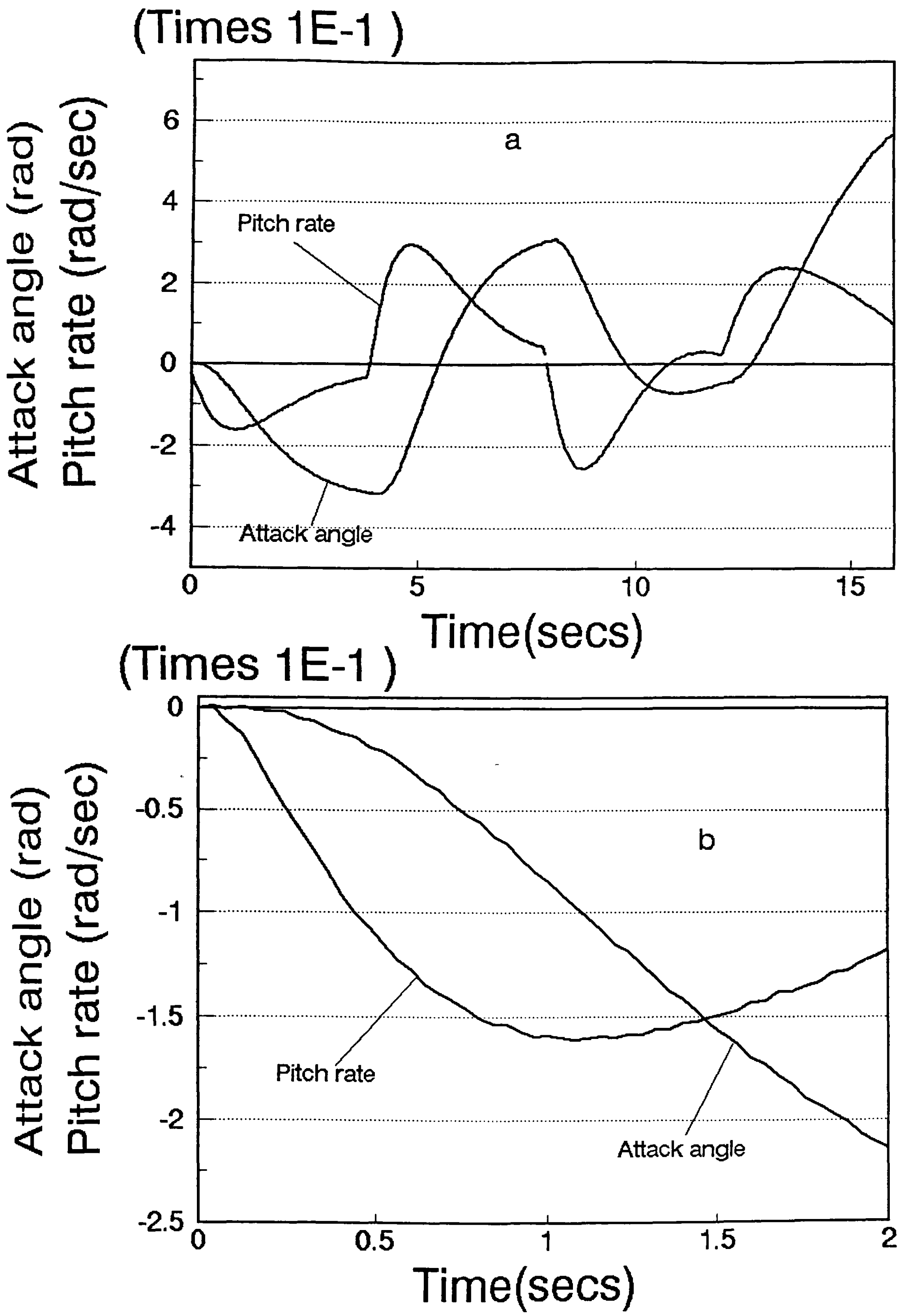


Fig.5.13. Output observations for noise amplitude 0.001

Fig.b. is the enlarging wiev of Fig.a.

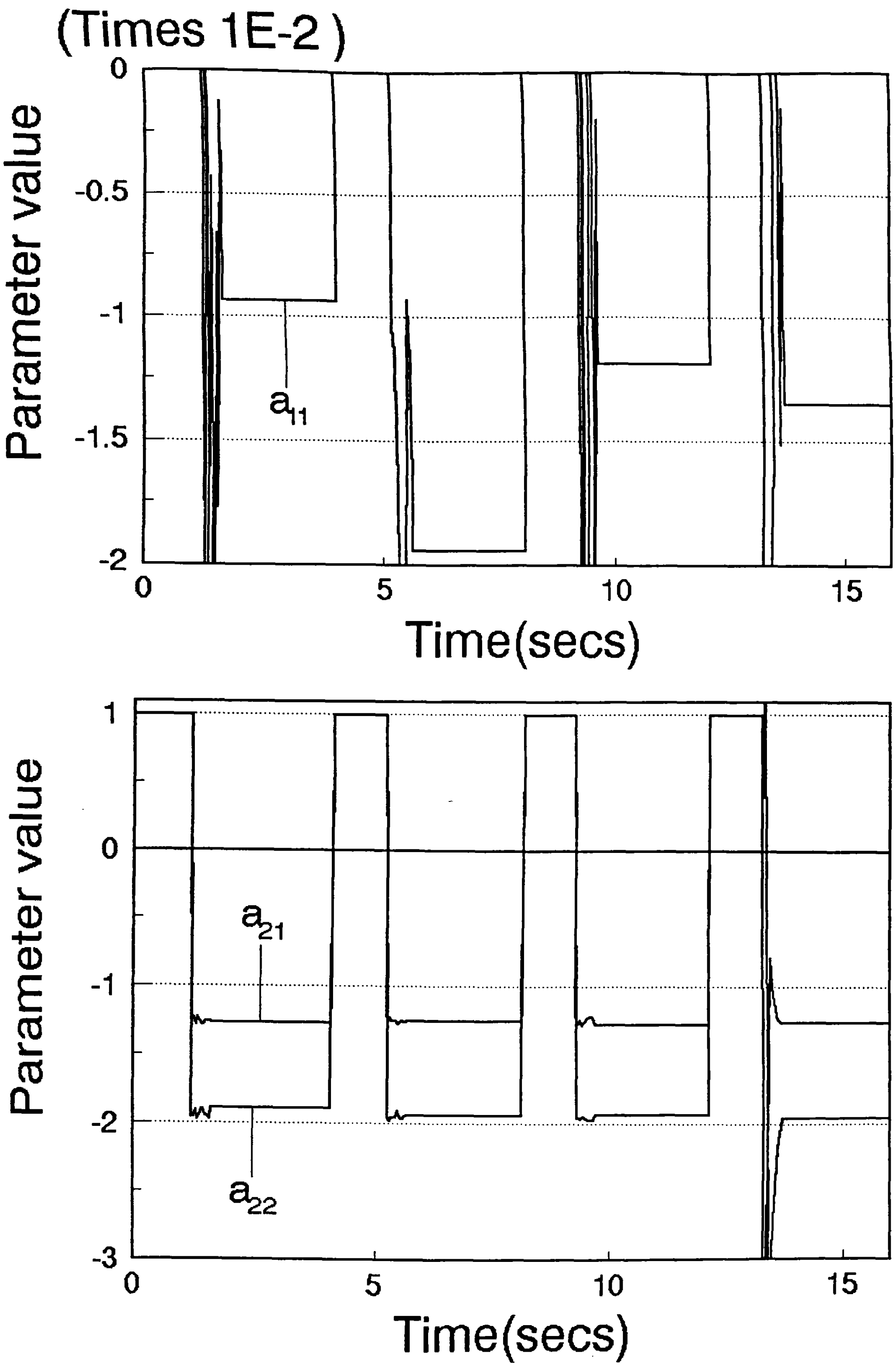
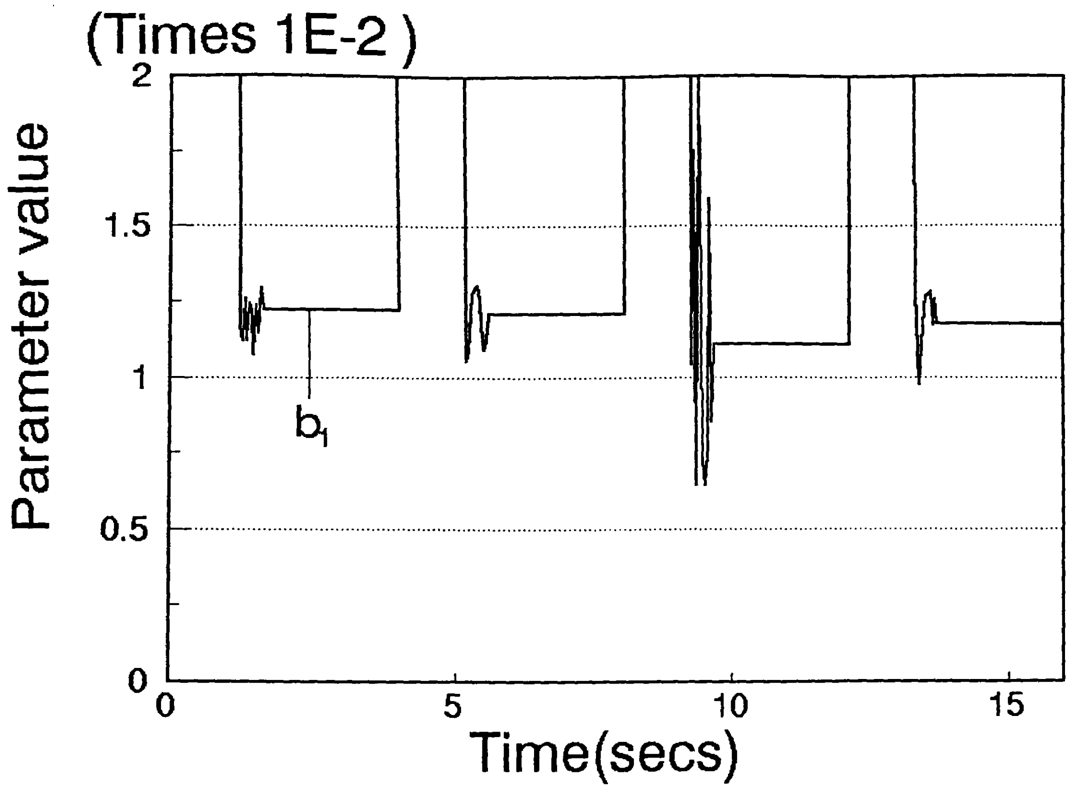


Fig.5.14. The identification of the system parameter with 30 step block data. Data are mixed with noise as Fig.5.13. a_{11} estimation is more affected from noise than other parameter.



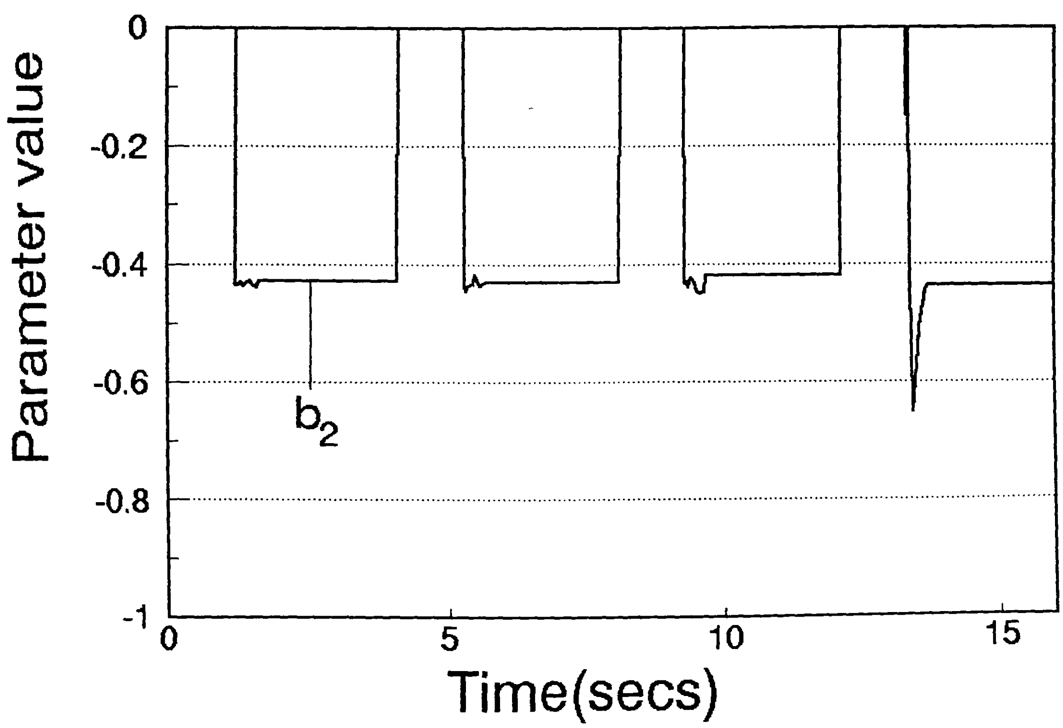
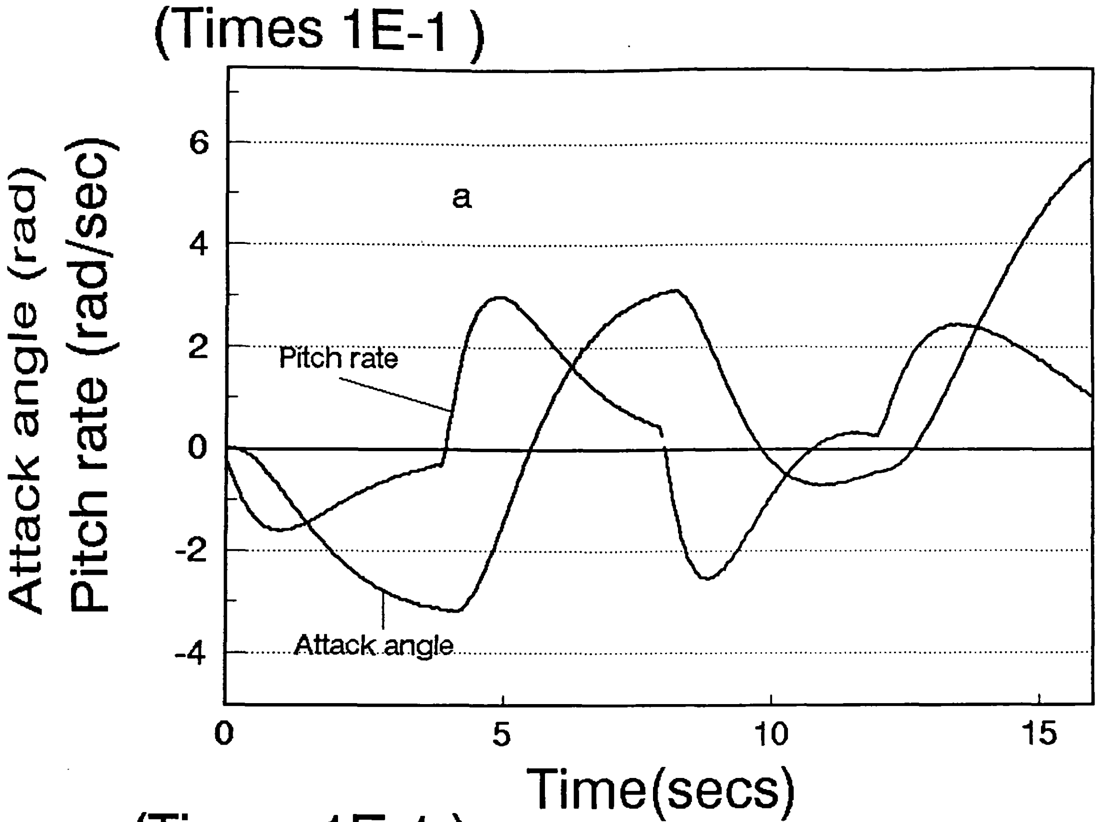


Fig.5.15. The control parameter identification with observation of Fig.5.13. Both of them are not affected from noise.



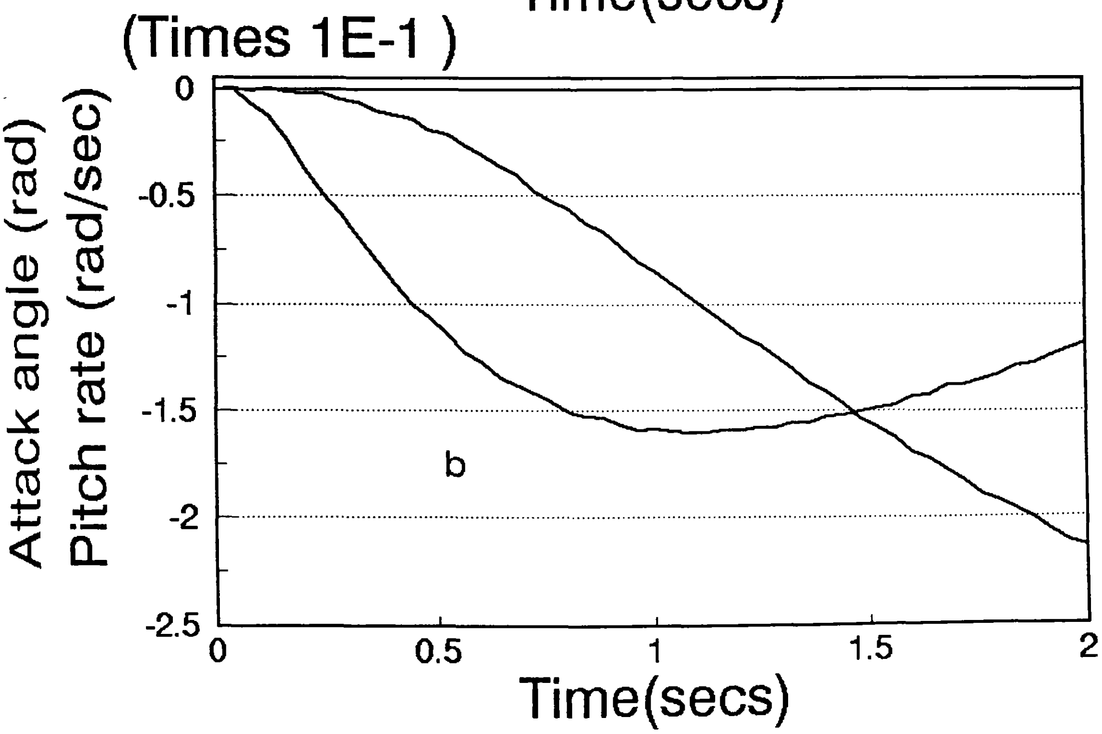


Fig.5.16. The system outputs for noise amplitude **0.003**

Fig.b. is the enlarging wiev of Fig.a.

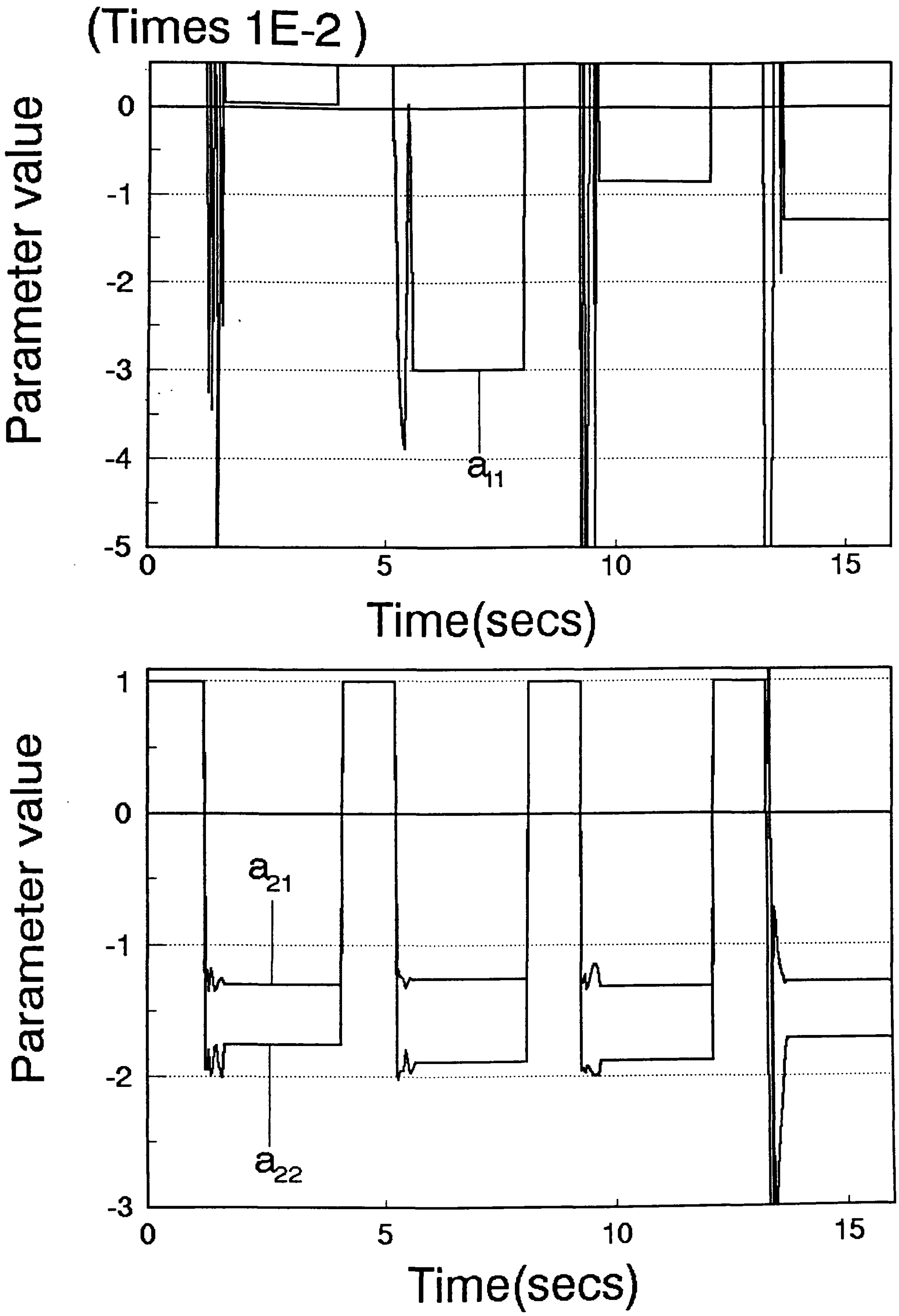


Fig.5.17. System parameter identification result.a_{11} value may be wrong. But other dominant parameters can be identified.

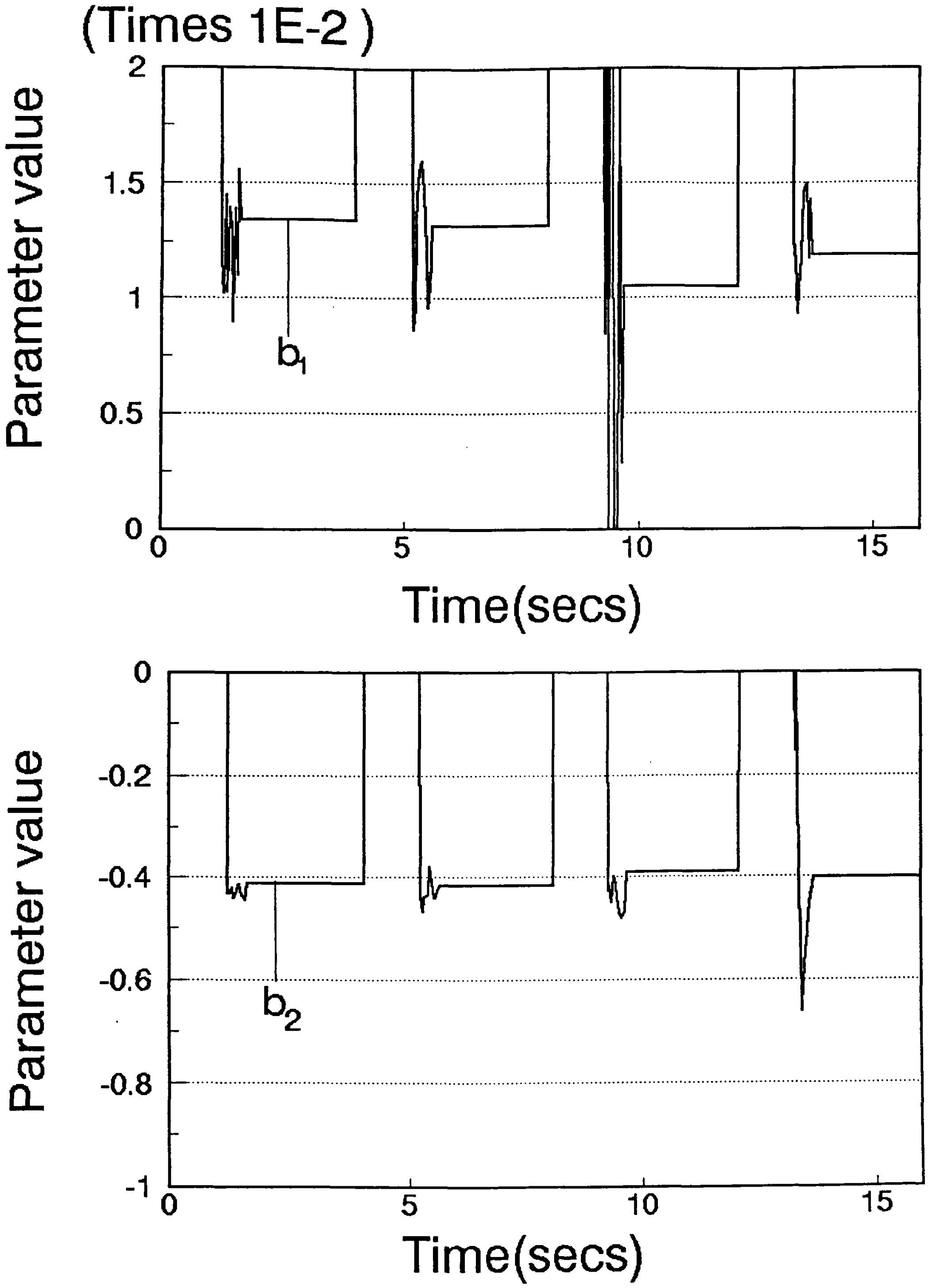


Fig.5.18. The control parameter identification for noise amplitude 0.003.

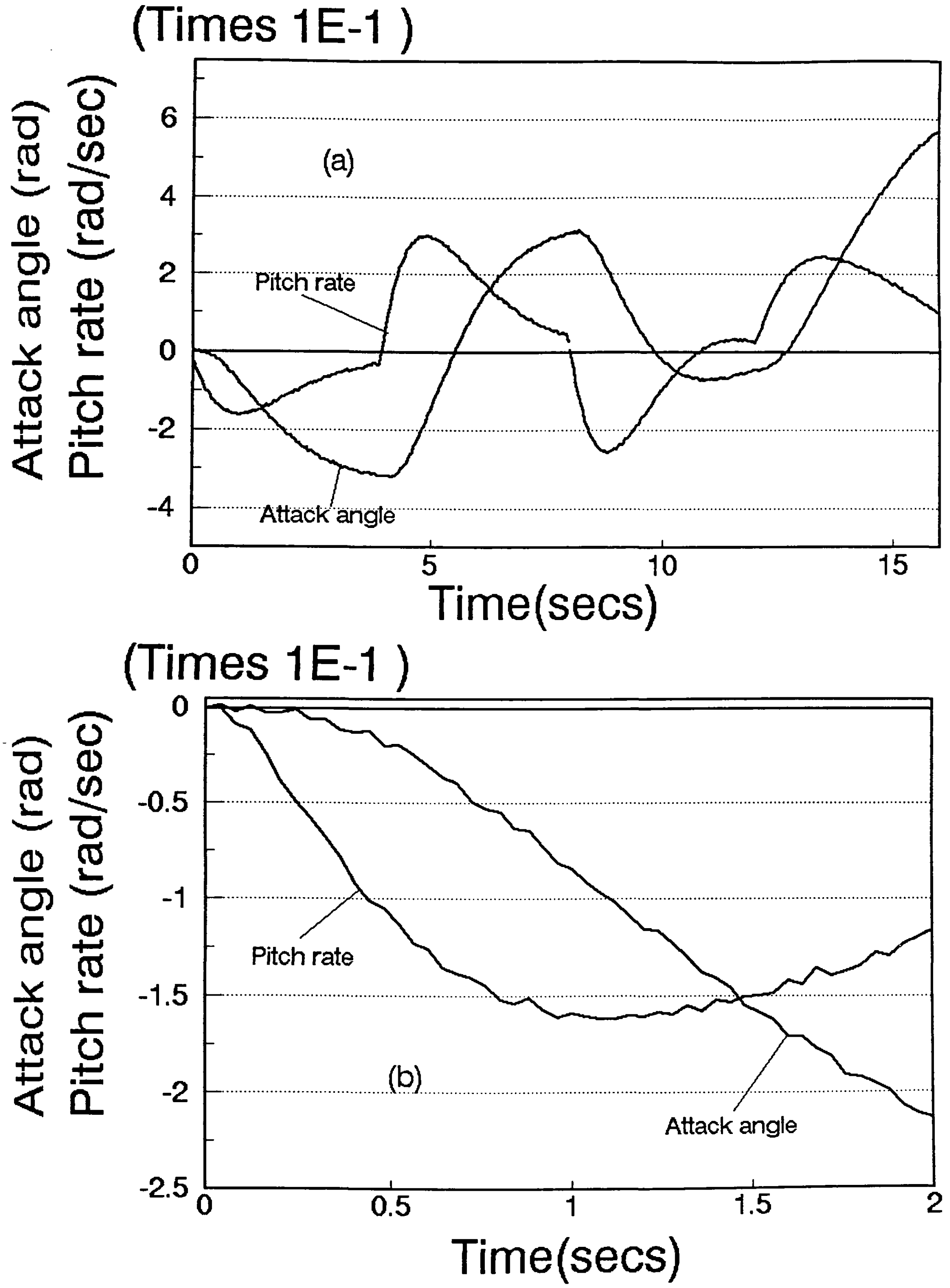


Fig.5.19. Output observation for noise amplitude **0.007**

Fig.b. is the enlarging wiev of Fig.a.

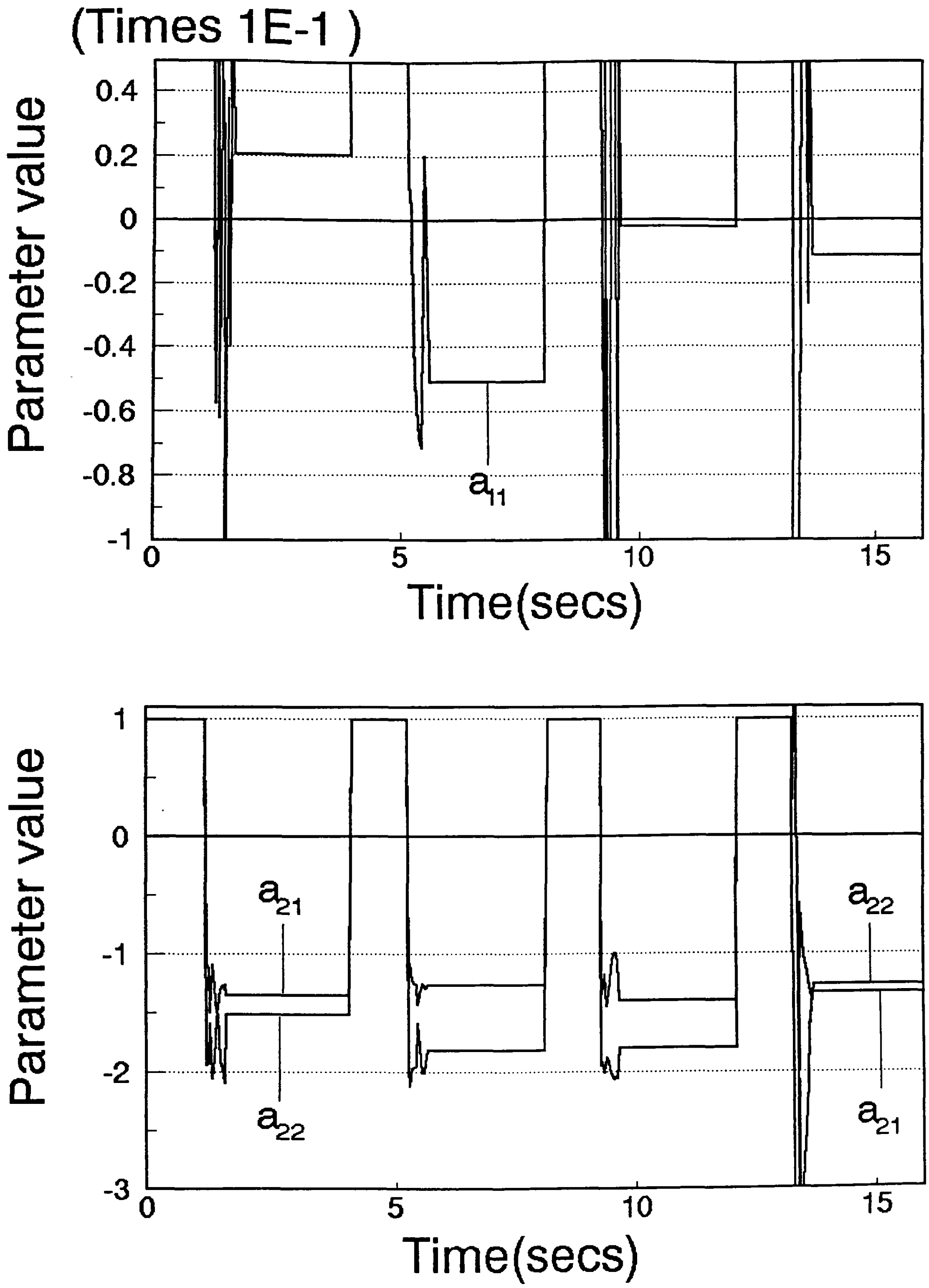


Fig.5.20. The identification of the system parameter with 30 step block data. It has been done with Fig.5.19. data. Error increases for all parameter.

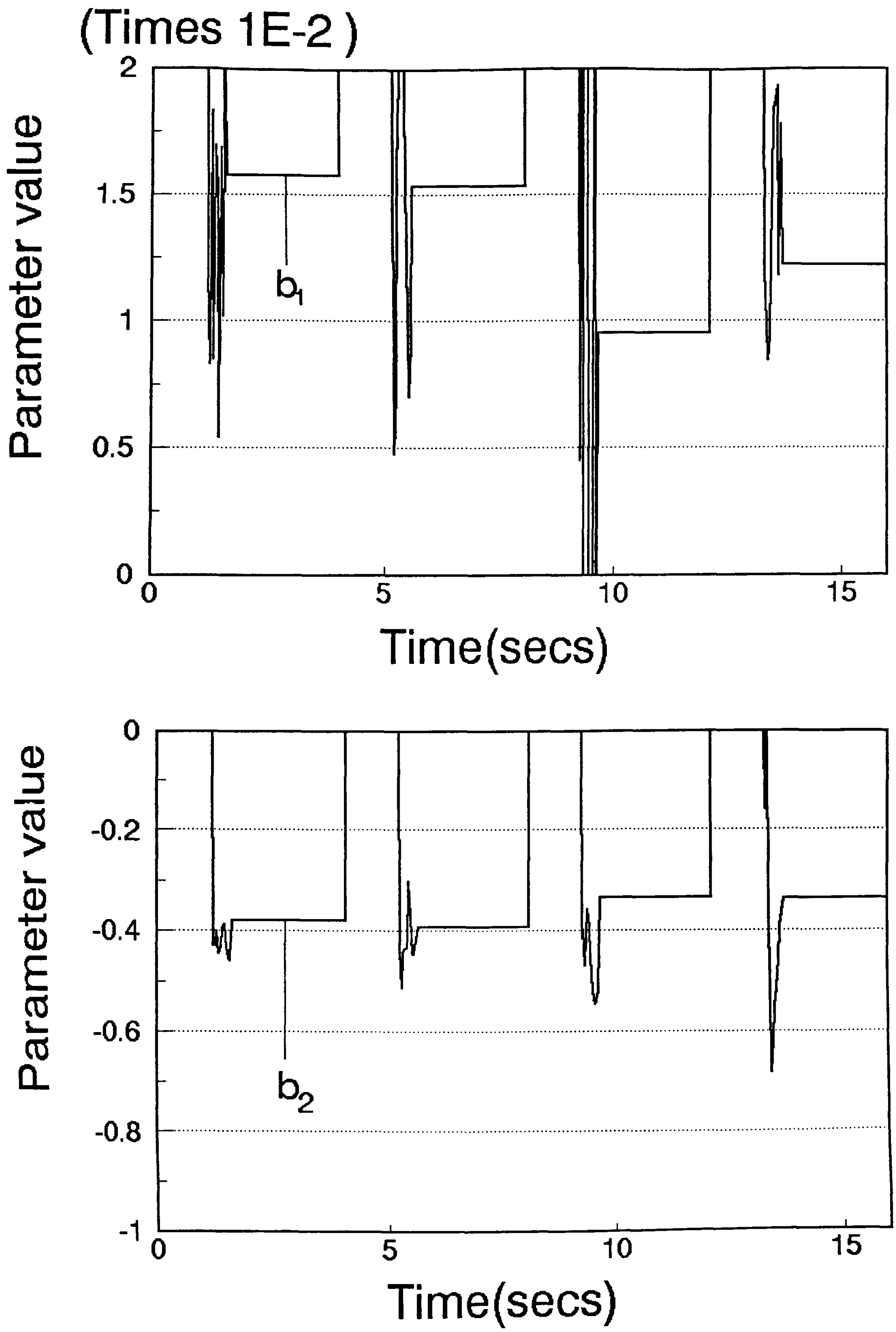
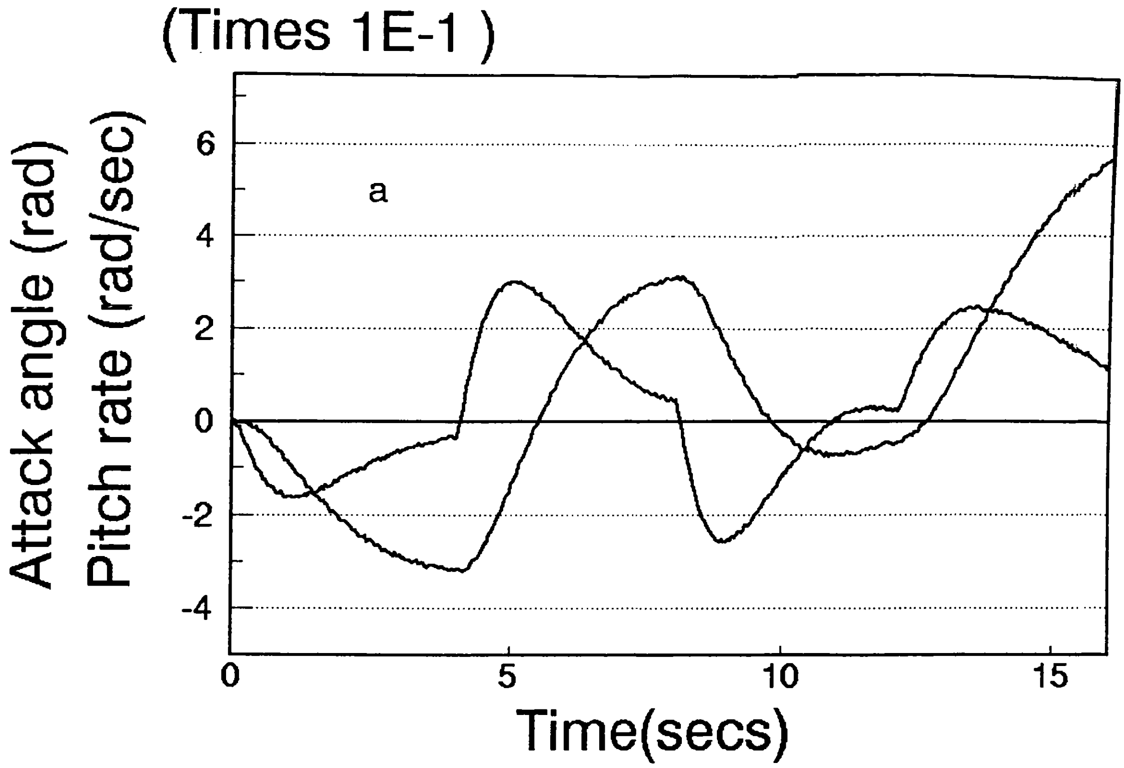


Fig.5.21. The identification of the control parameters. Error is still ignorant. Identification has been achieved with Fig.5.19. data



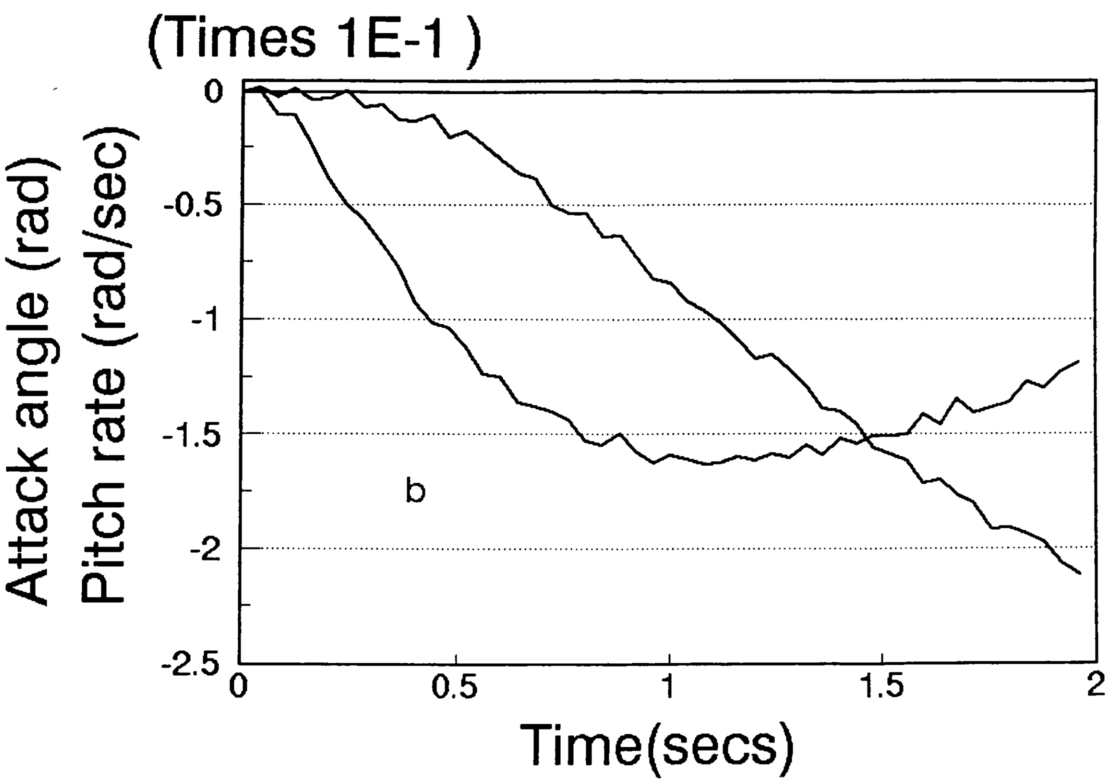


Fig.5.22. Output observation for noise amplitude 0.01

Fig.b. is the enlarging wiev of Fig.a.

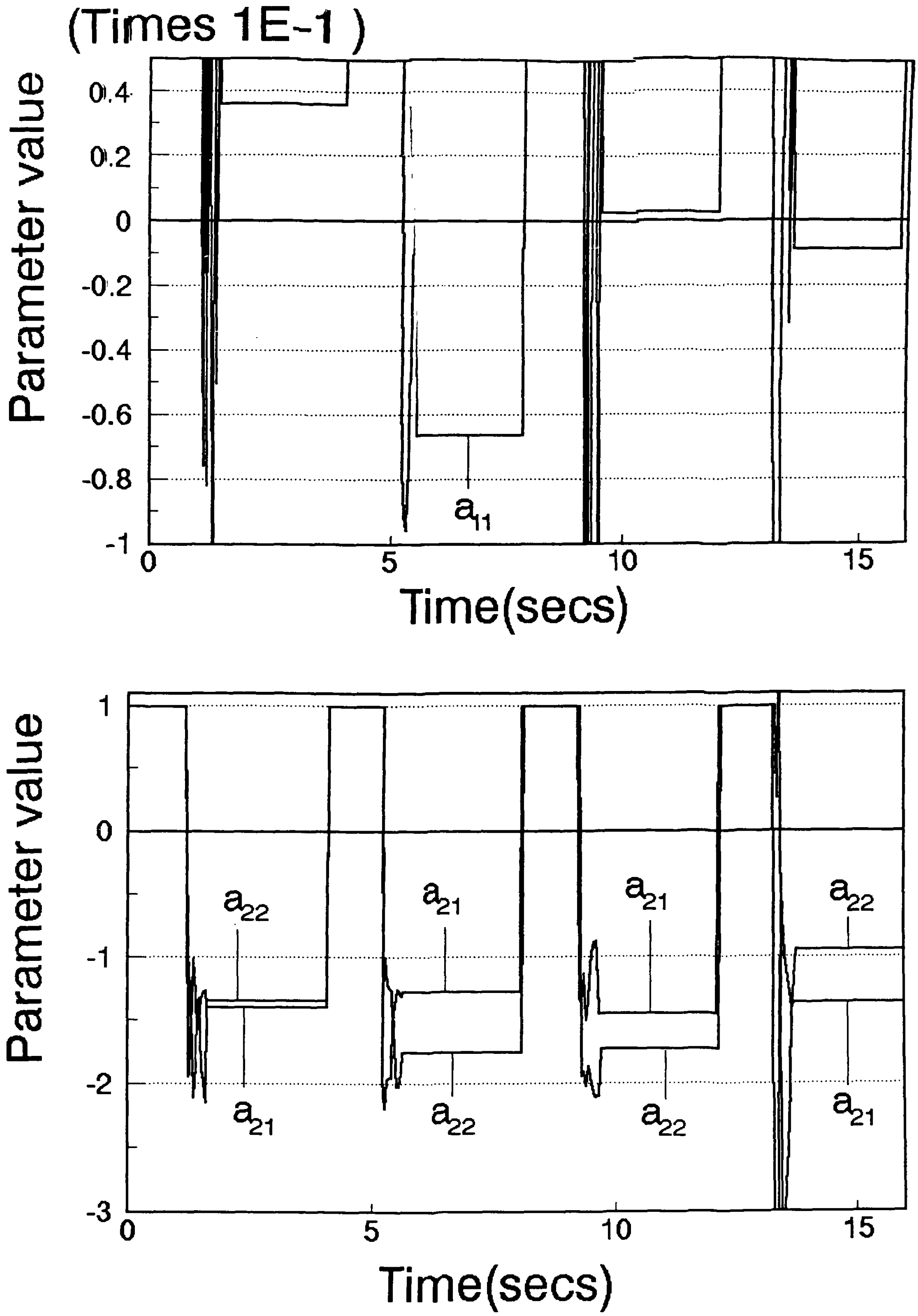


Fig.5.23. The identification error increase up to 50% for 30 step time-averaging with noise amplitude 0.01 .

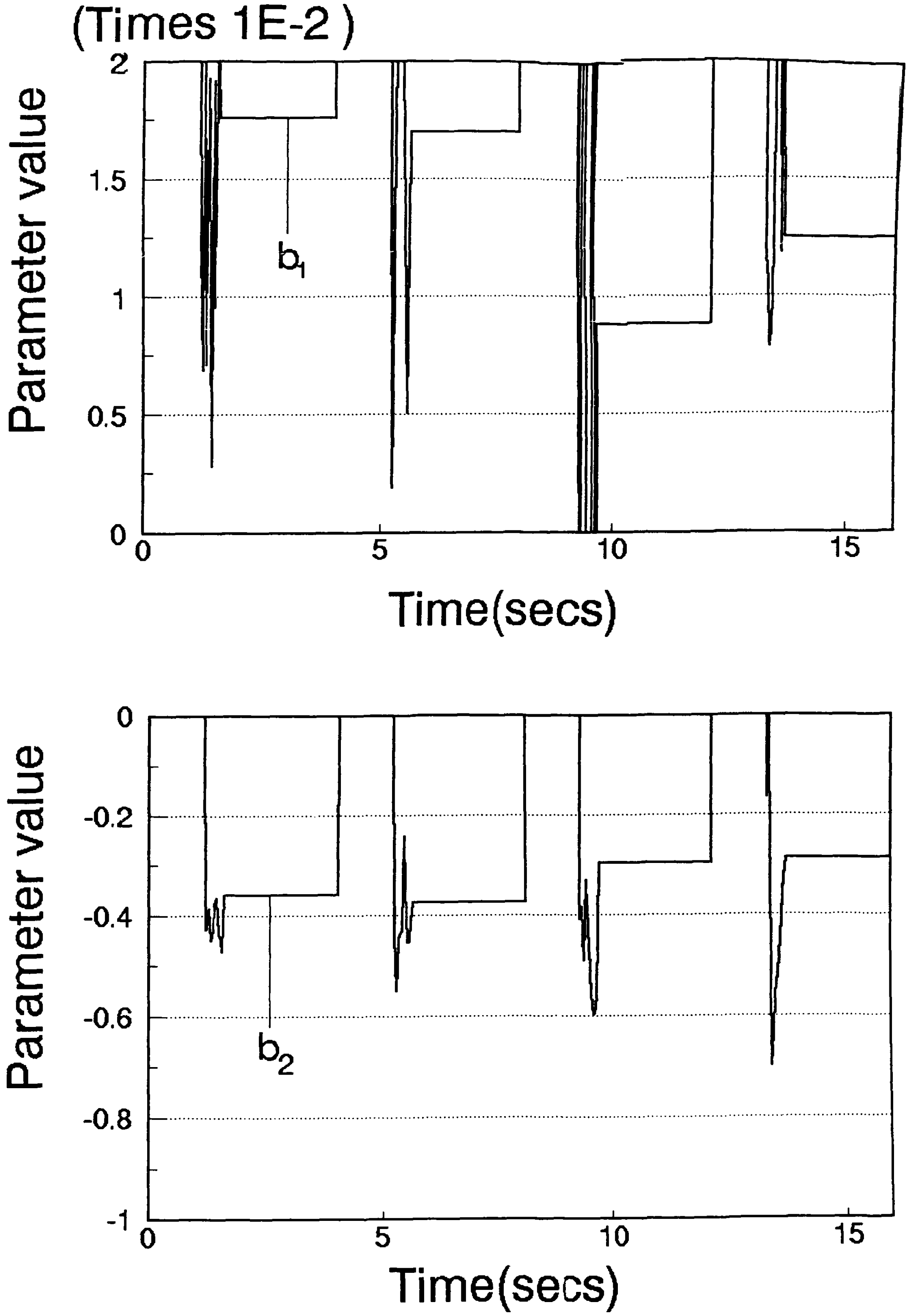


Fig.5.24. The identification of control parameters. Error is still less than 25% for the observation of Fig.5.22.

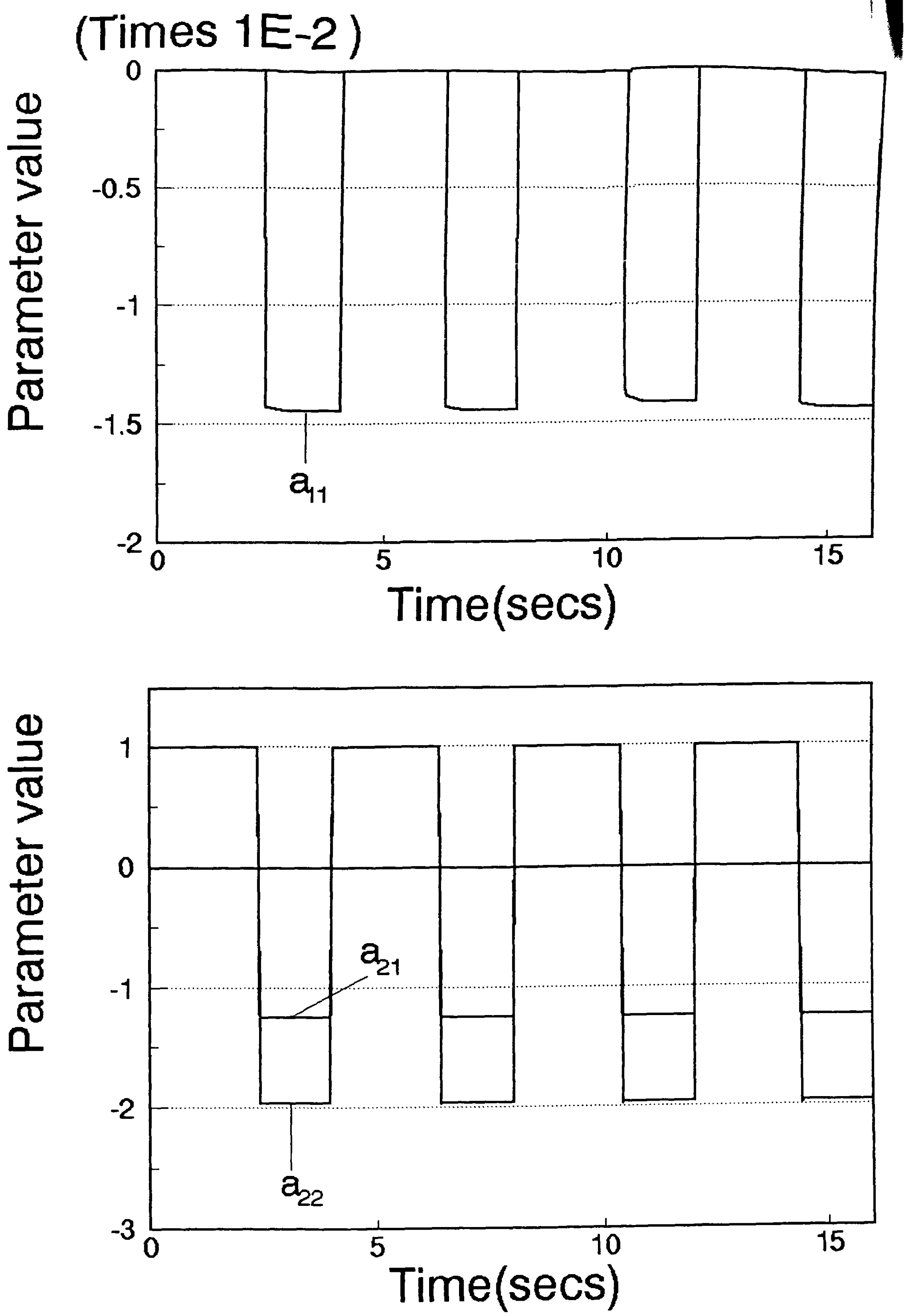


Fig.5.25. The identification results of noise-free system with 60 step time-averaging. Identification is achieved accurately after 60th step.

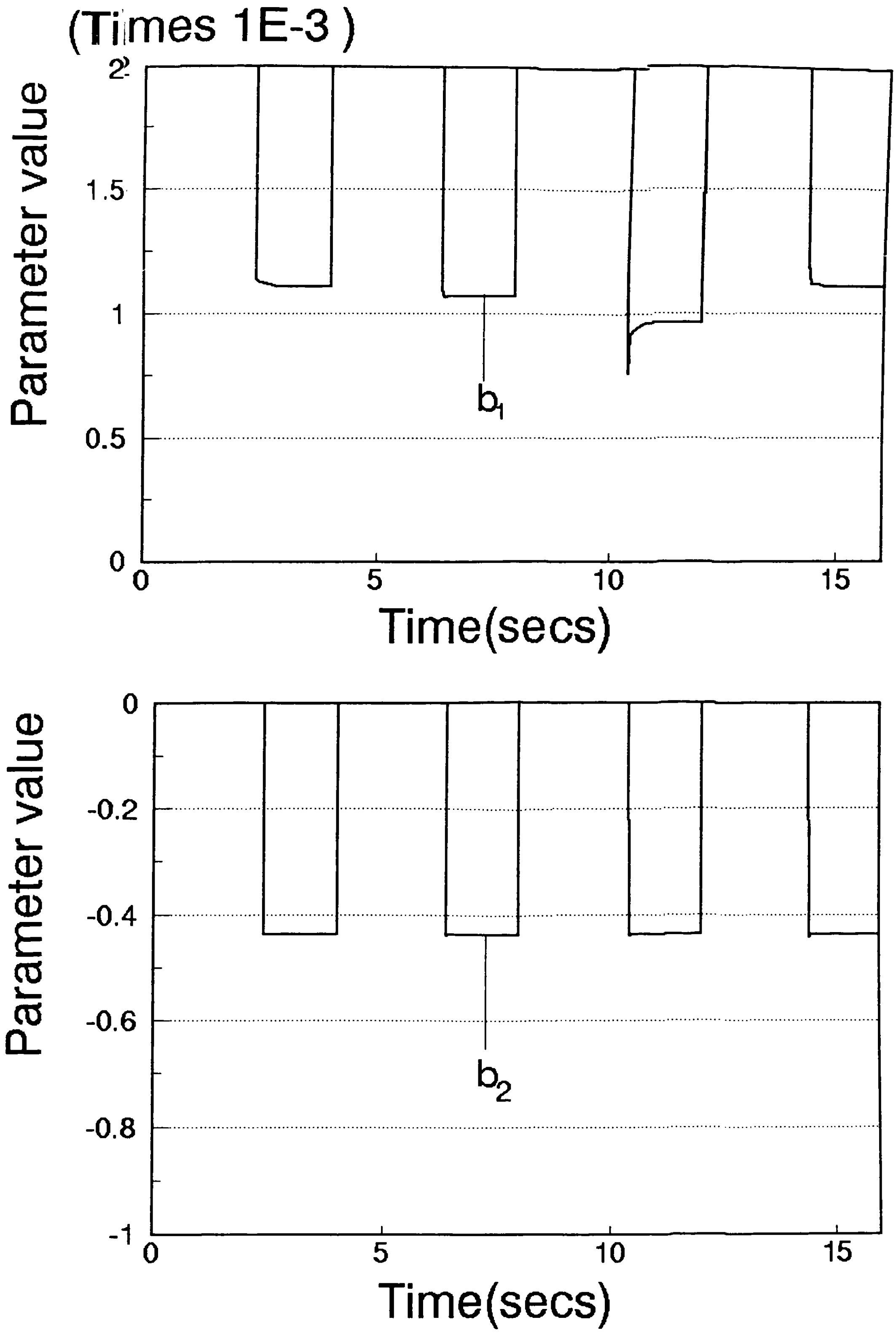


Fig.5.26. The identification of the control parameters of noise-free system with 60 step time-averaging.

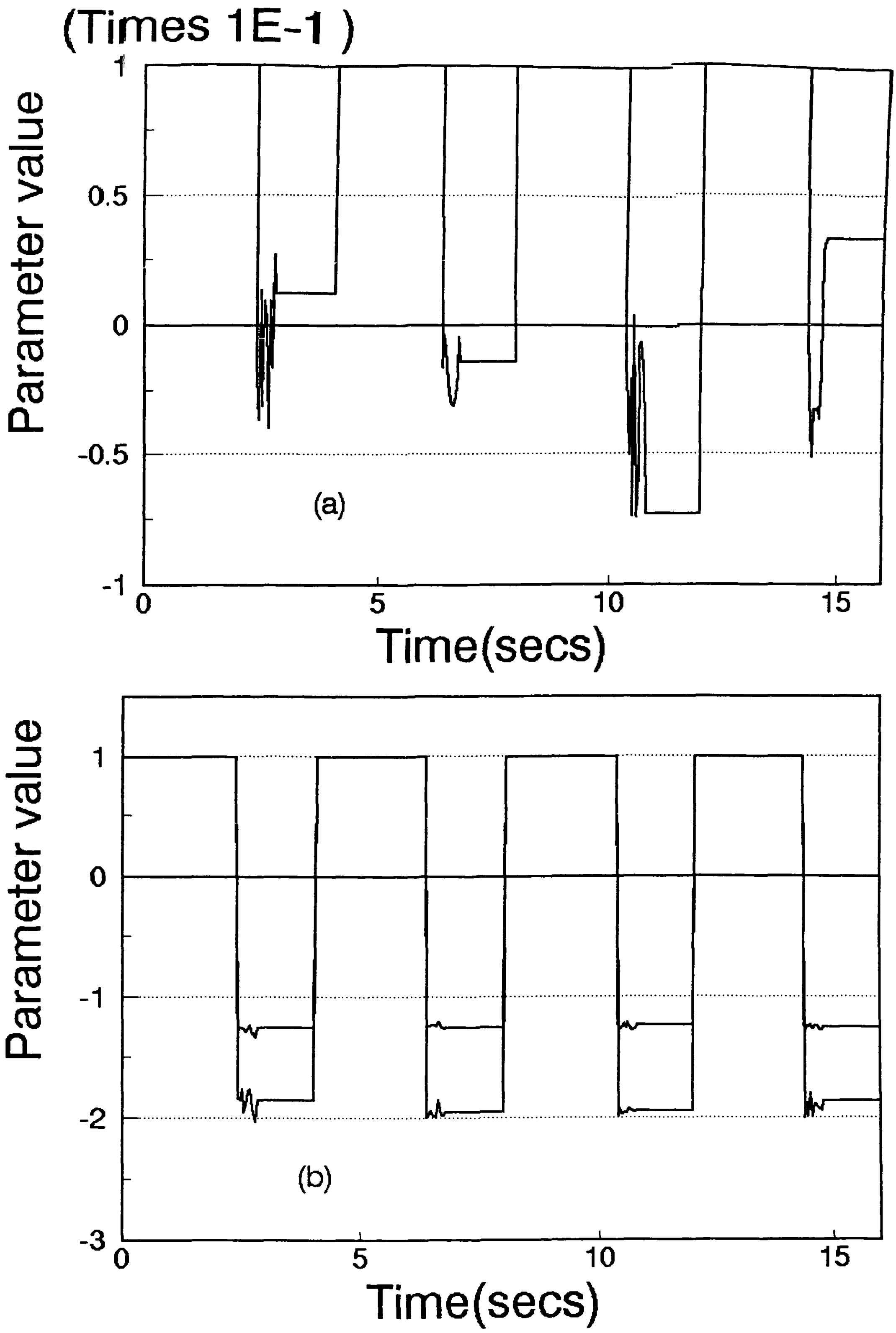


Fig.5.27. The identification result for the noise 0.01 with 60 step time-averaging. It can be seen that dominant parameter can be identified, accurately. It was not possible with 30 step data.

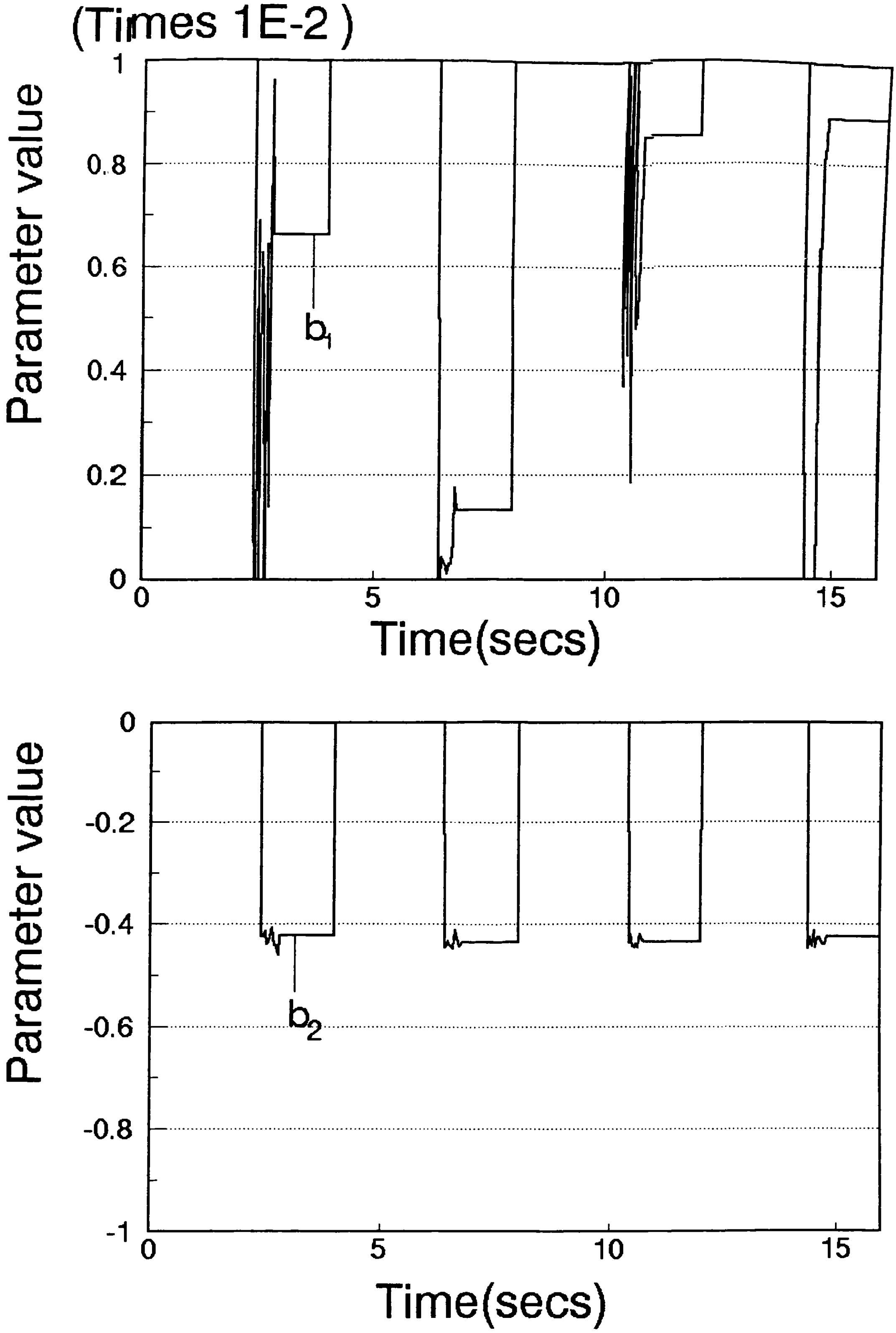


Fig.5.28. The control parameter identification for noise 0.01 with 60 step time-averaging. Error decreases from 25% to 1-2%, when it is compared with 30 step time averaging.

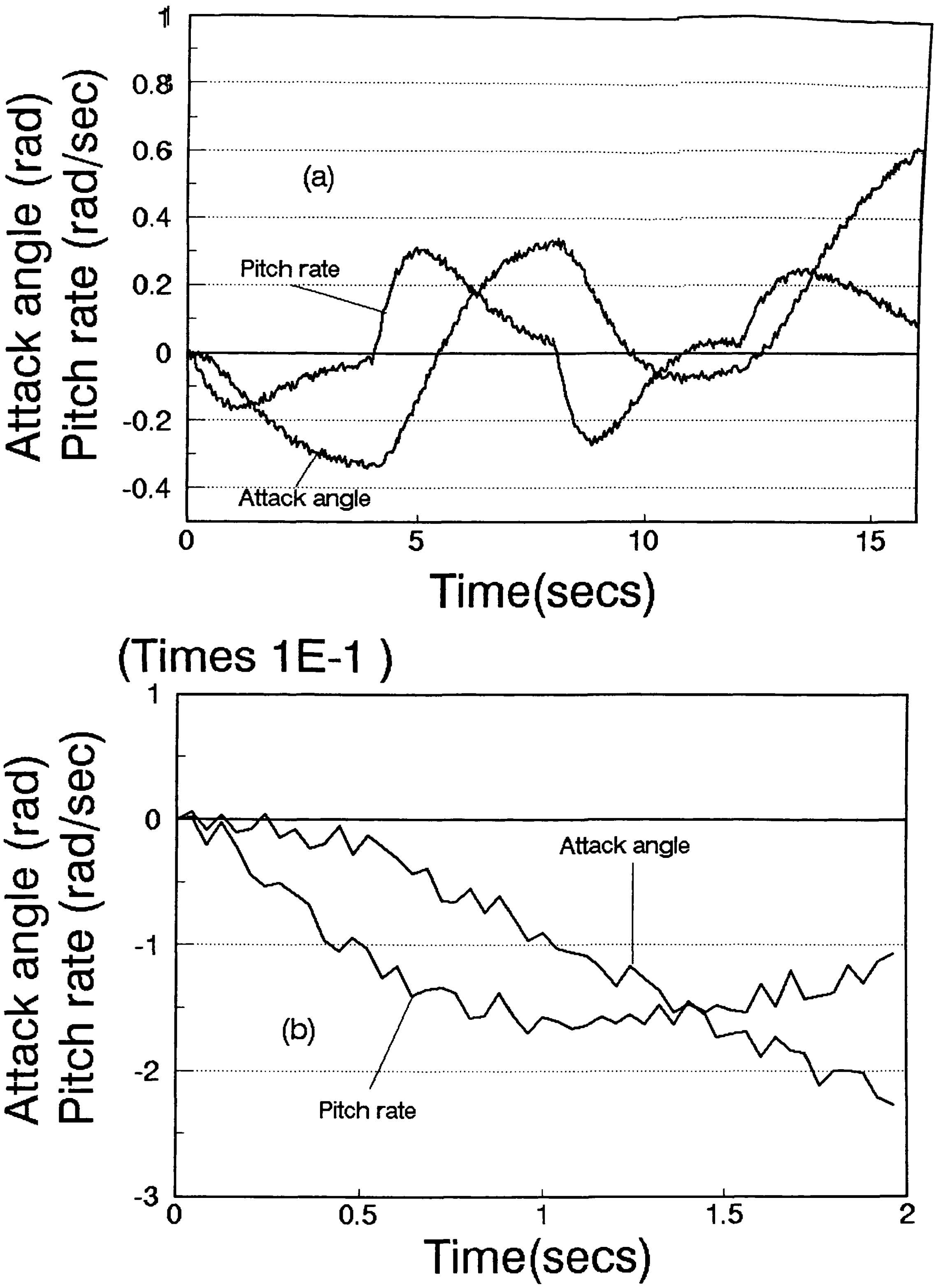


Fig.5.29. Output observation for noise amplitude 0.03. Fig.b. is the enlarging of Fig.a.

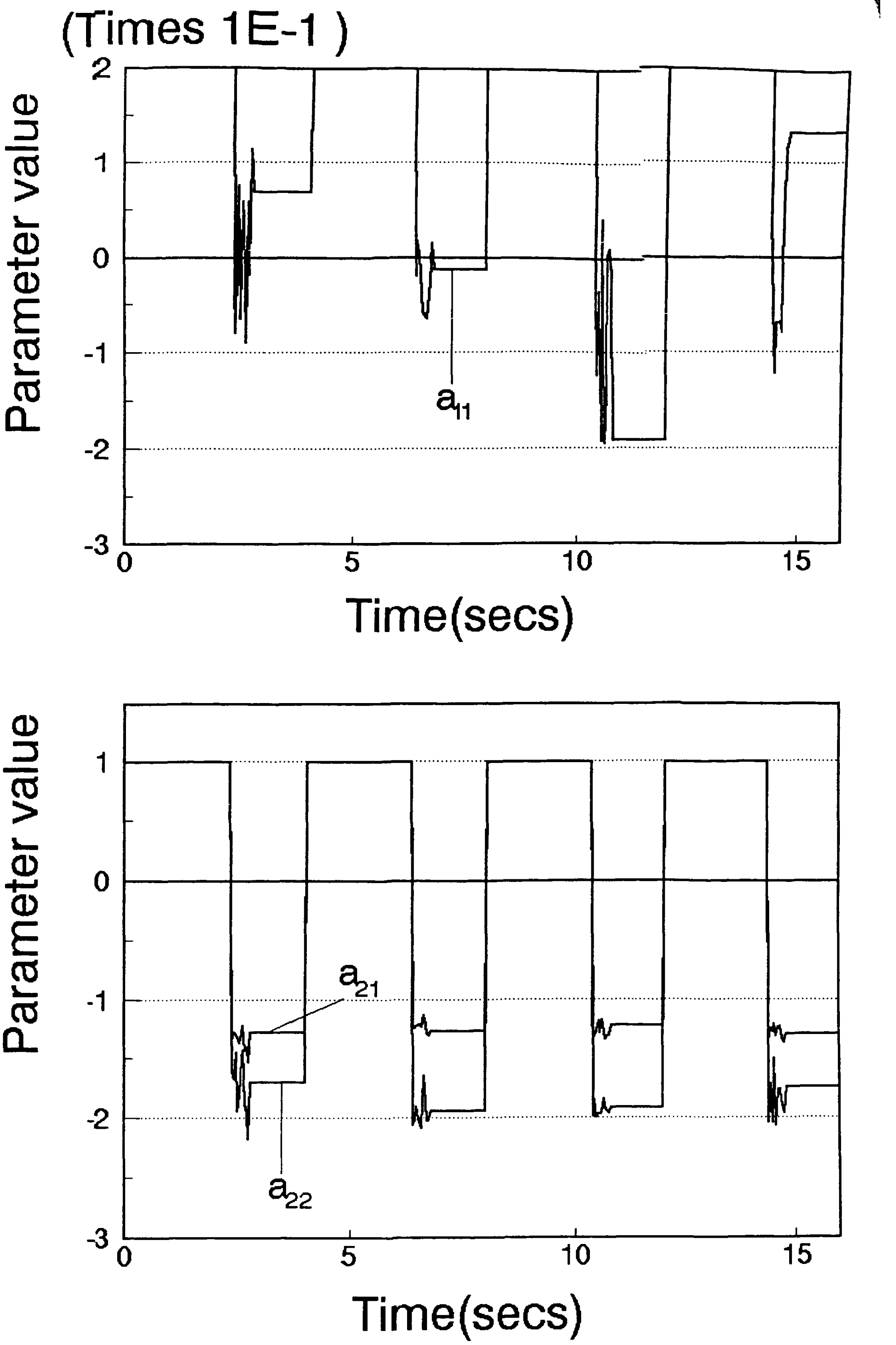


Fig.5.30. The identification results for noise amplitude 0.03 with 60 step time-averaging method.

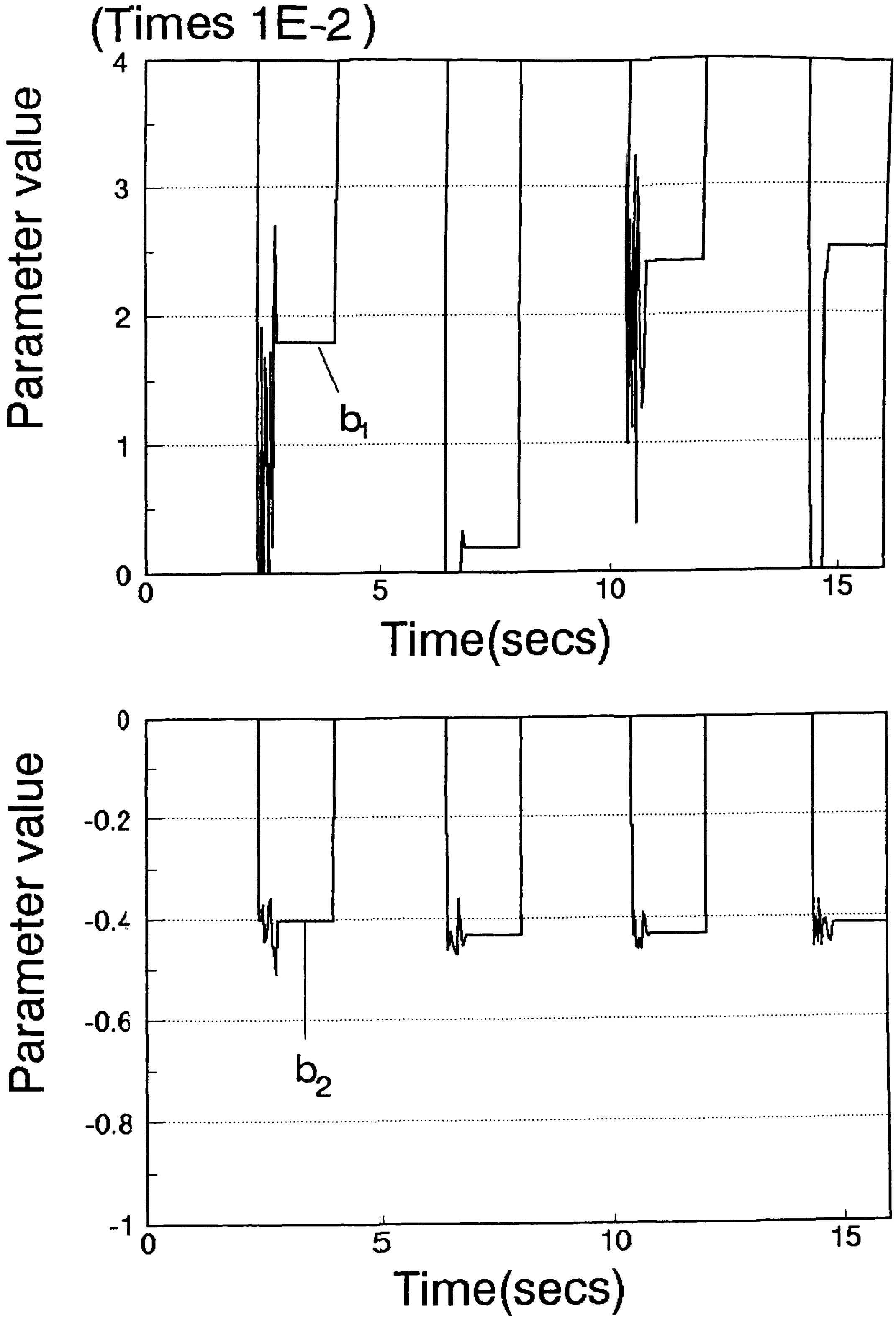
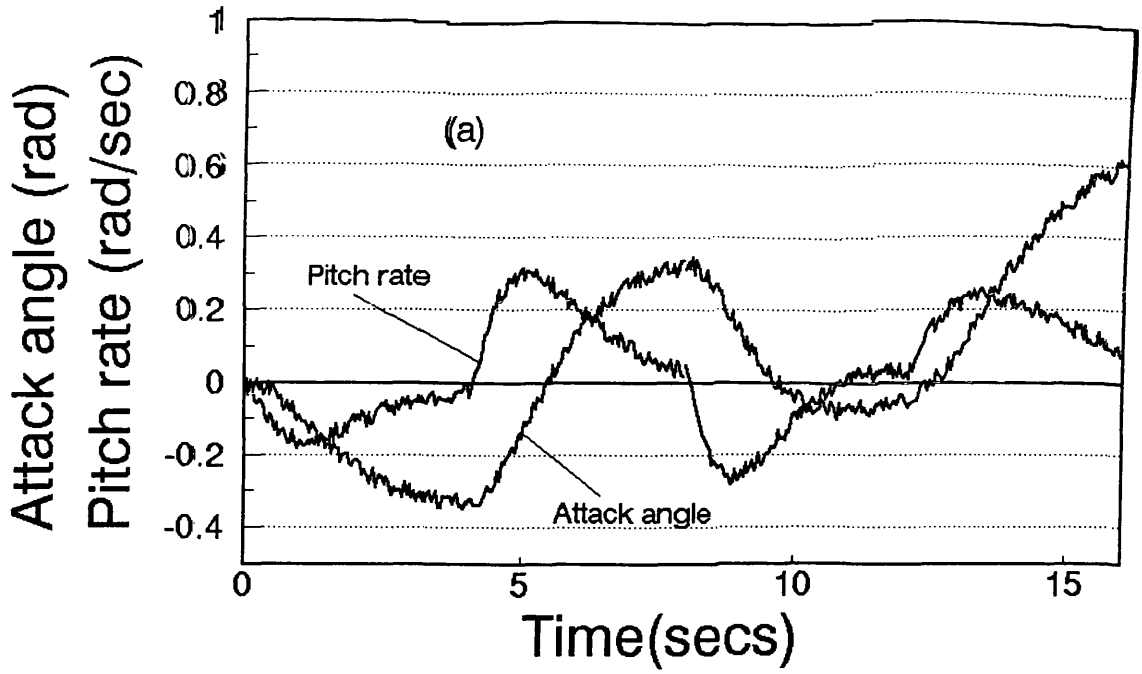


Fig.5.31. The control parameter identification for noise amplitude 0.03 with 60 step data.



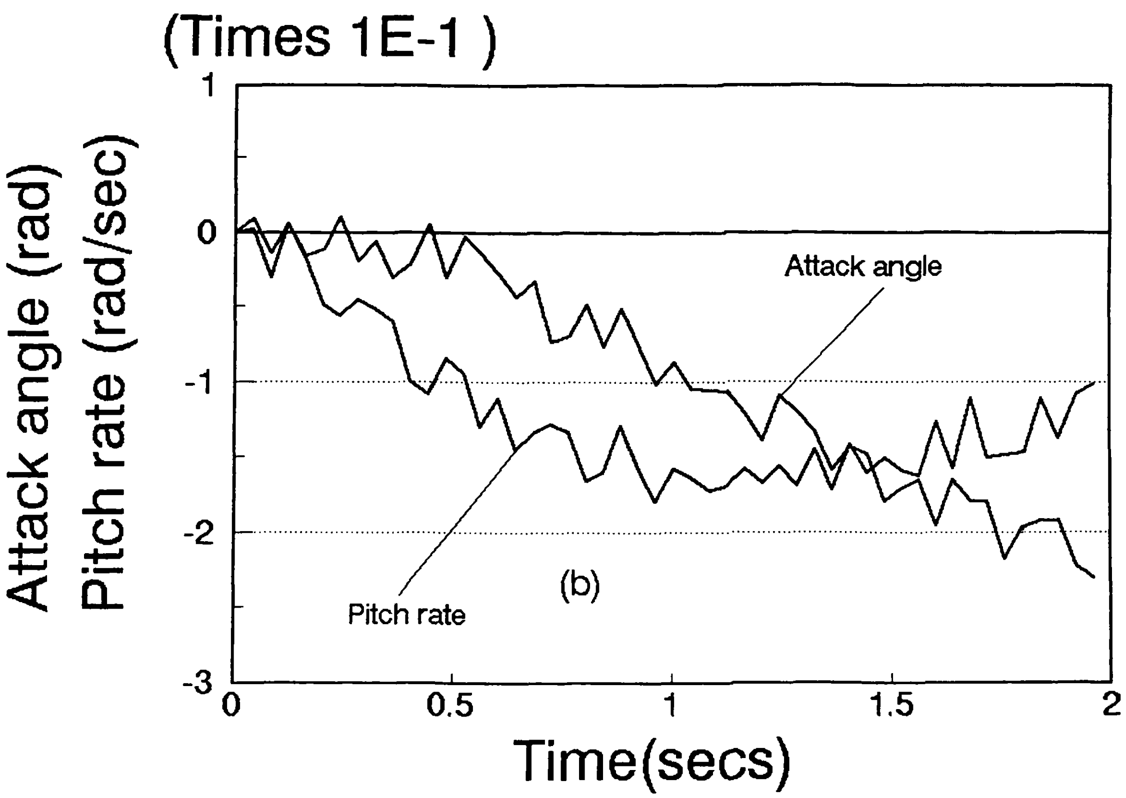


Fig.5.32. Output observations for noise amplitude 0.05

Fig.b. is the enlarging wiev of Fig.a.

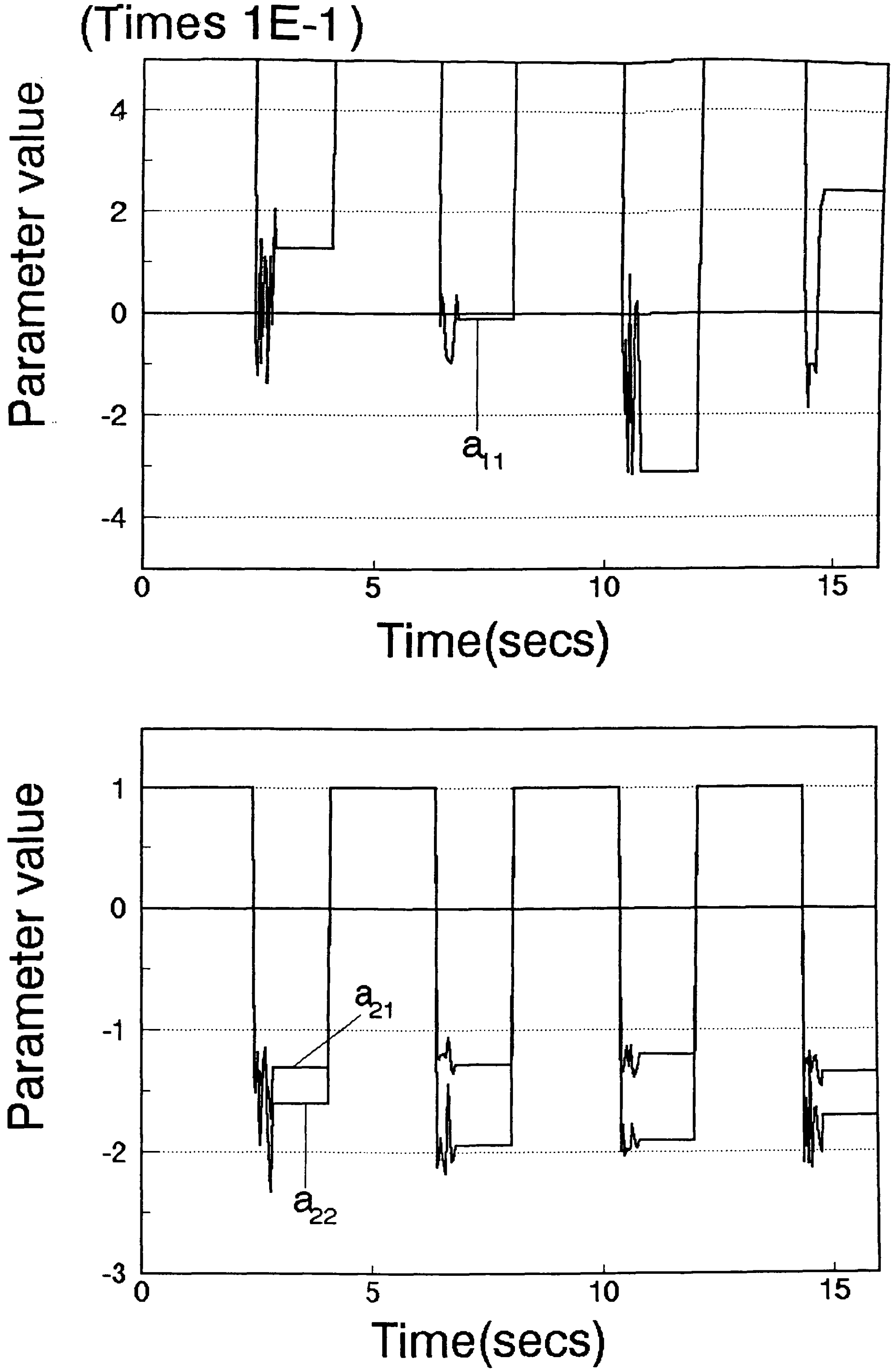


Fig.5.33. The idenfication of the system for noise 0.05 with 60 step block data.

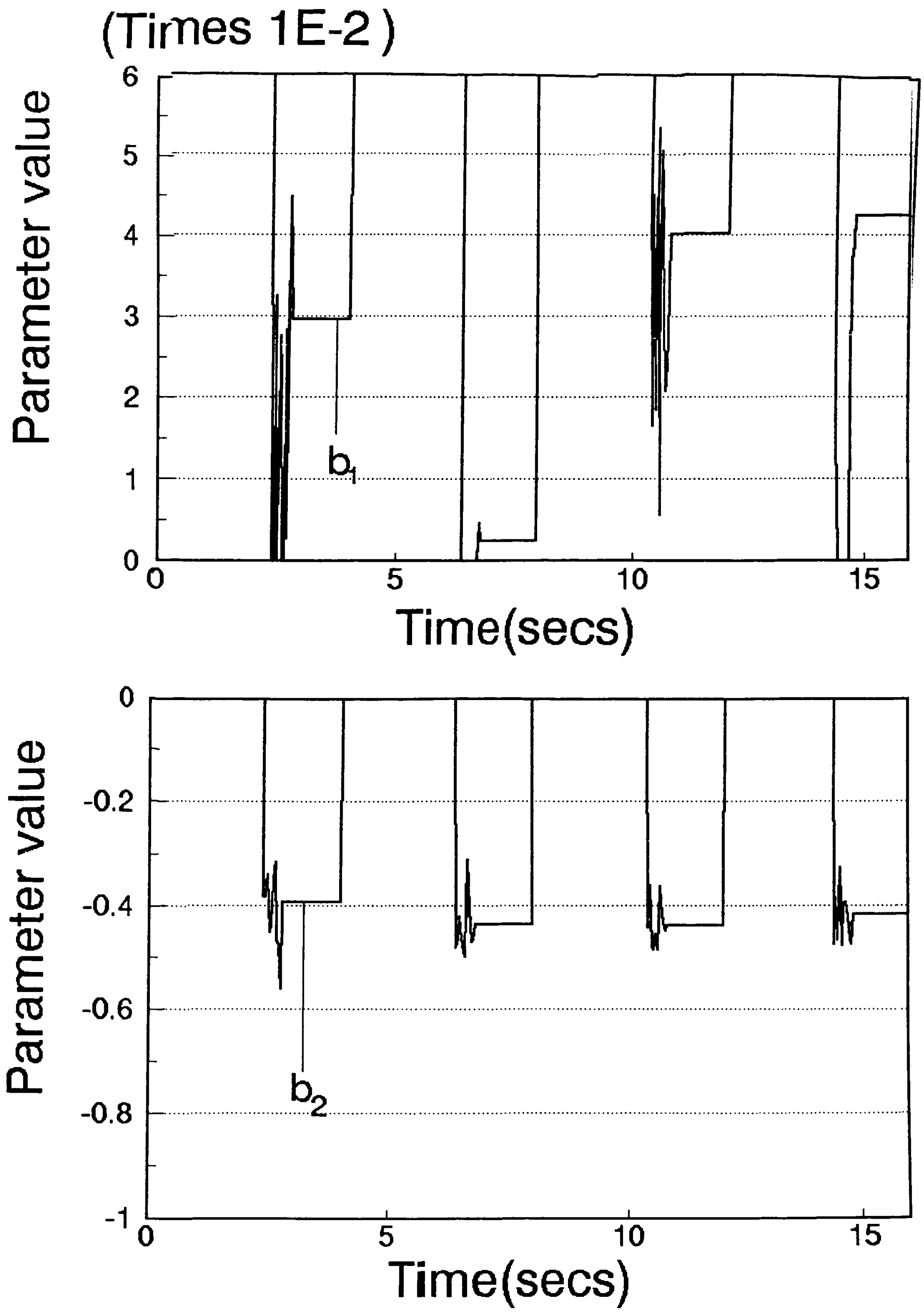
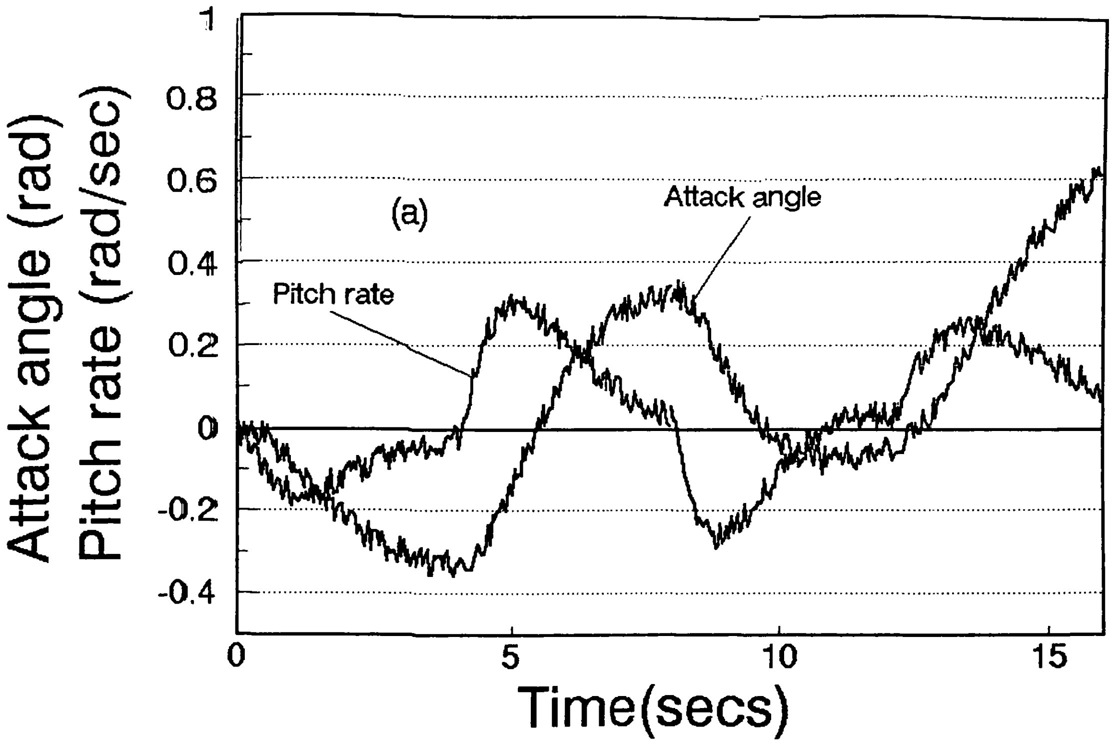


Fig.5.34. The identification of the control parameters for noise 0.05

60 step block data. Error is still ignorant.



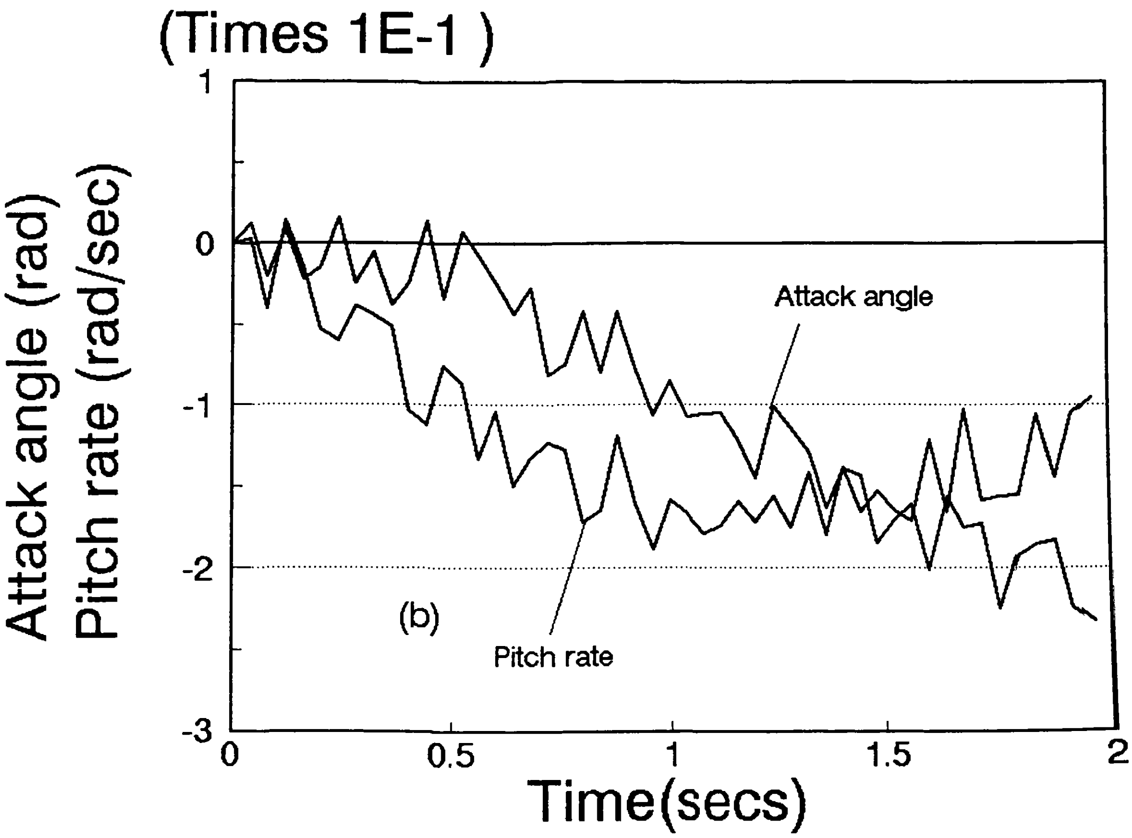


Fig.5.35. Outputs observations for noise amplitude 0.07.

Fig.b is the enlarging view of Fig.a.

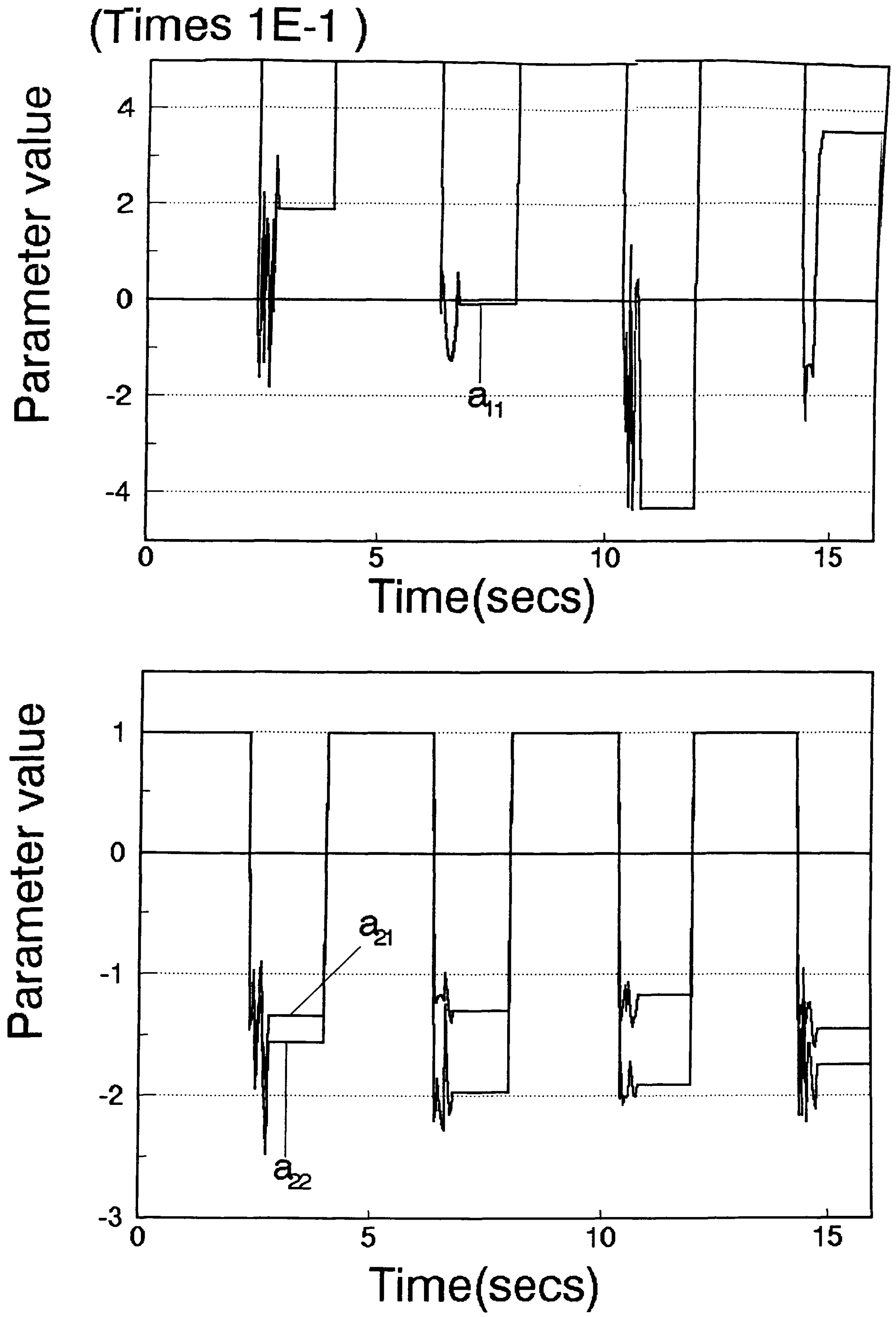
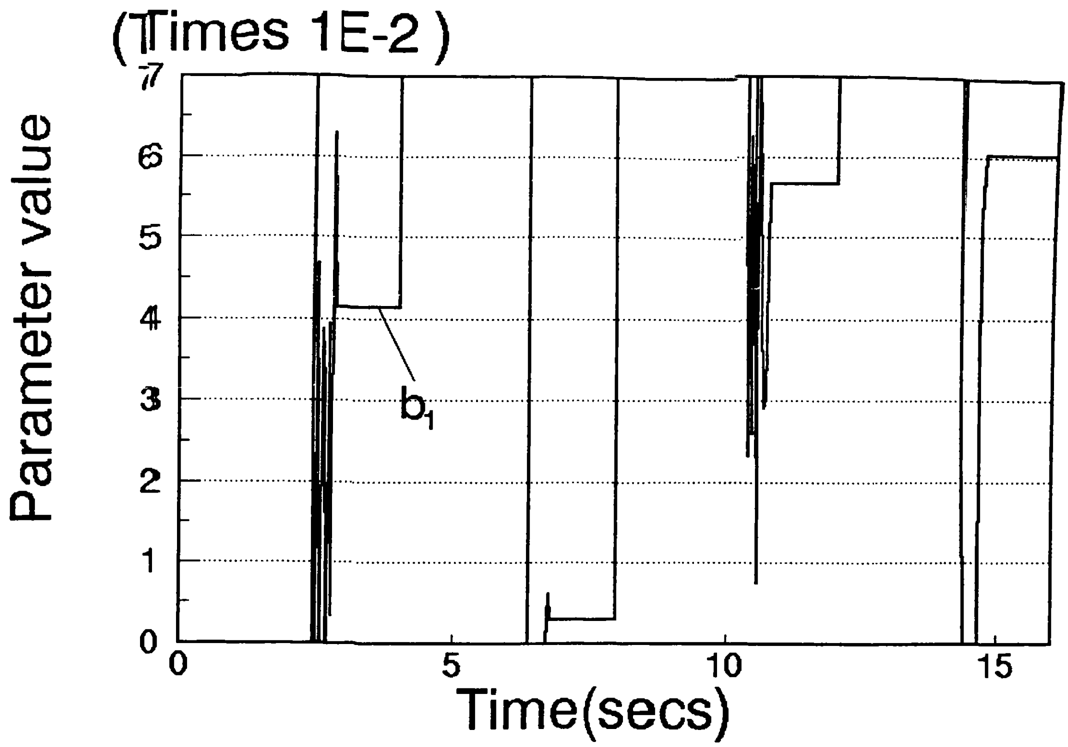


Fig.5.36. The identification of the system parameter with 60 step block data. Error incerases on the dominant parameters, but it is still around 10% . Noise amplitude is 0.07 .



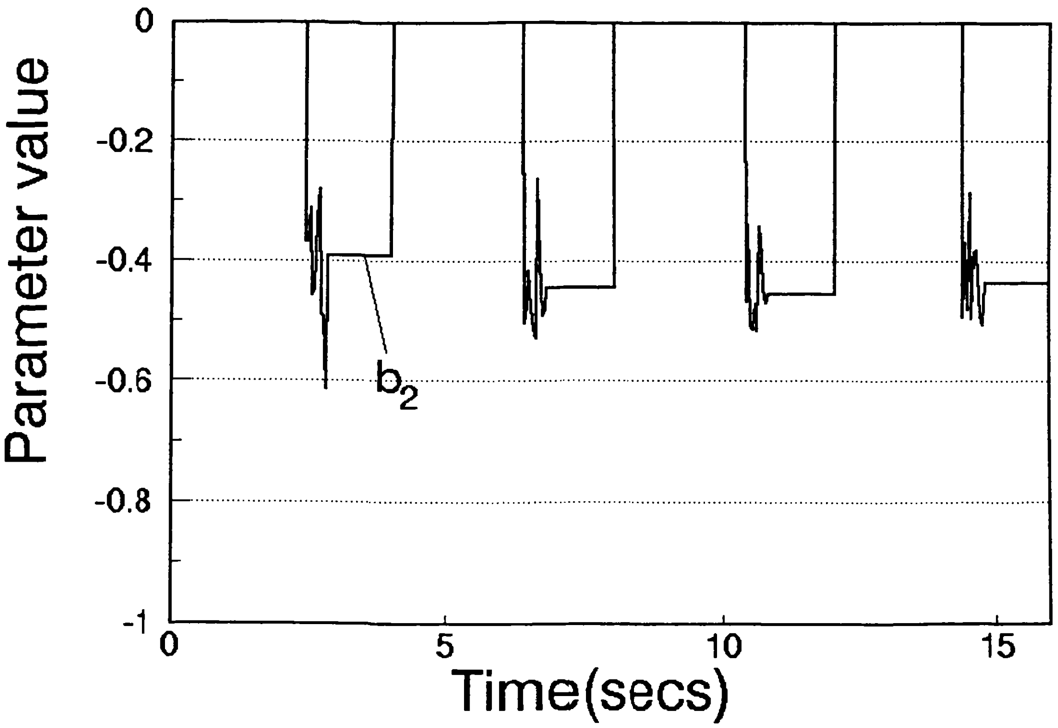
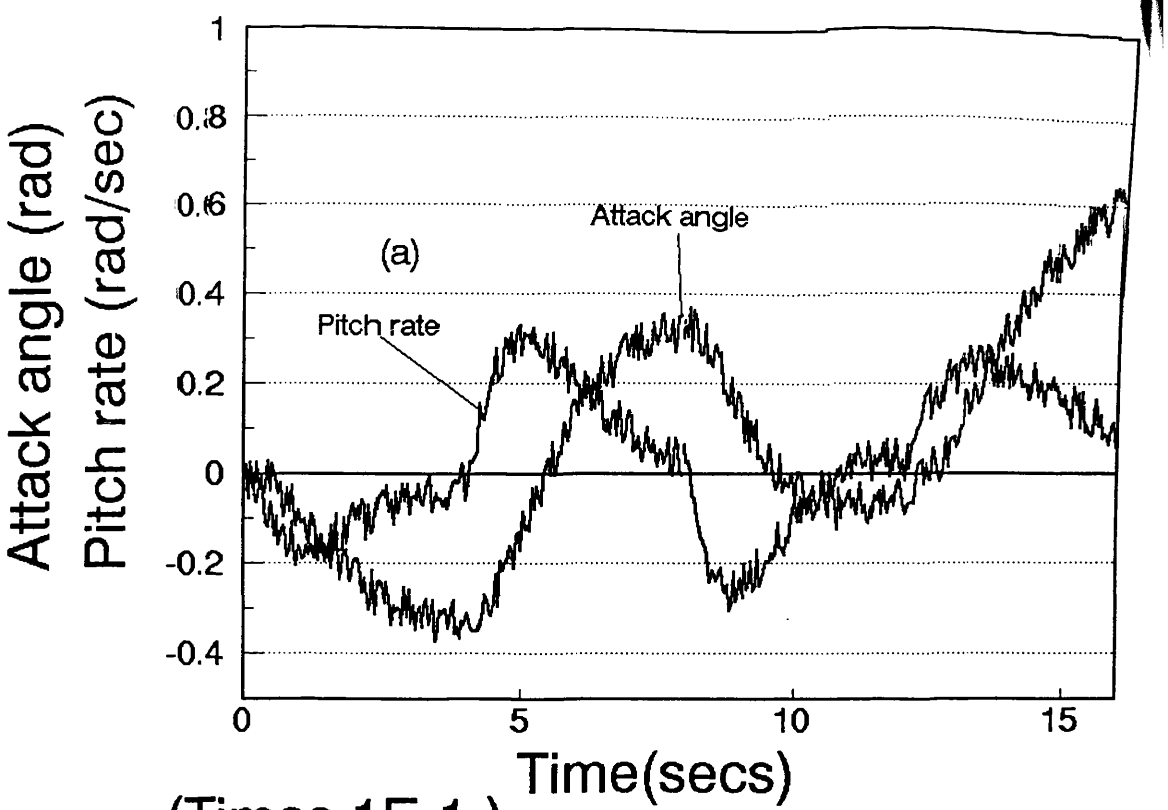


Fig.5.37. The identification of the control parameters for noise 0.07

Error of b_2 is still less than 10% .



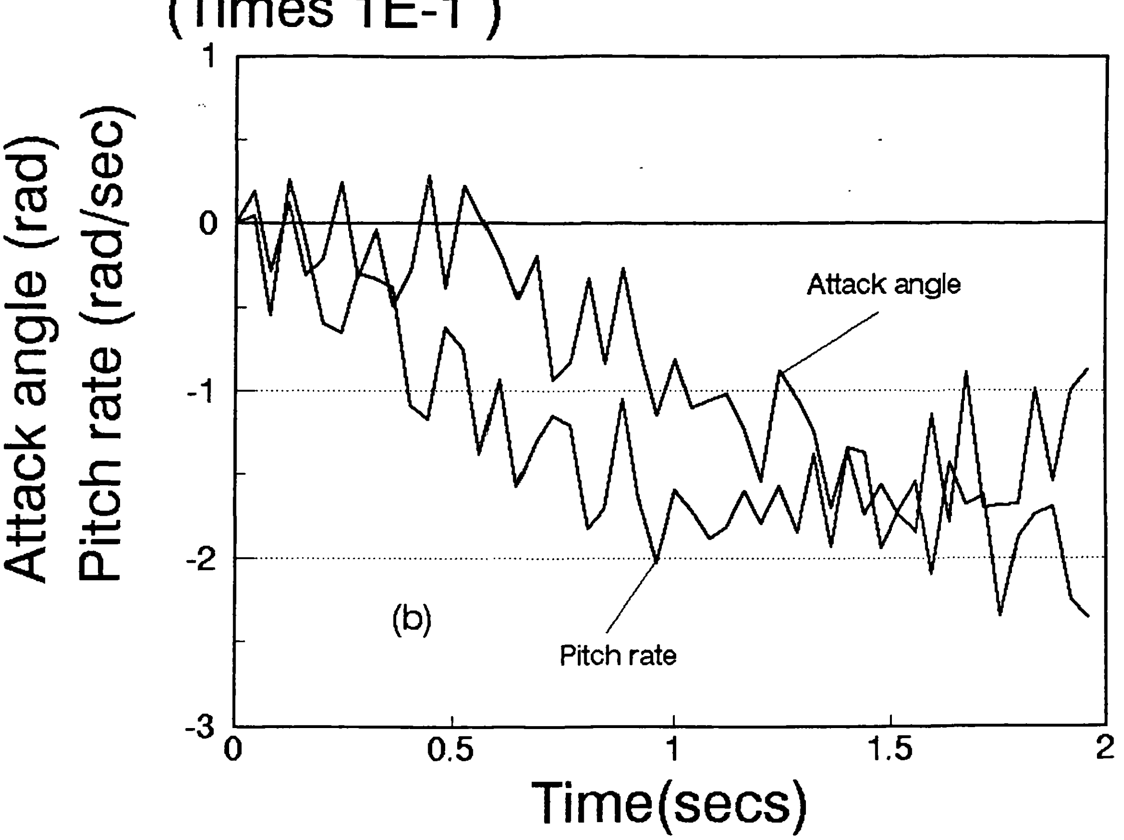


Fig.5.38. Output observations for noise amplitude 0.1 .

Fig.b is the enlarging view of Fig.a.

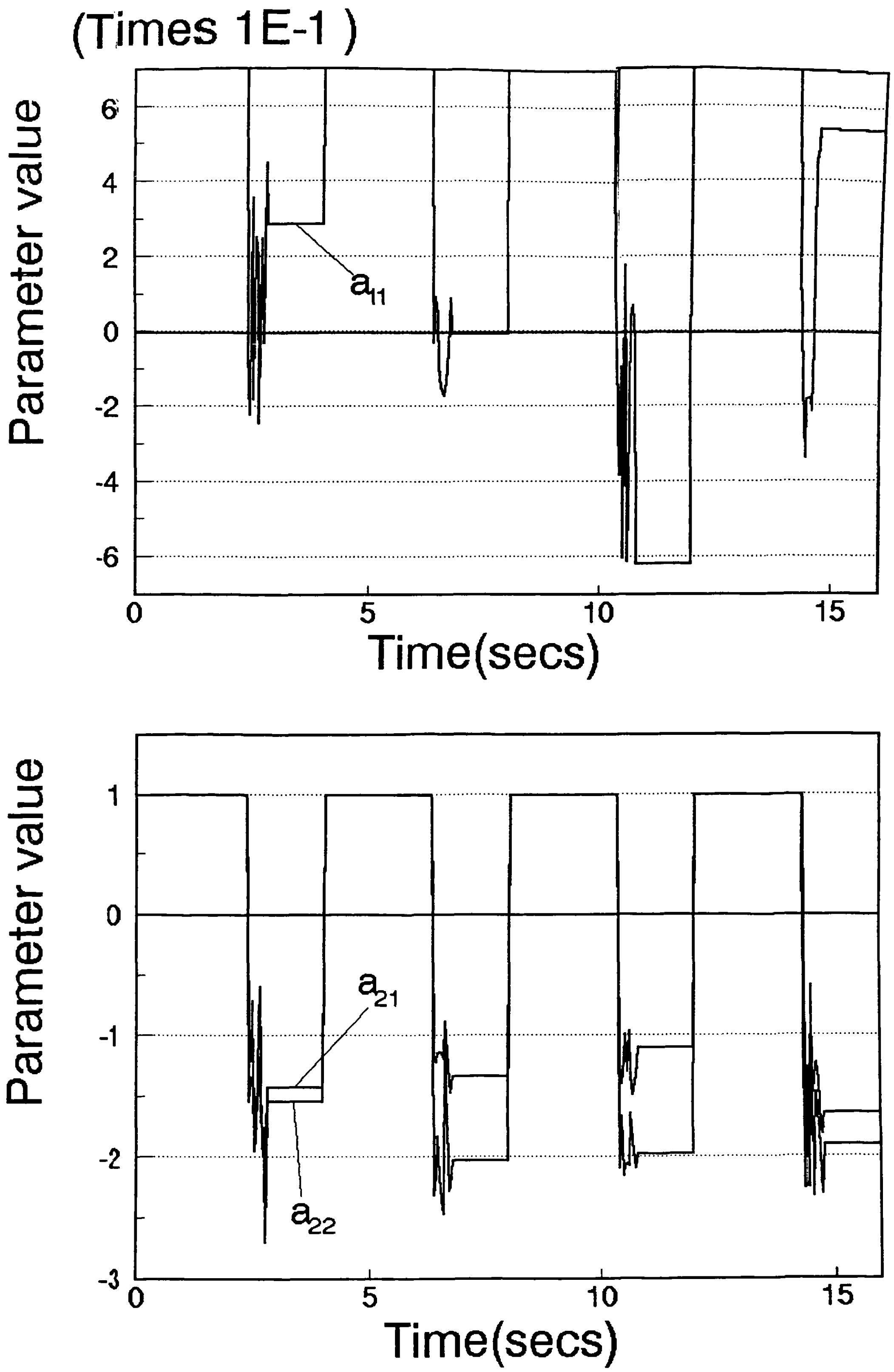


Fig.5.39. The identification of the parameters for noise amplitude 0.1 , the error of dominant parameter is still less than 20% .

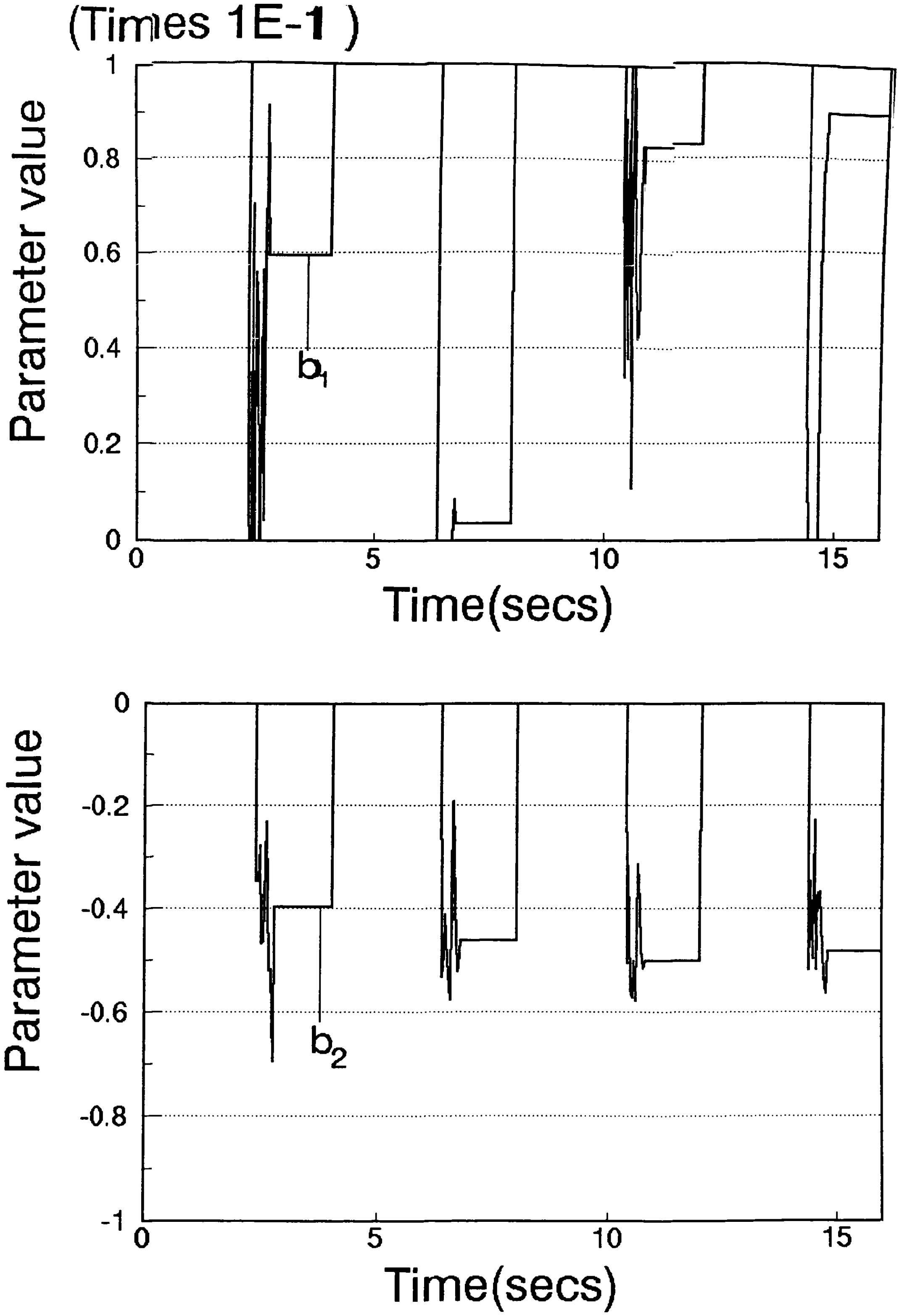


Fig.5.40. The control parameters identification for noise amplitude 0.1 , the error of b_2 is still less 10% .

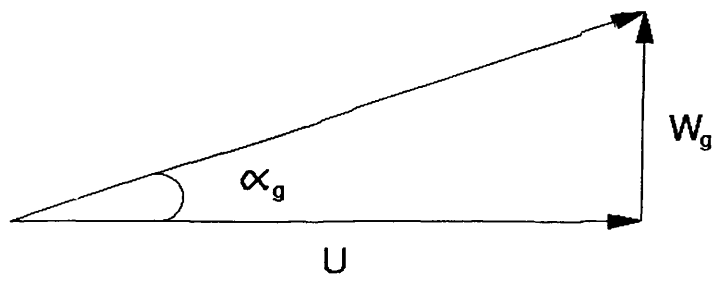


Fig.5.41. The attack angle definition

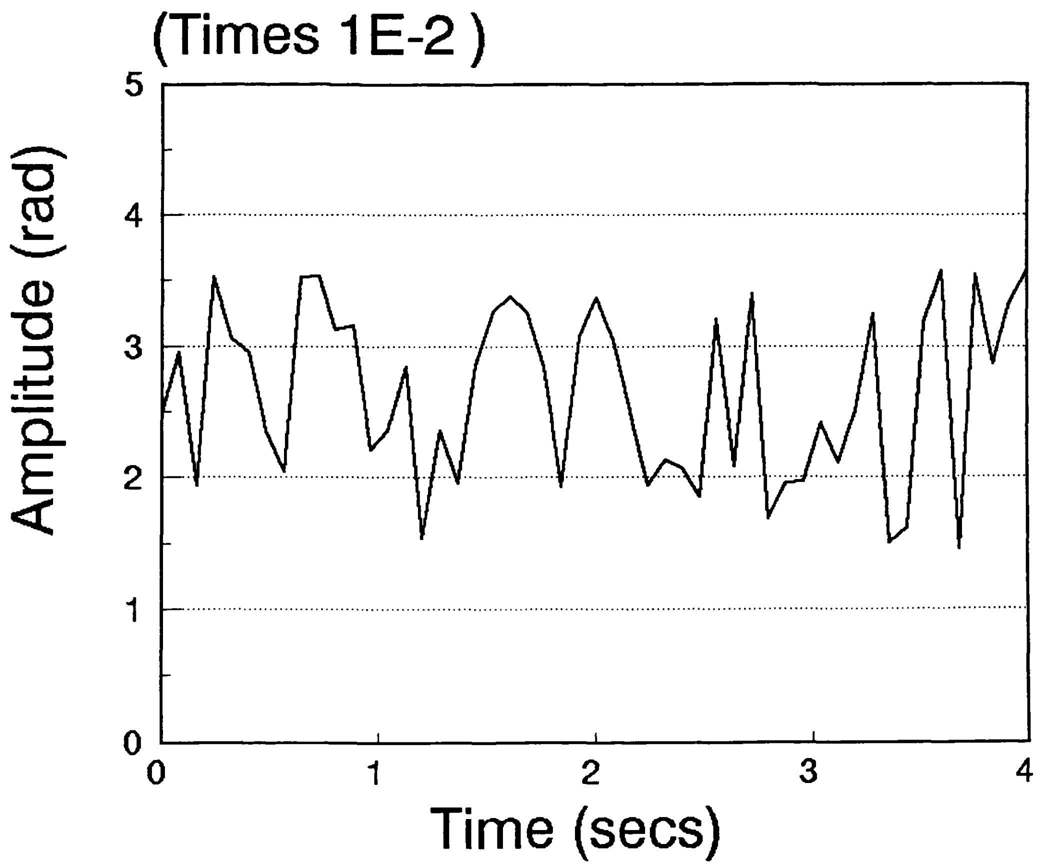


Fig.5.42. The gust changing around the intensities.

The figures 5.43 to 5.66 show the effect of air turbulence on the system identification. The stationary air velocity, which is represented by a constant α_g in the figures, has no effect on these system parameters which are coefficients of the variables, it affects only the control parameters. Random turbulence effects all parameters. Noise effect is the same with the normal identification. Different turbulence intensities effects, which are given in Table 5.1 , have been shown in the figures 5.43 to 5.66 . Each figure has denoted a different air velocity.

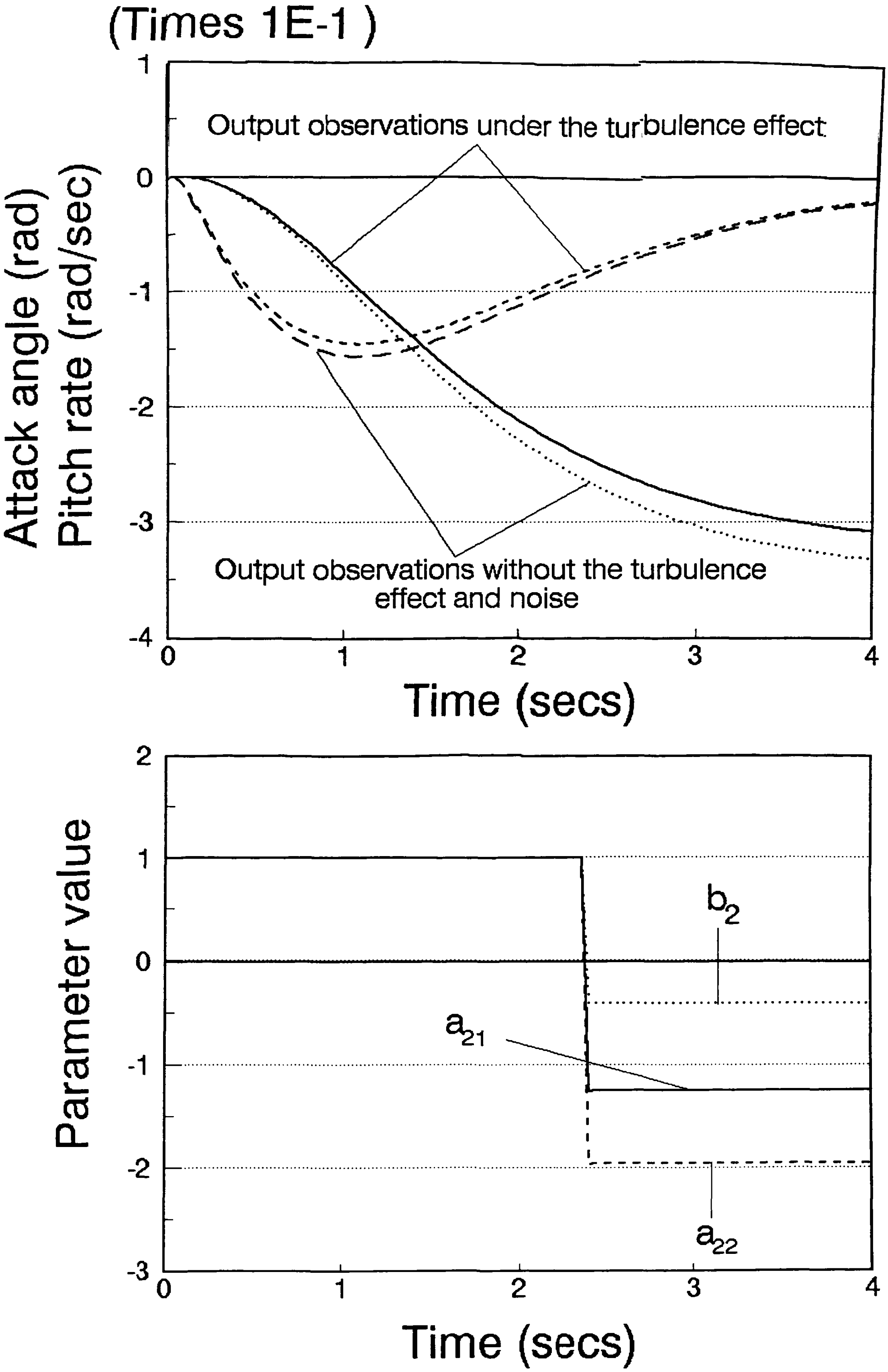
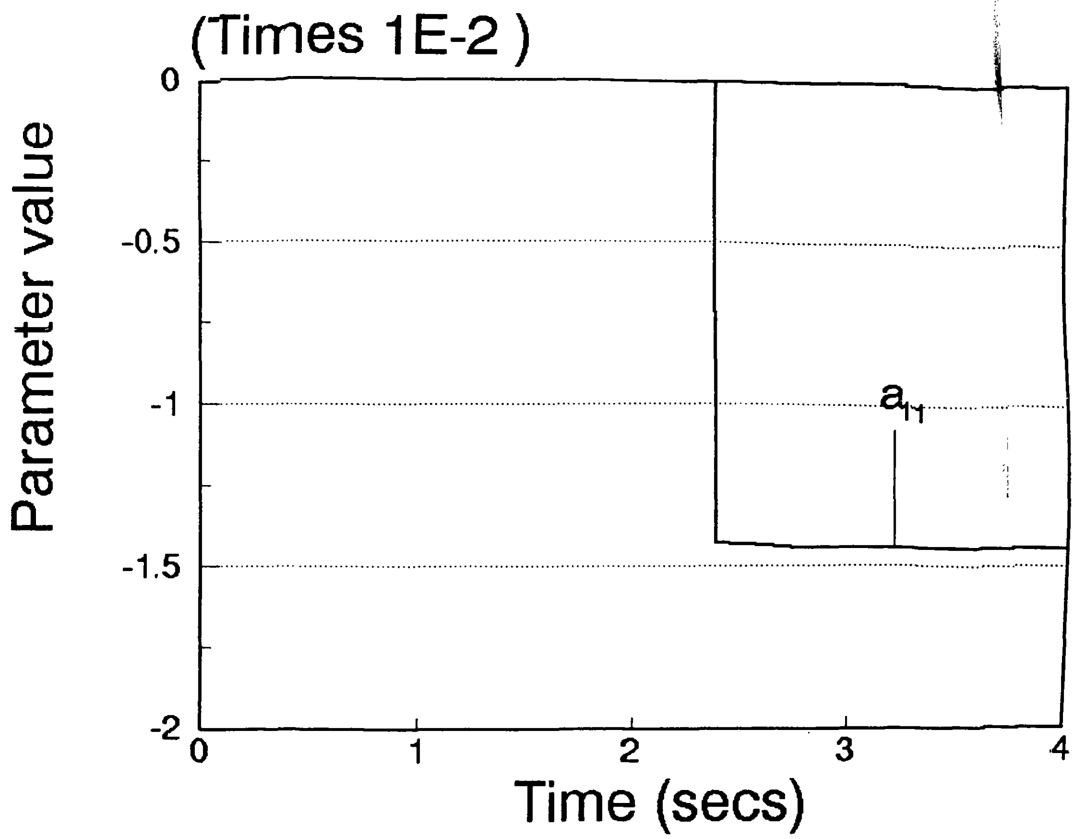


Fig.5.43. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.025$



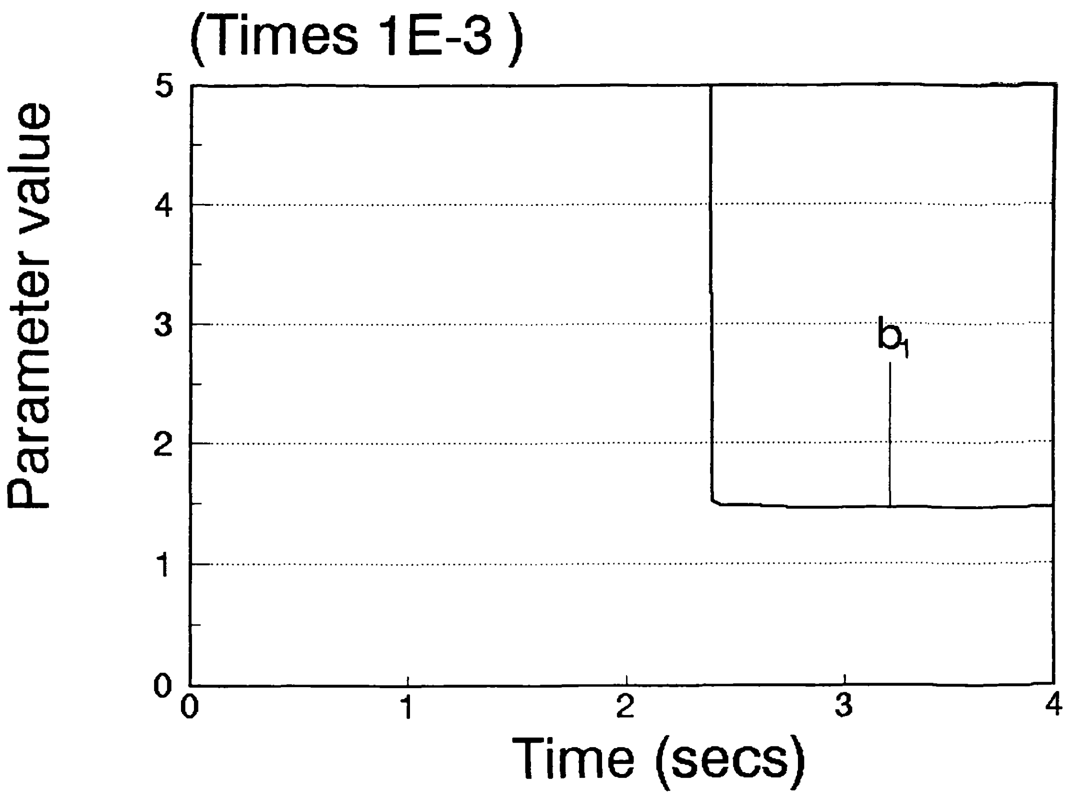


Fig.5.44. The identification results with Fig.5.43. data

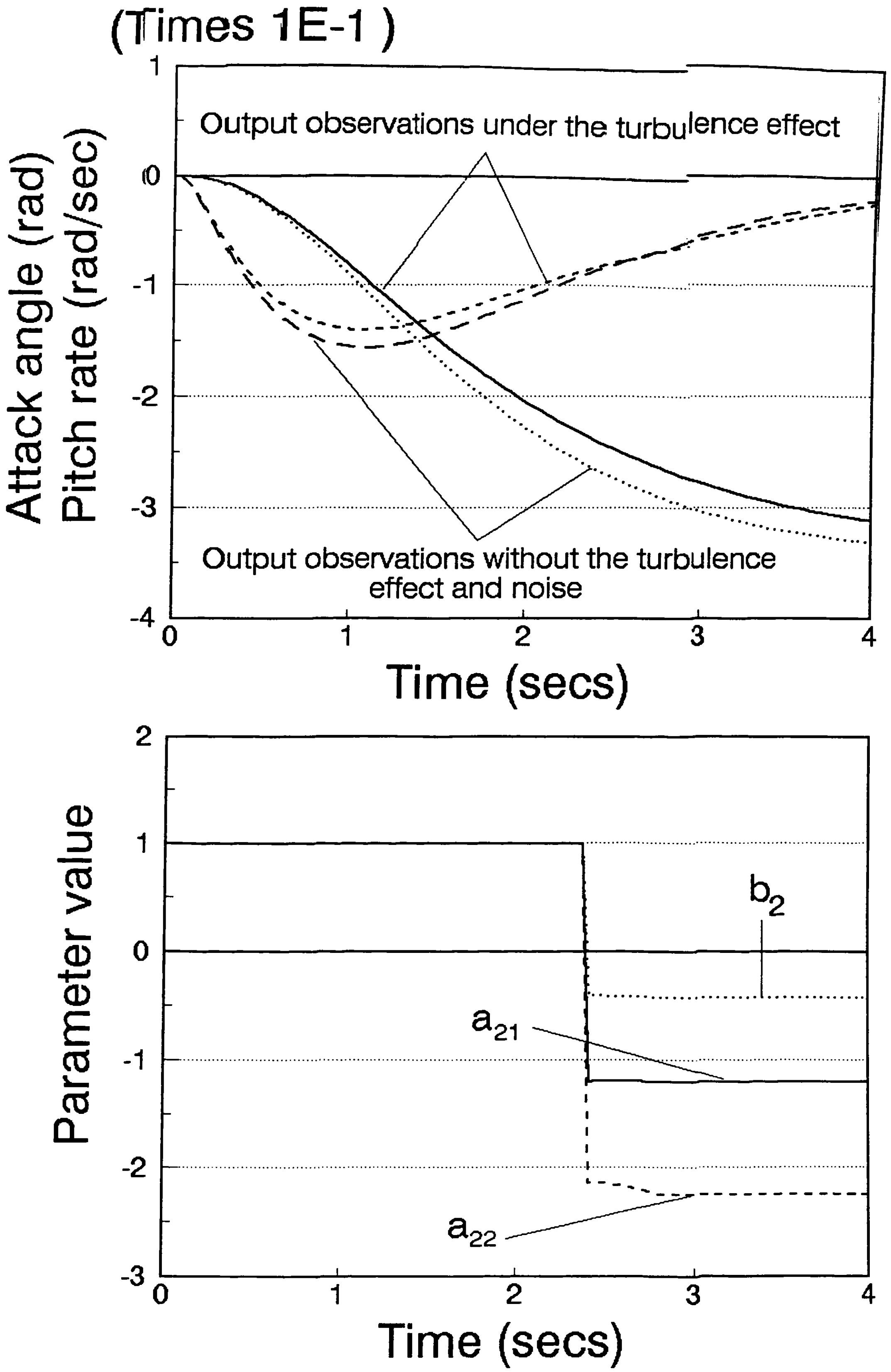


Fig.5.45. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.025*(1 + 0.5*\text{random func.})$

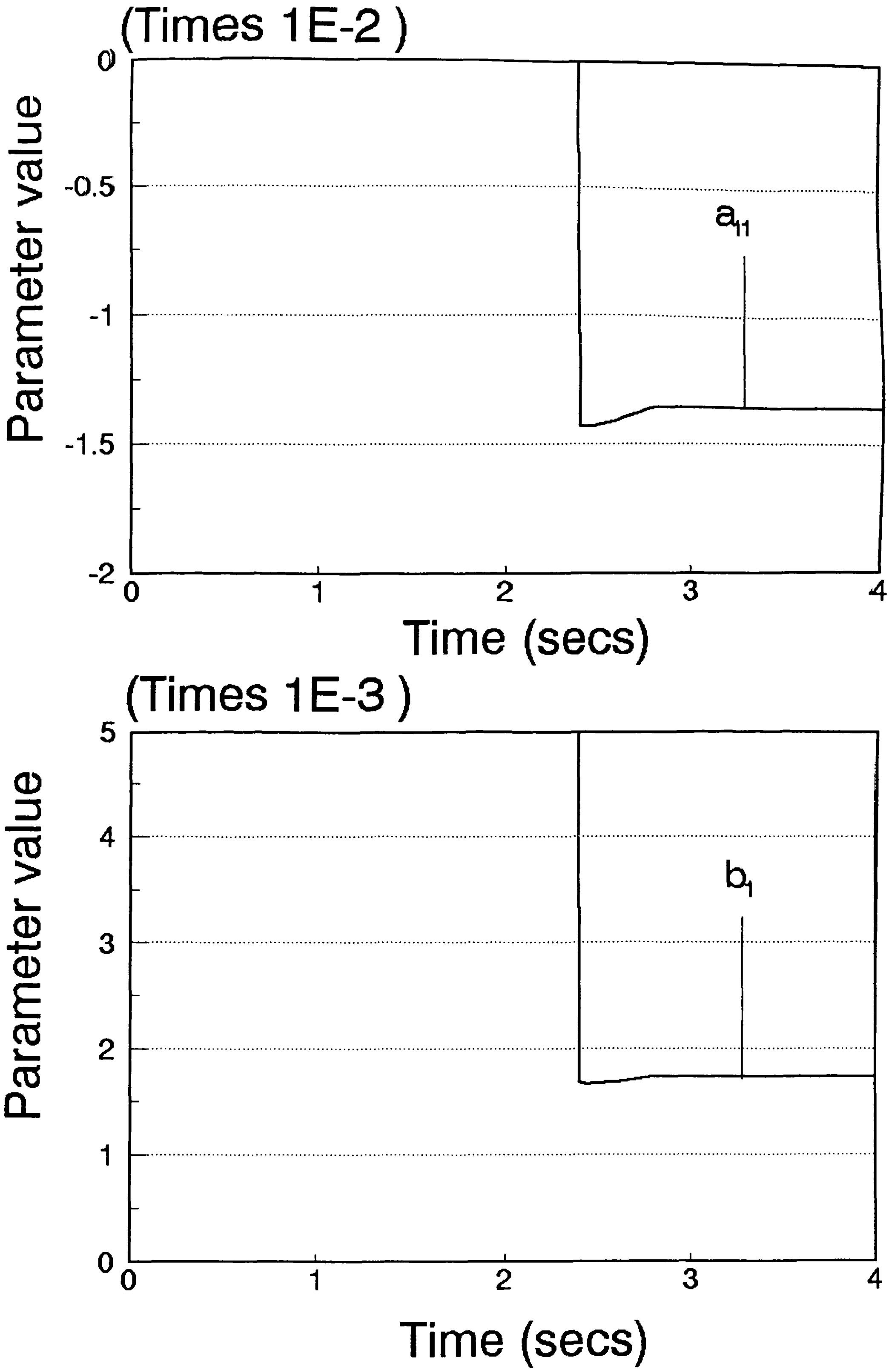


Fig.5.46. The identification results with the observation of Fig.5.43

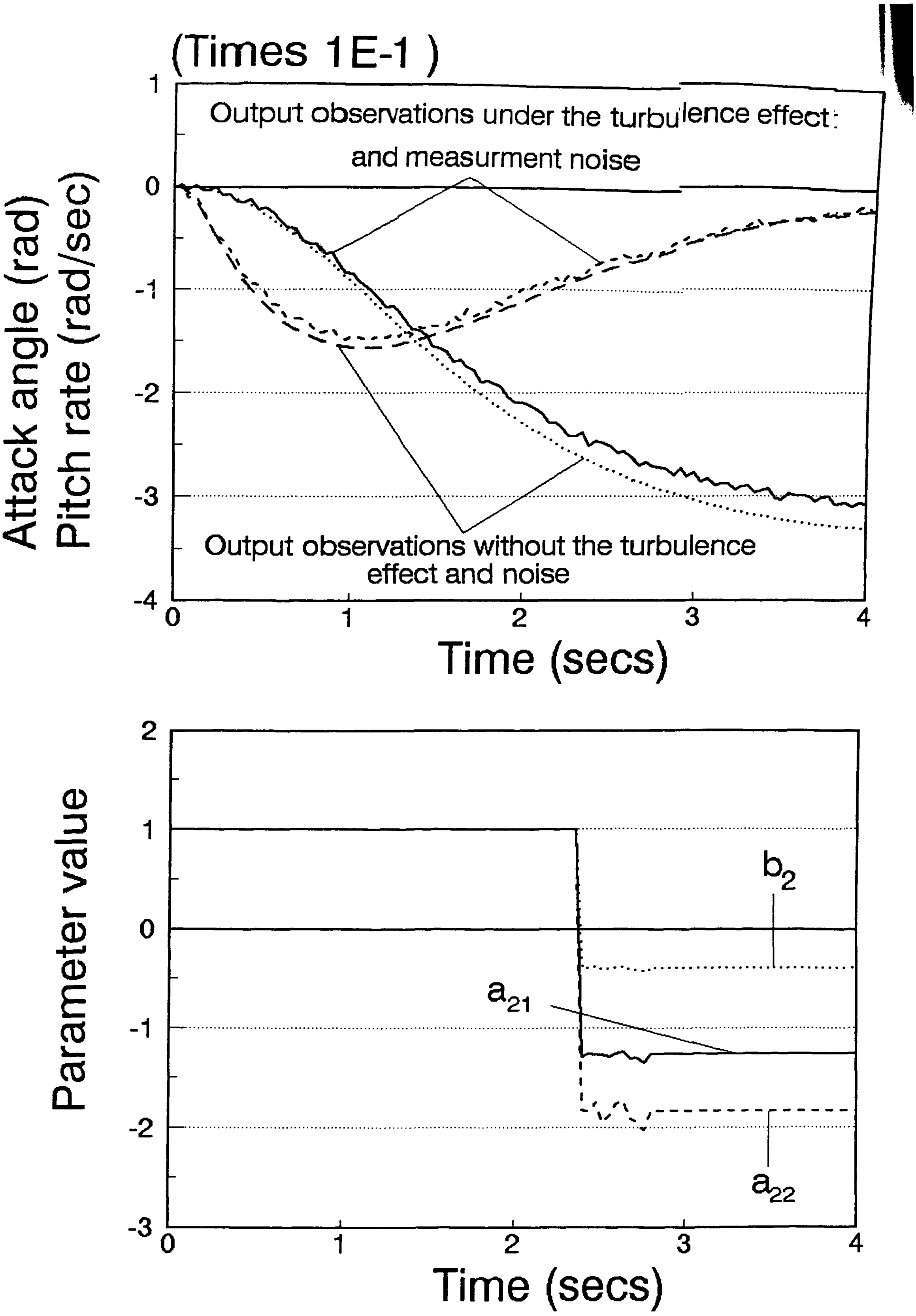


Fig.5.47. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.025$ and observations are $s_i = x_i + 0.01$*random func.

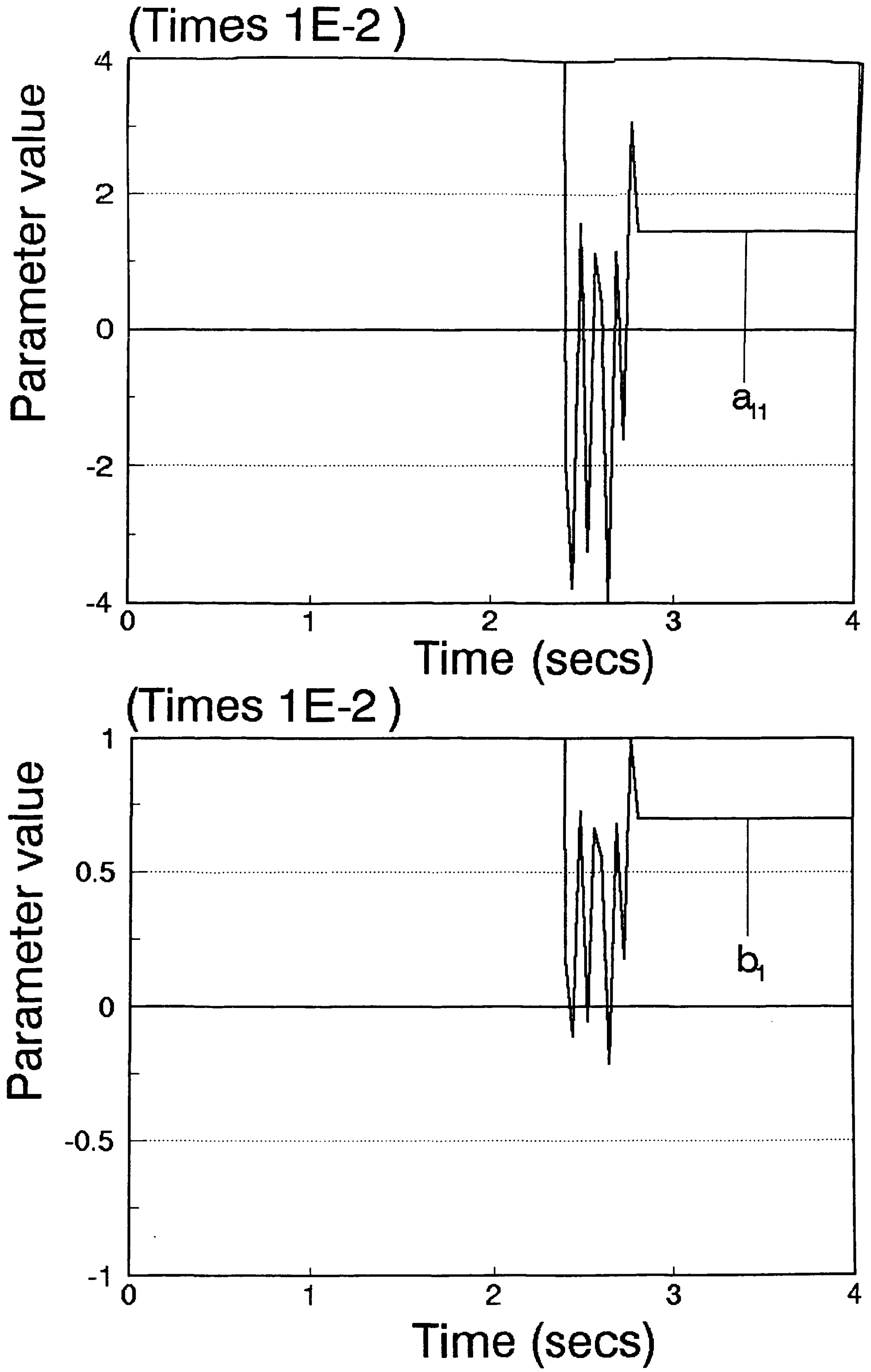
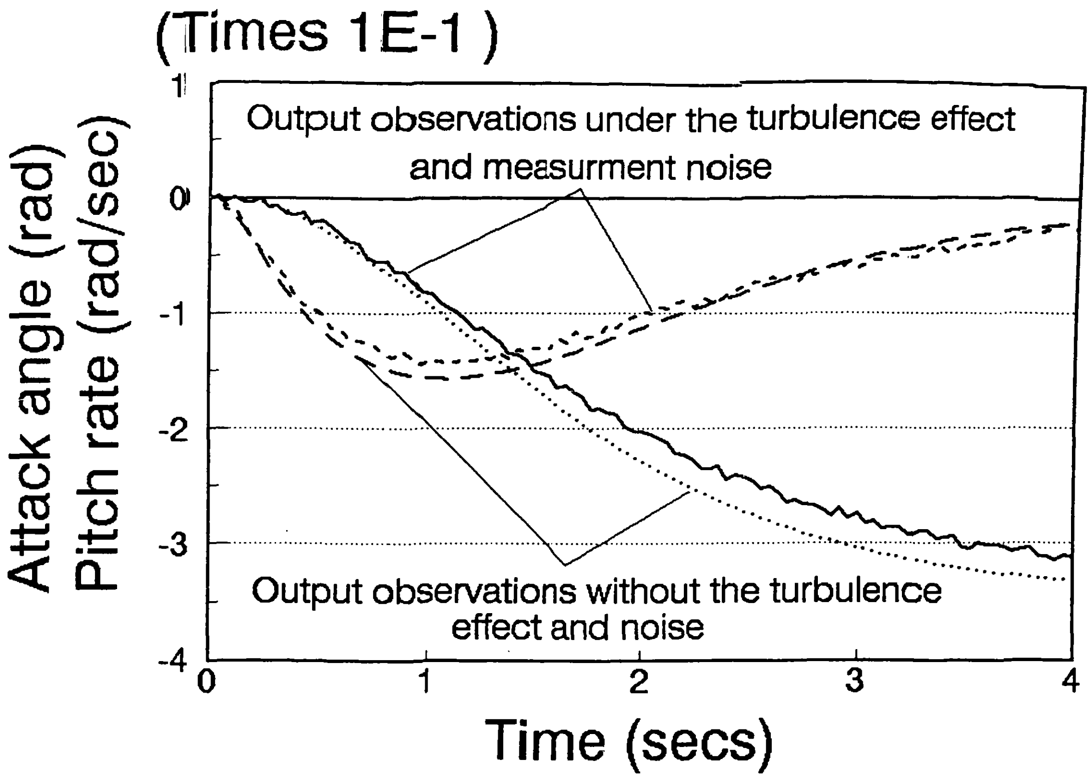


Fig.5.48. The identification results with the observation of Fig.5.43

Both of them are changed with noise and turbulence effect.



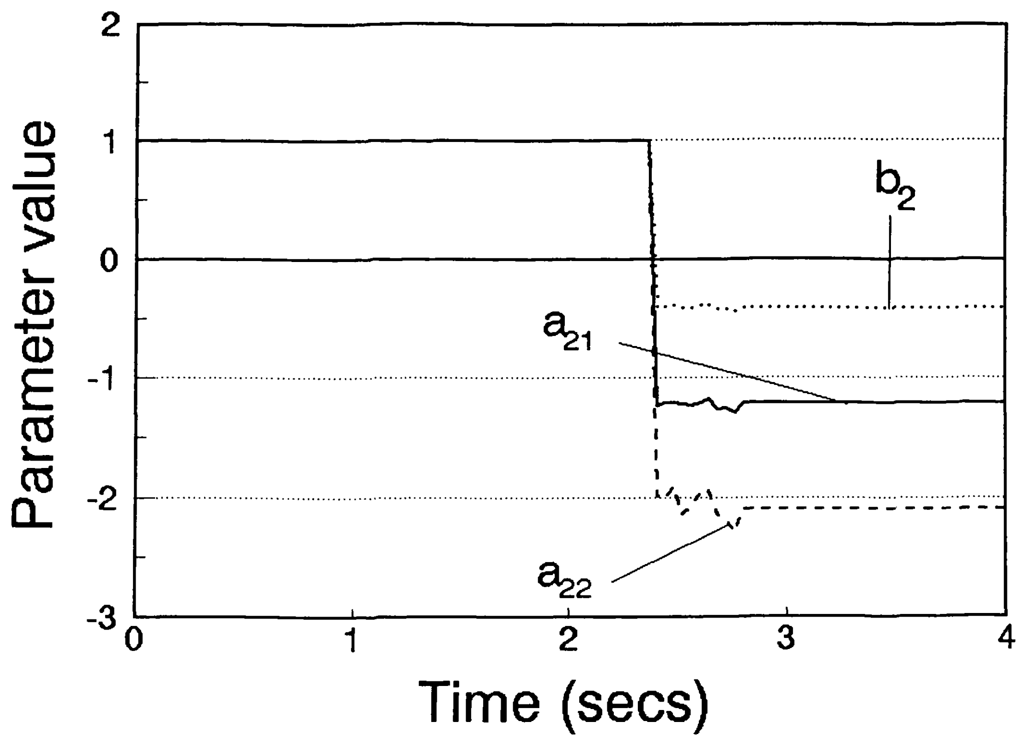


Fig.5.49. The observation and the identification result for atmospheric turbulence effect on the attack angle

$\alpha_g = 0.025*(1 + 0.5*\text{random func.})$

and observations are $s_i = x_i + 0.01*\text{random func.}$

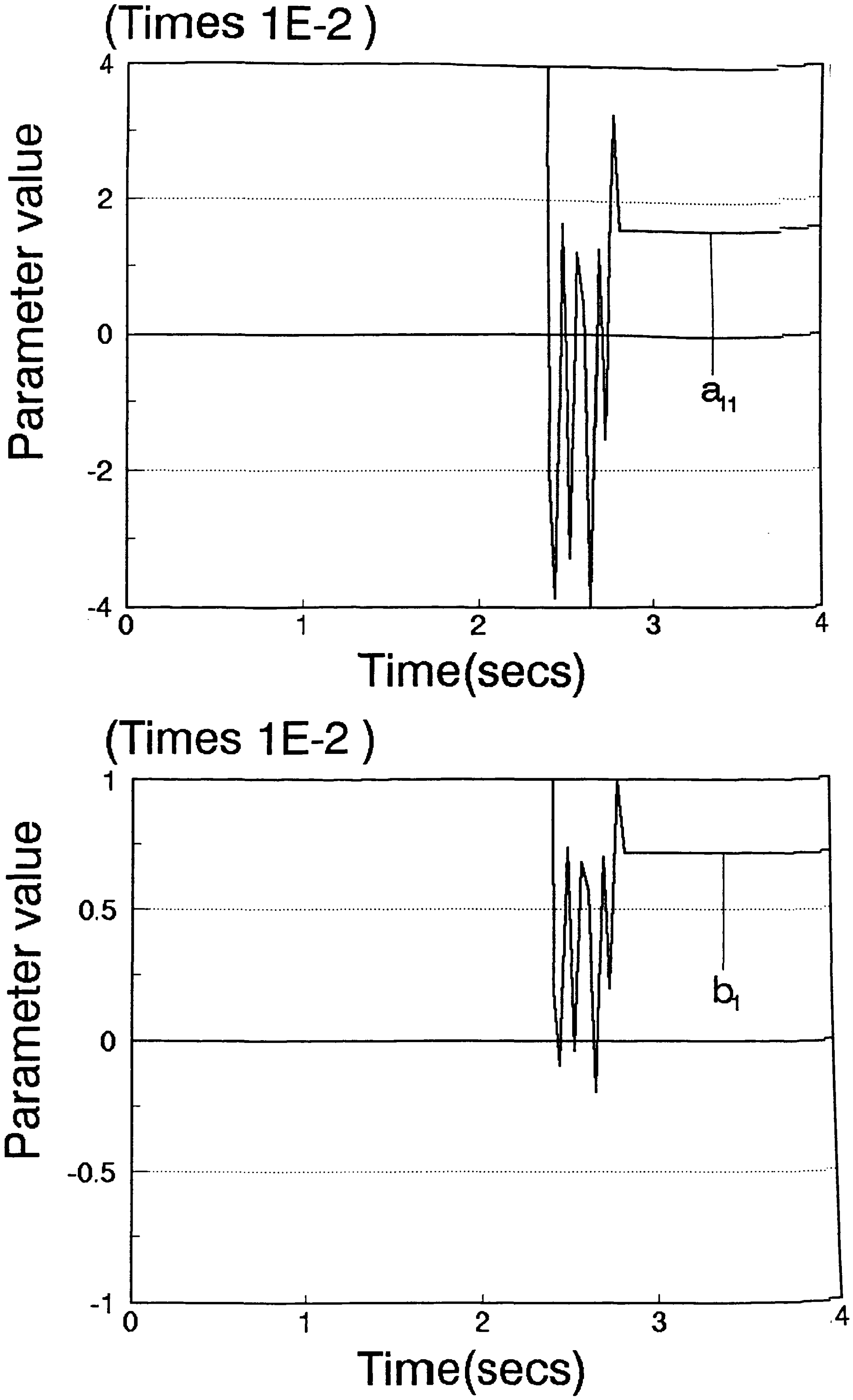
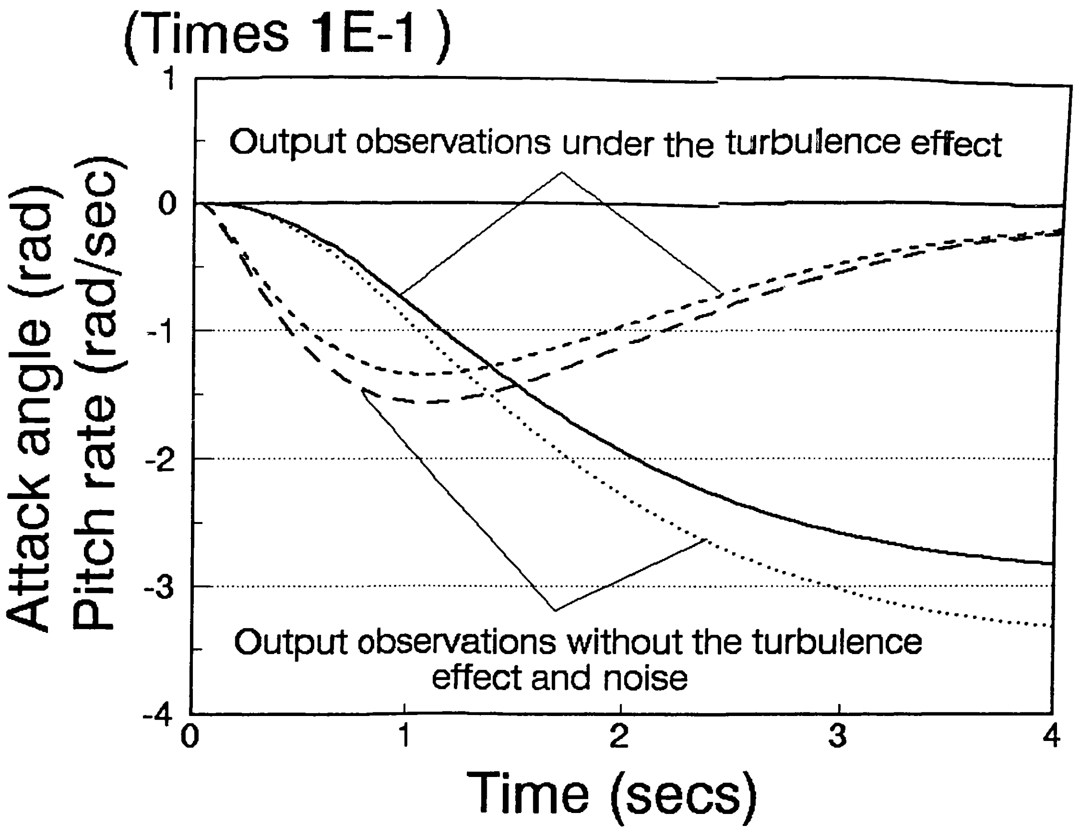


Fig.5.50. The identifivation results with the observation of Fig.5.49



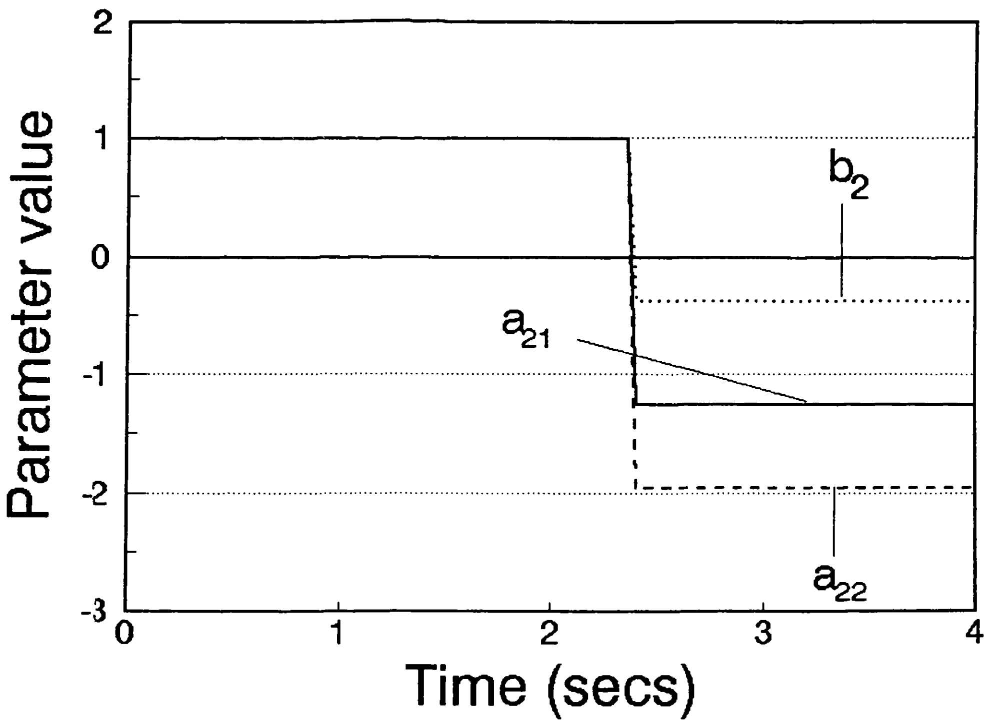


Fig.5.51. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.05$

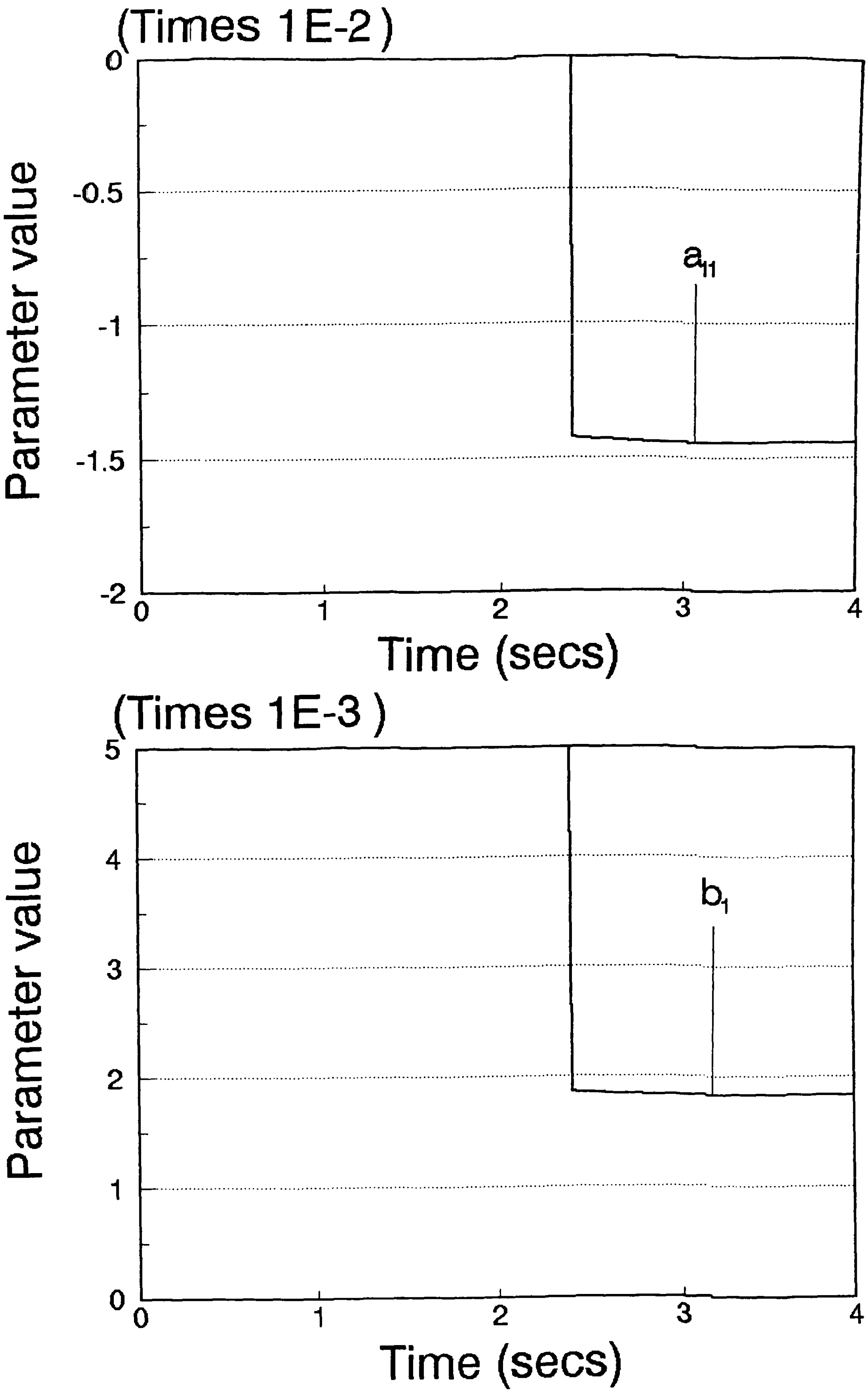


Fig.5.52. The identification results with the observation of Fig.5.51

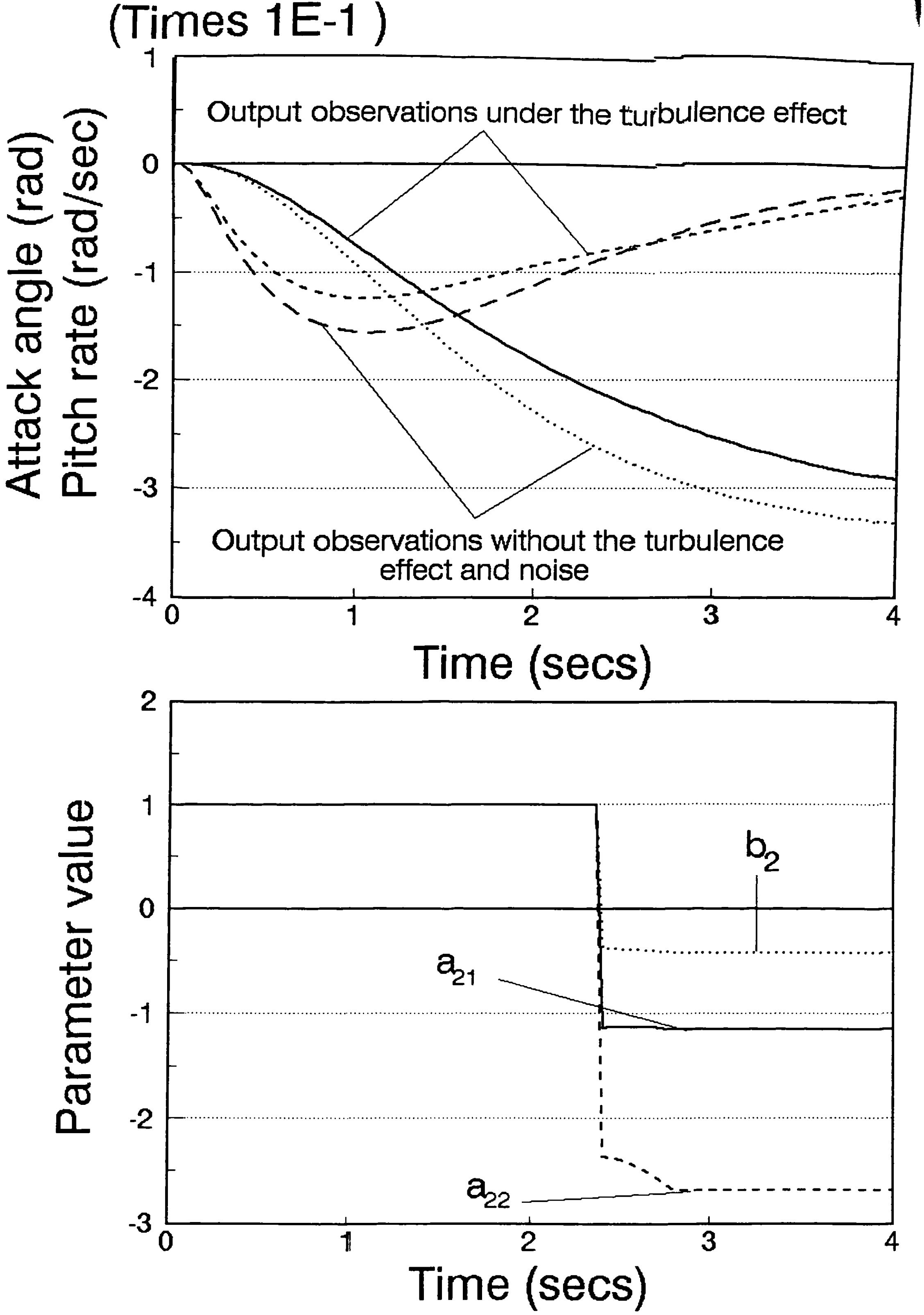


Fig.5.53. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.05*(1 + 0.5*\text{random func.})$

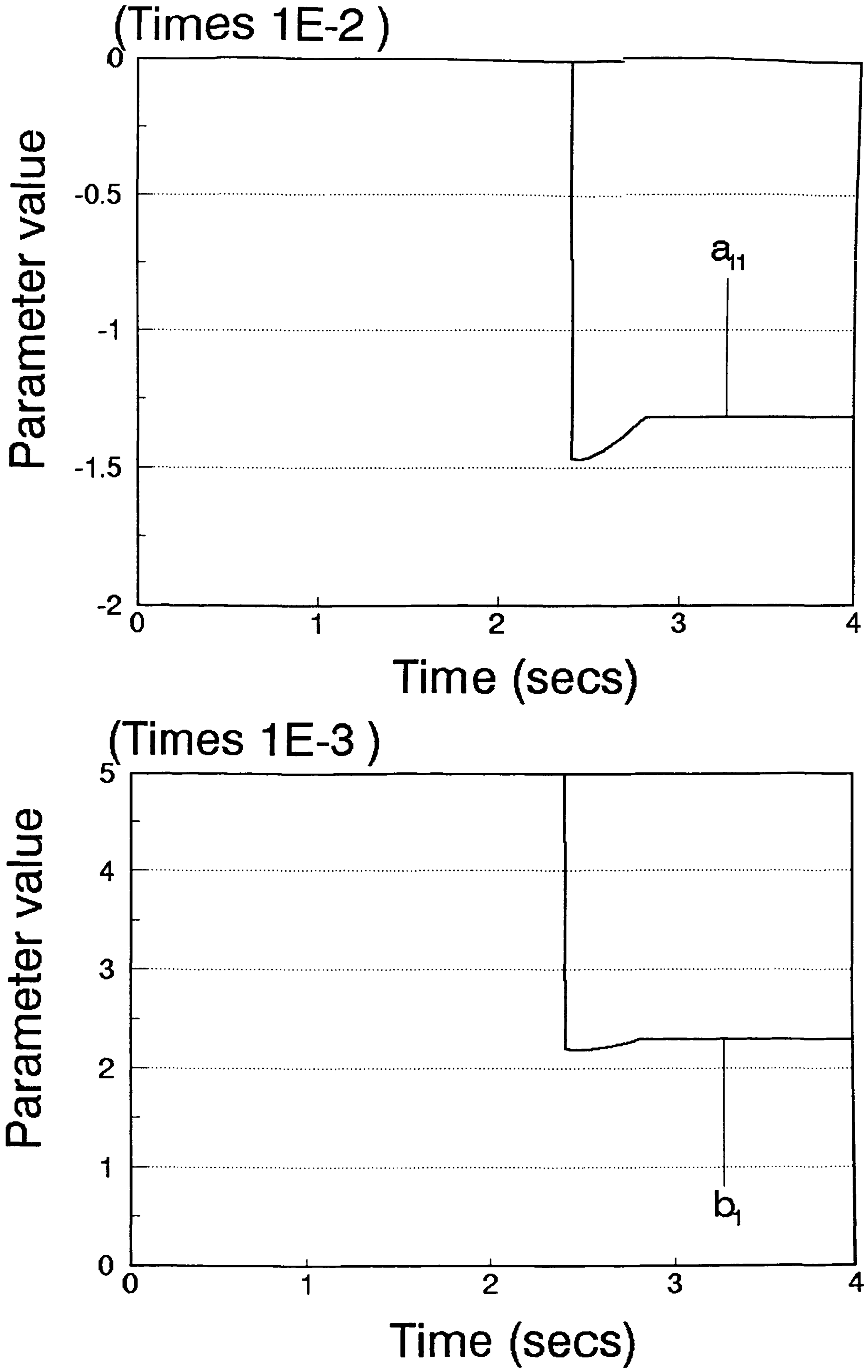


Fig.5.54. Theidentification results with the observation of Fig.5..53

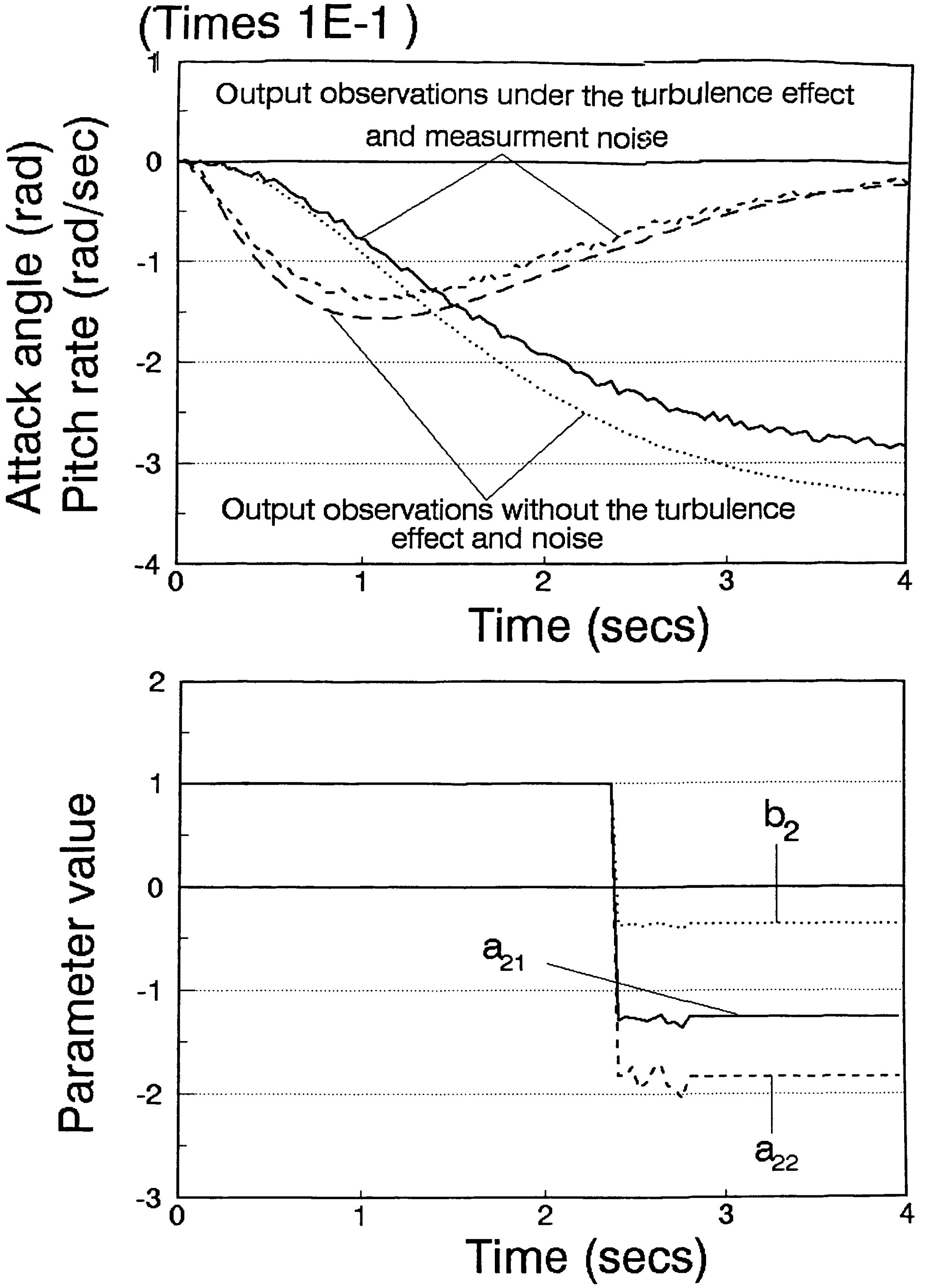
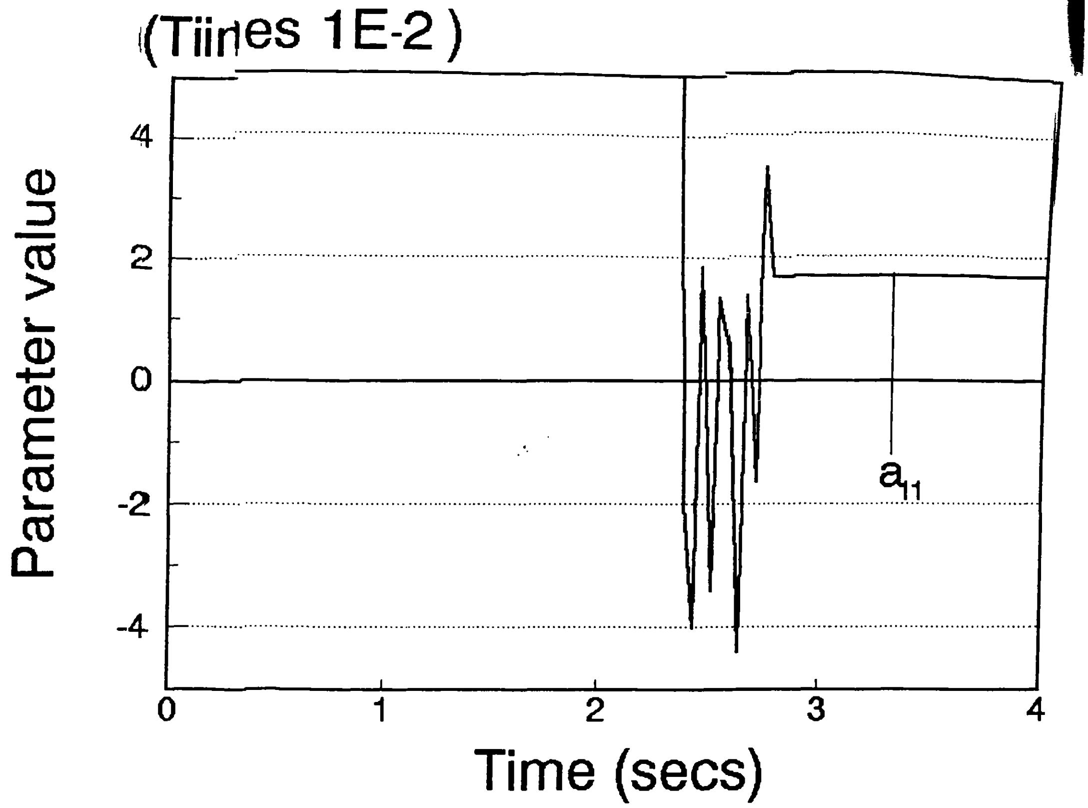


Fig.5.55. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.05$ and observations are $s_i = x_i + 0.01$*random func.



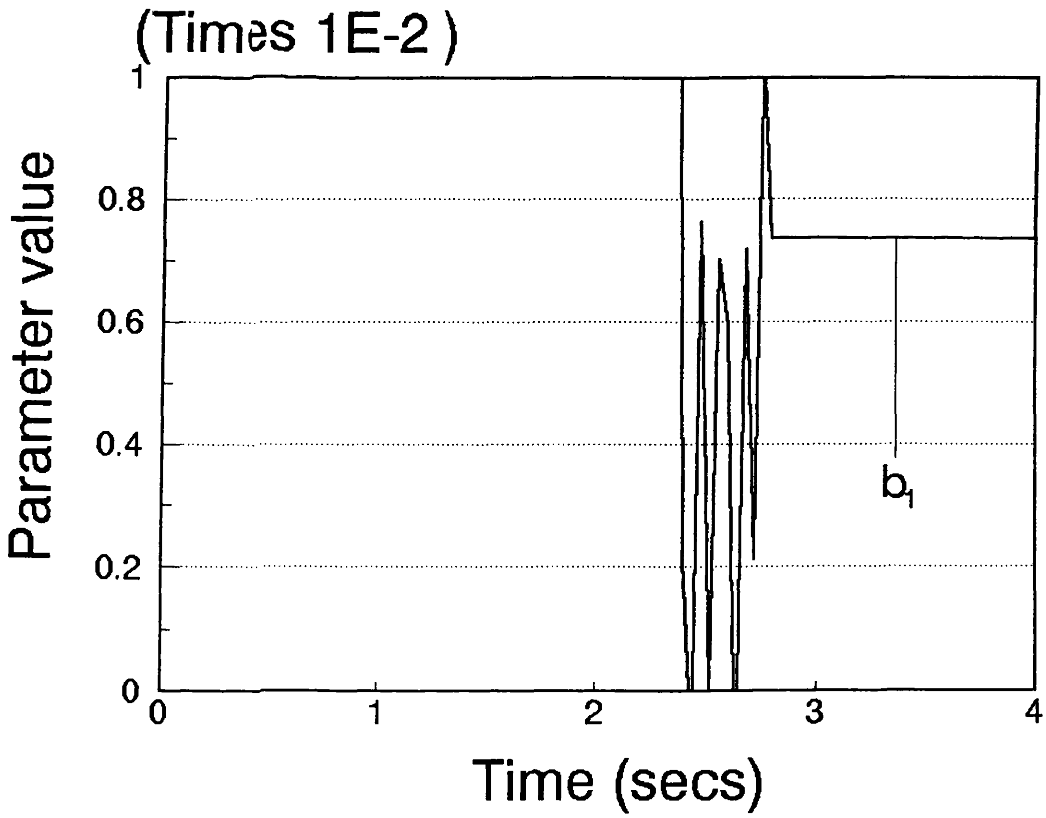


Fig.5.56. The identification results with the observation of Fig.5.55

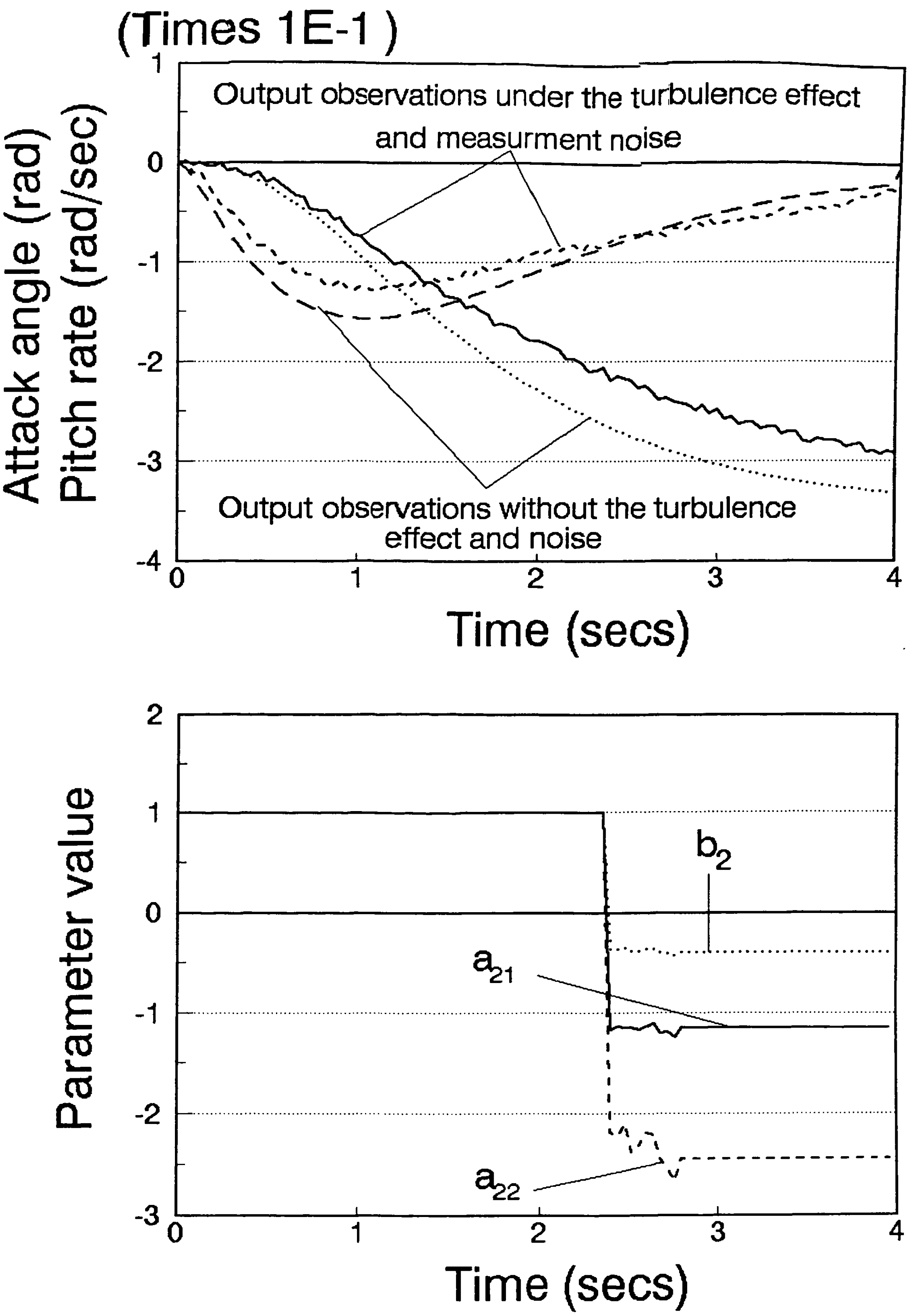


Fig.5.57. The observation and the identification result for atmospheric turbulence effect on the attack angle

$\alpha_g = 0.05*(1 + 0.5*\text{random func.})$

and observations are $s_i = x_i + 0.01*\text{random func.}$

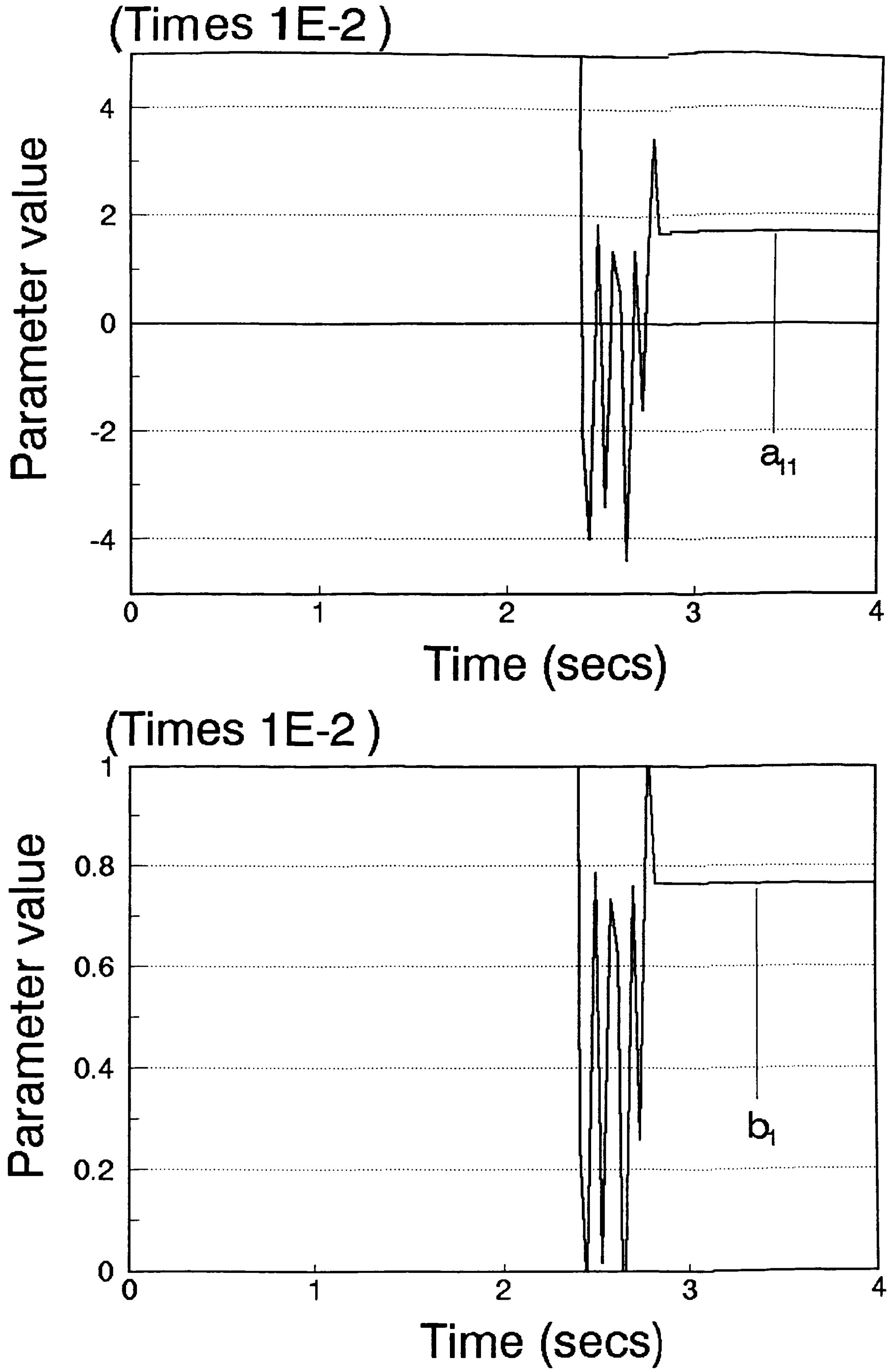
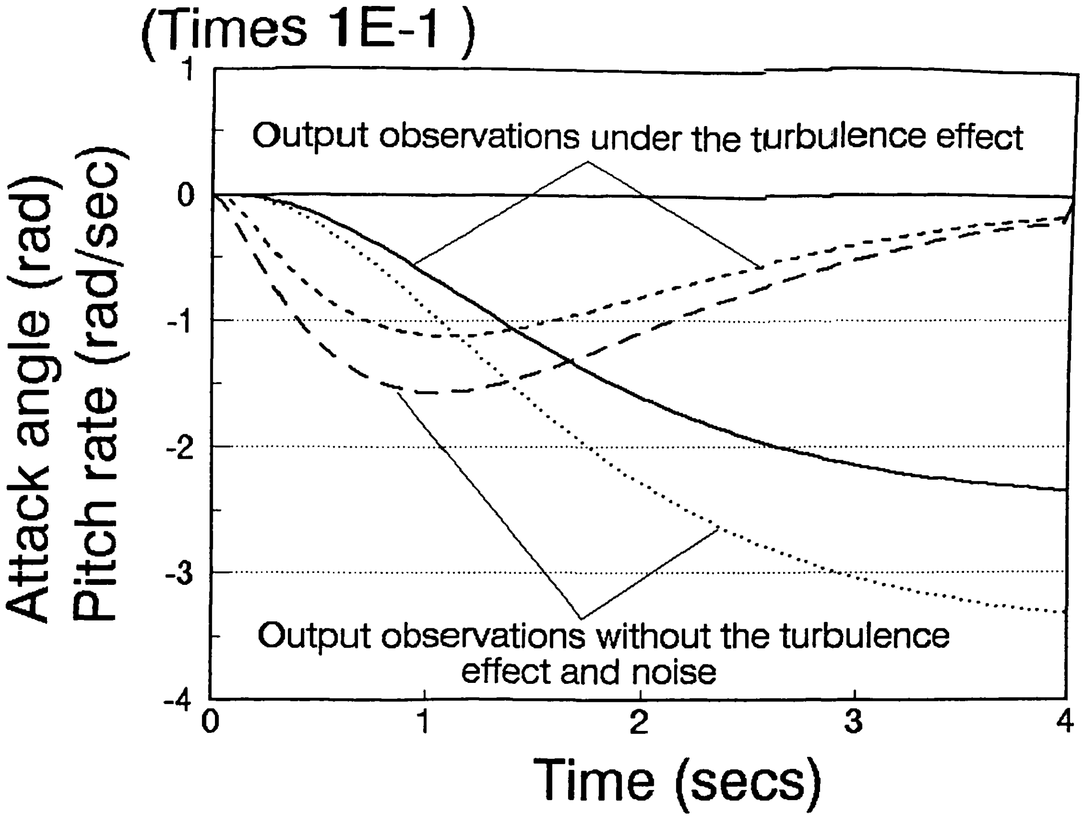


Fig.5.58. The identification results with the observation of Fig.5.57



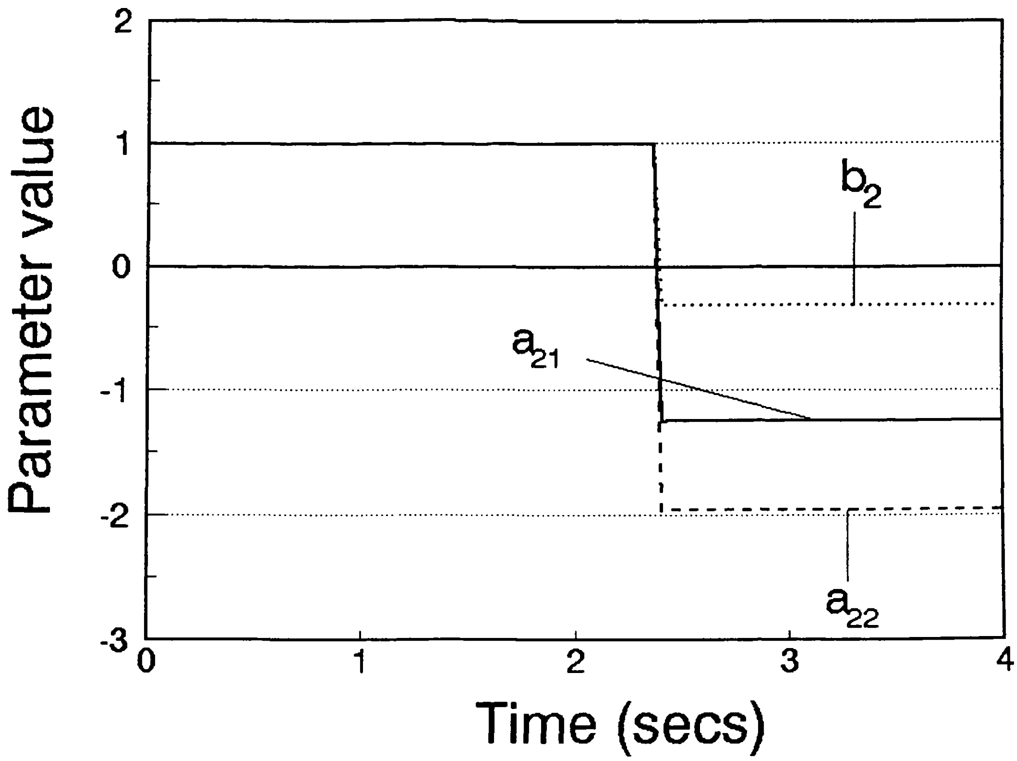


Fig.5.59. The observatiion and the identification result for atmospheric turbulencie effect on the attack angle $\alpha_g = 0.1$

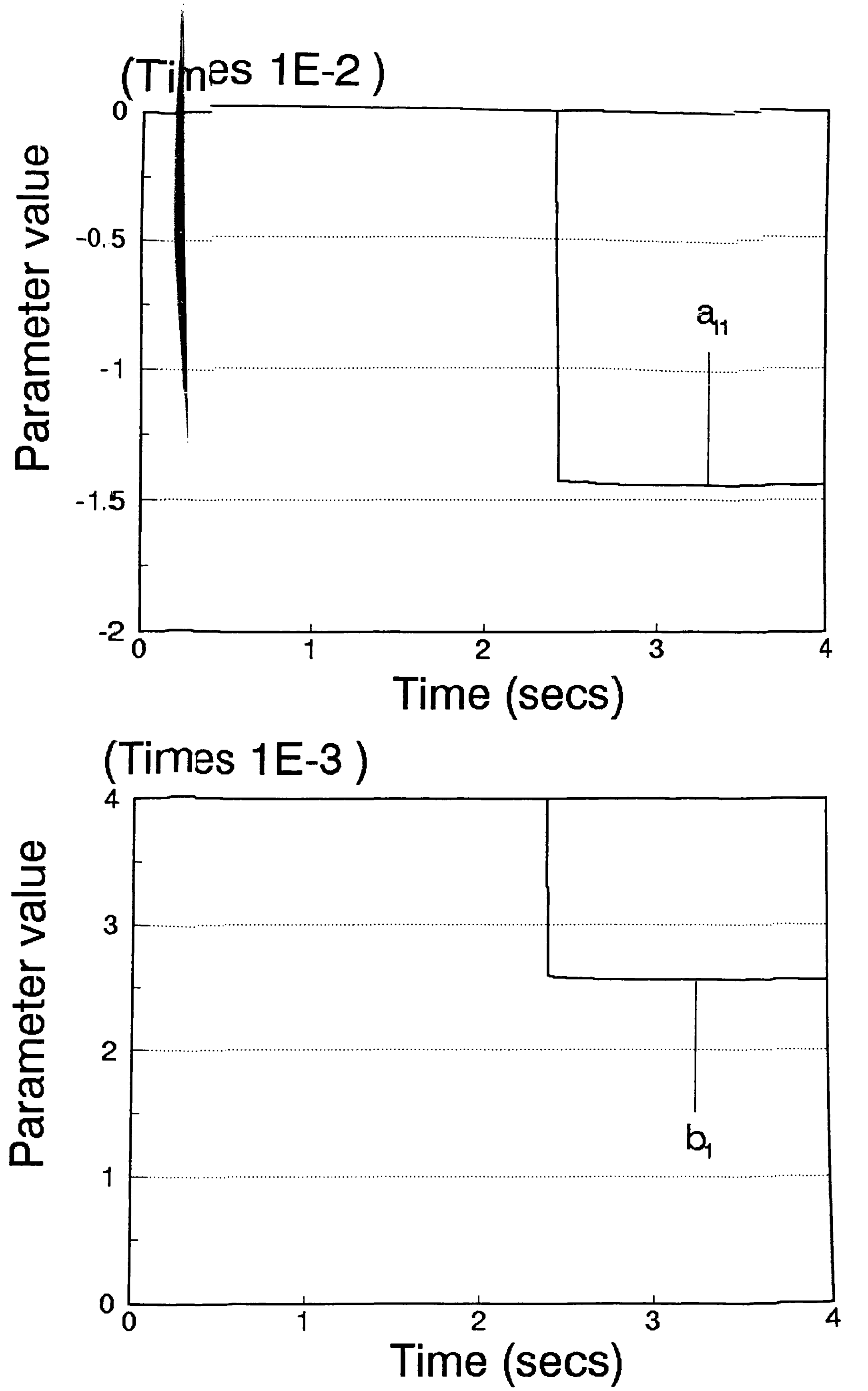


Fig.5.60. The identification results with the observation of Fig.5.59

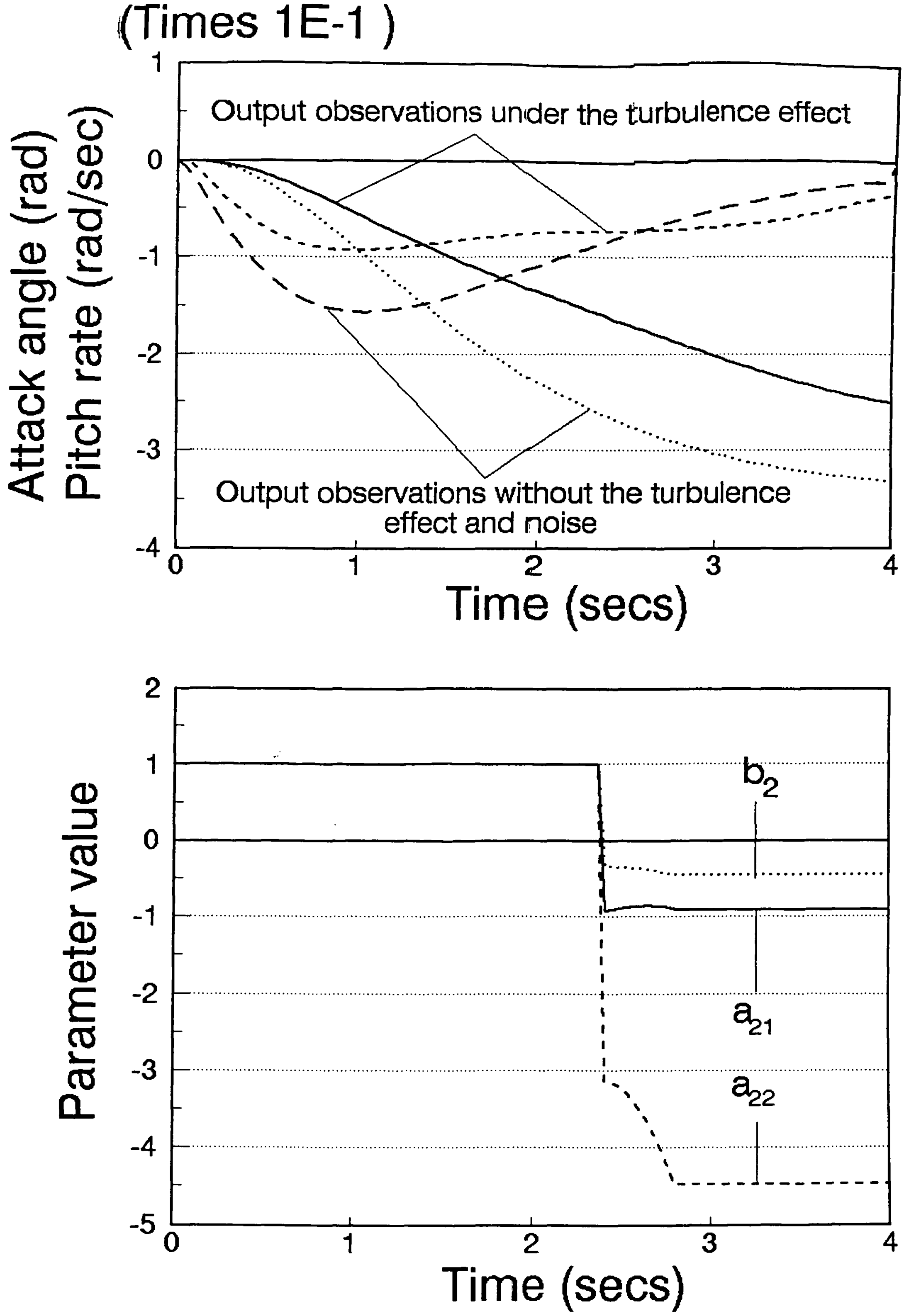


Fig.5.61. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.1*(1 + 0.5*\text{random func.})$

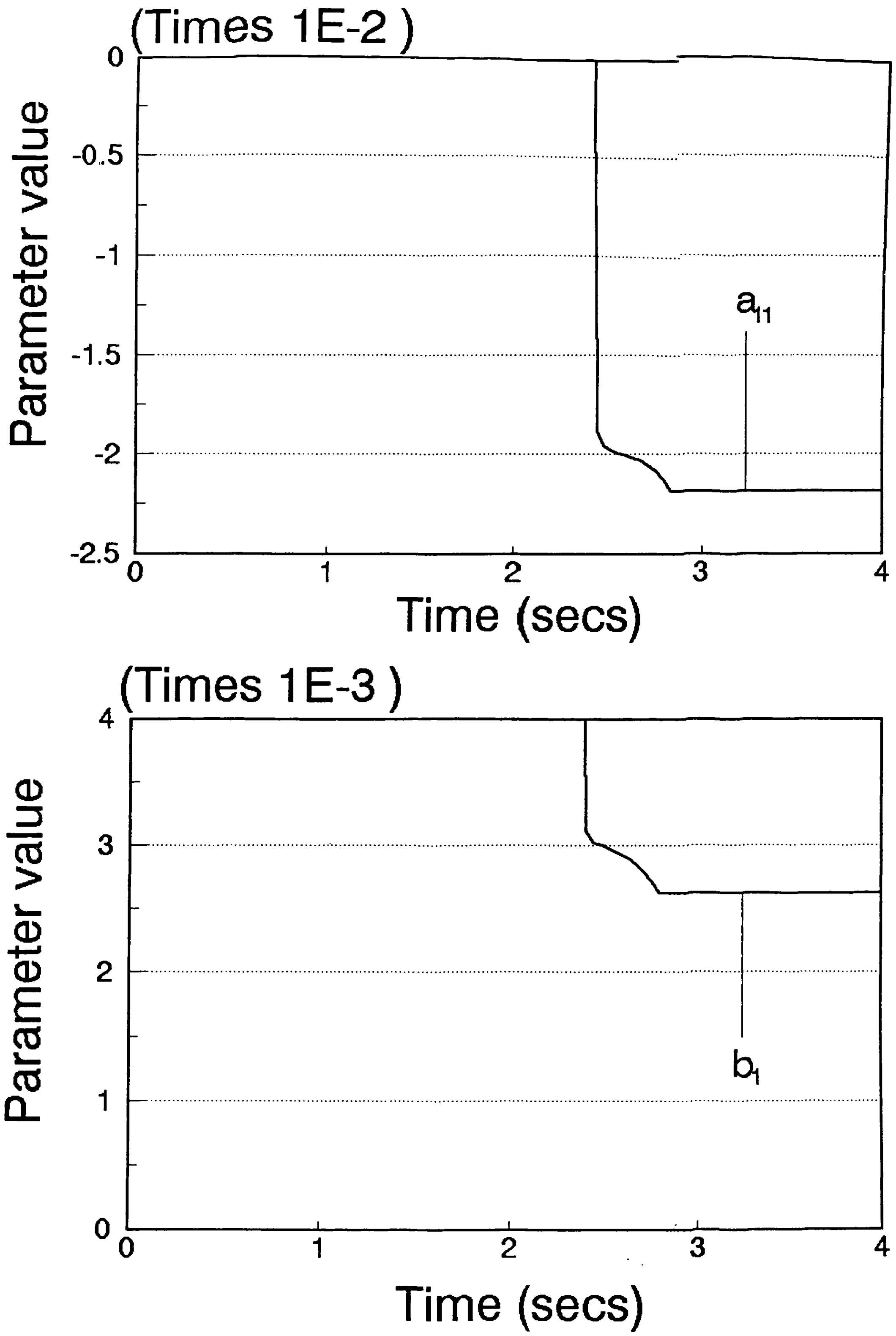


Fig.5.62. The identification results with the observation of Fig.5.61

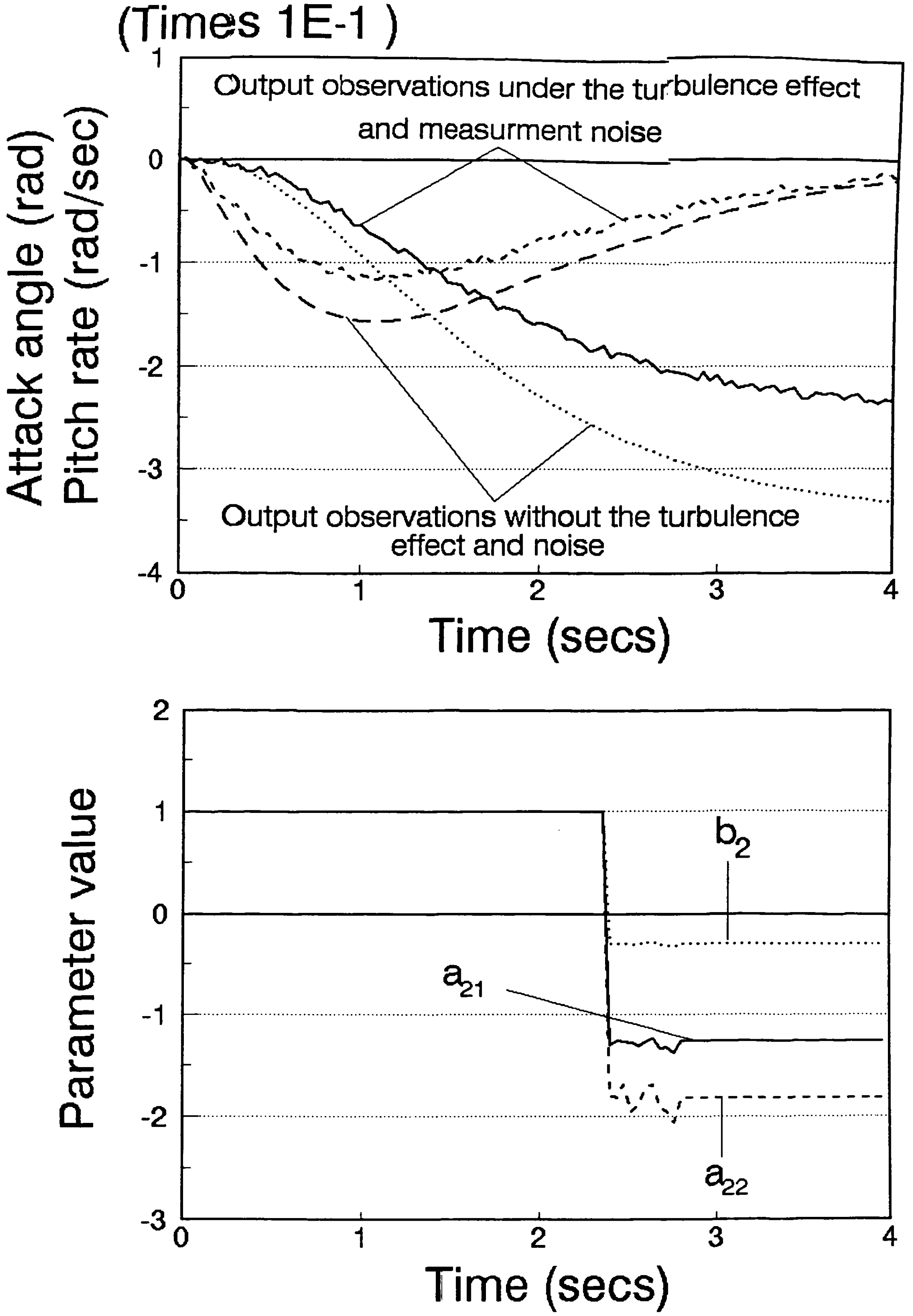


Fig.5.63. The observation and the identification result for atmospheric turbulence effect on the attack angle $\alpha_g = 0.1$ and observations are $s_i = x_i + 0.01$*random func.

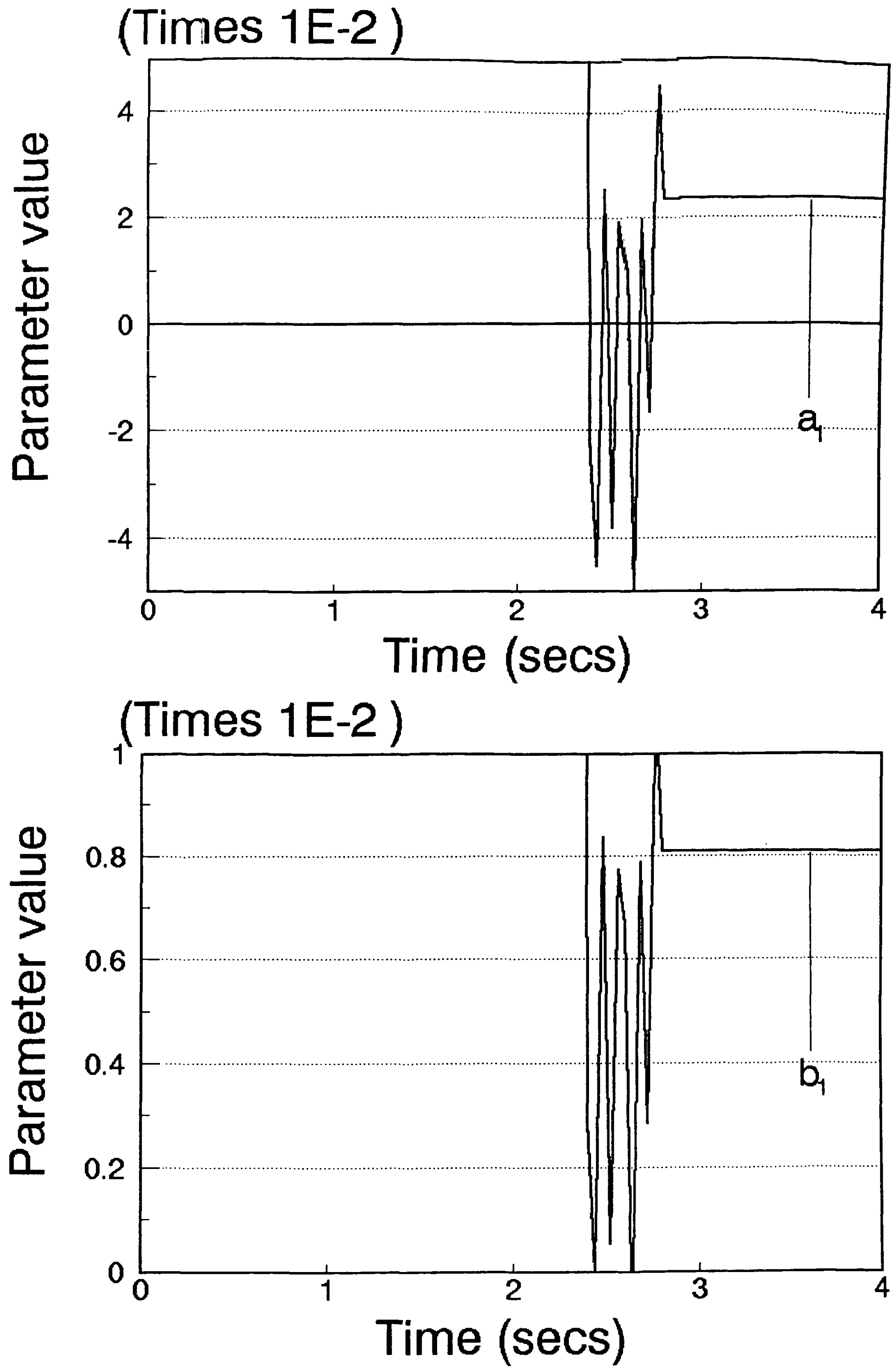


Fig.5.64. The identification results with yhe observation of Fig.5.63

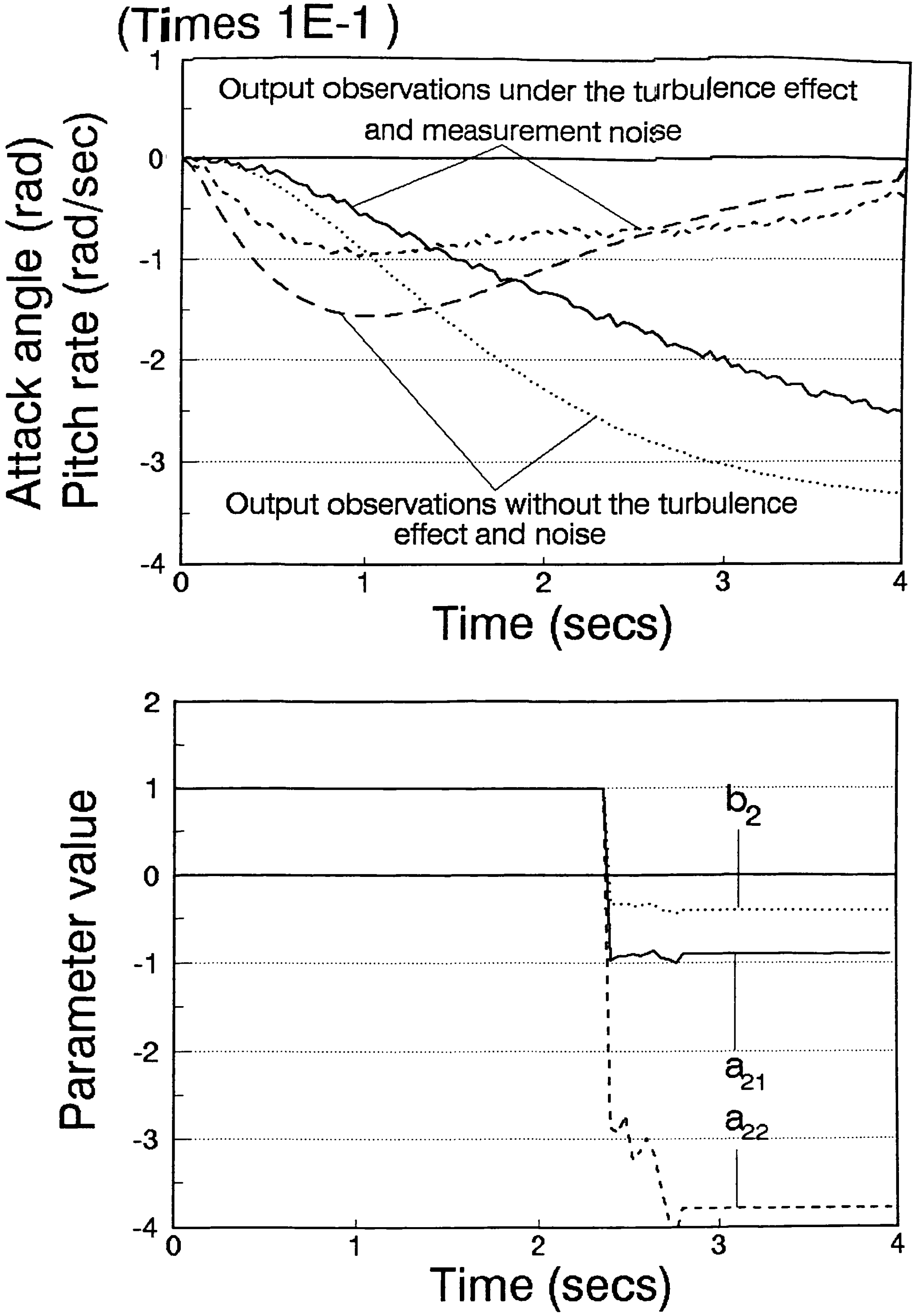


Fig.5.65. The observation and the identification result for atmospheric turbulence effect on the attack angle

$\alpha_g = 0.1*(1 + 0.5*\text{random func.})$

and observations are $s_i = x_i + 0.01*\text{random func.}$

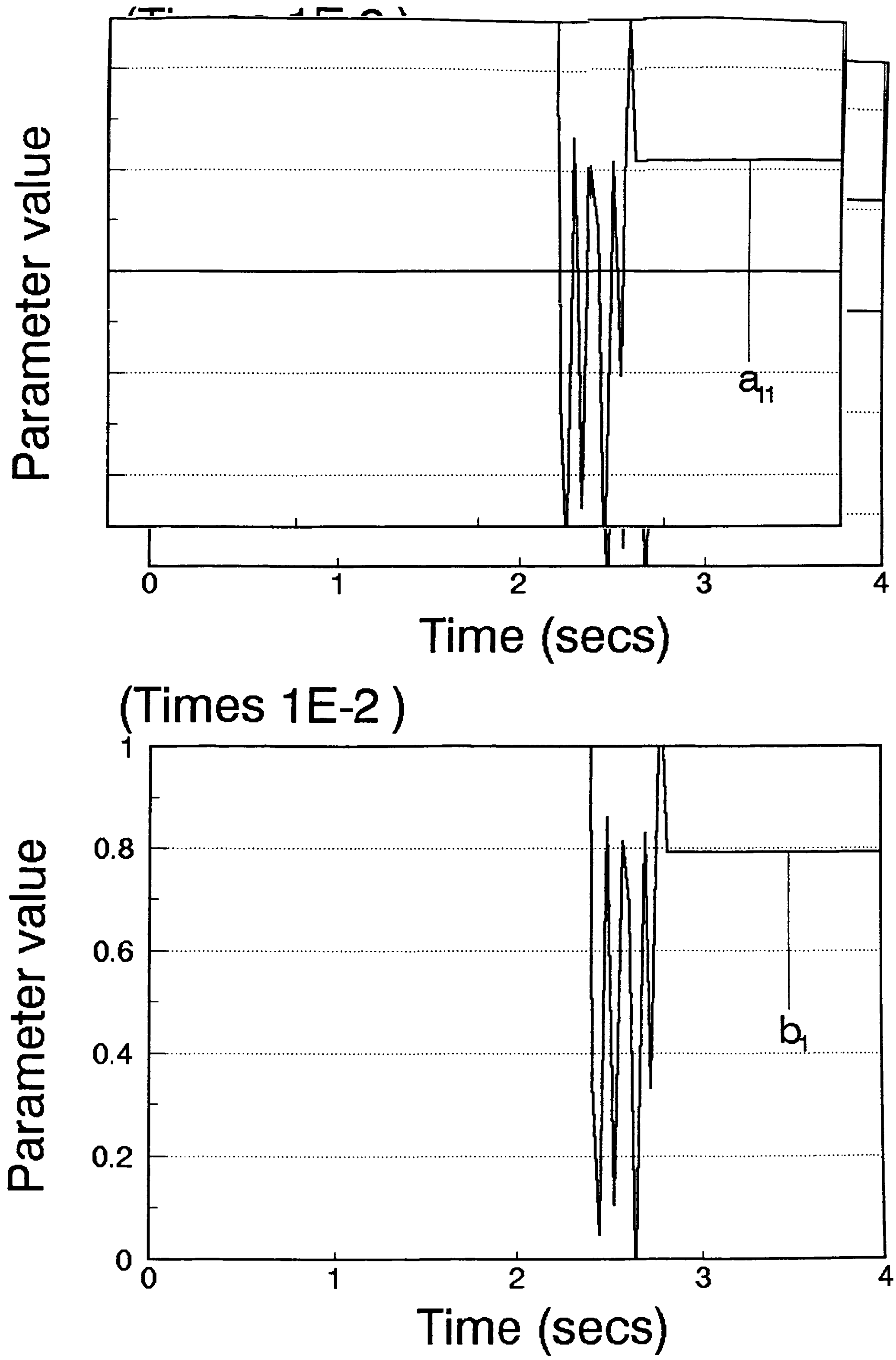


Fig.5.66. The identification results with the observation og fig.5.65

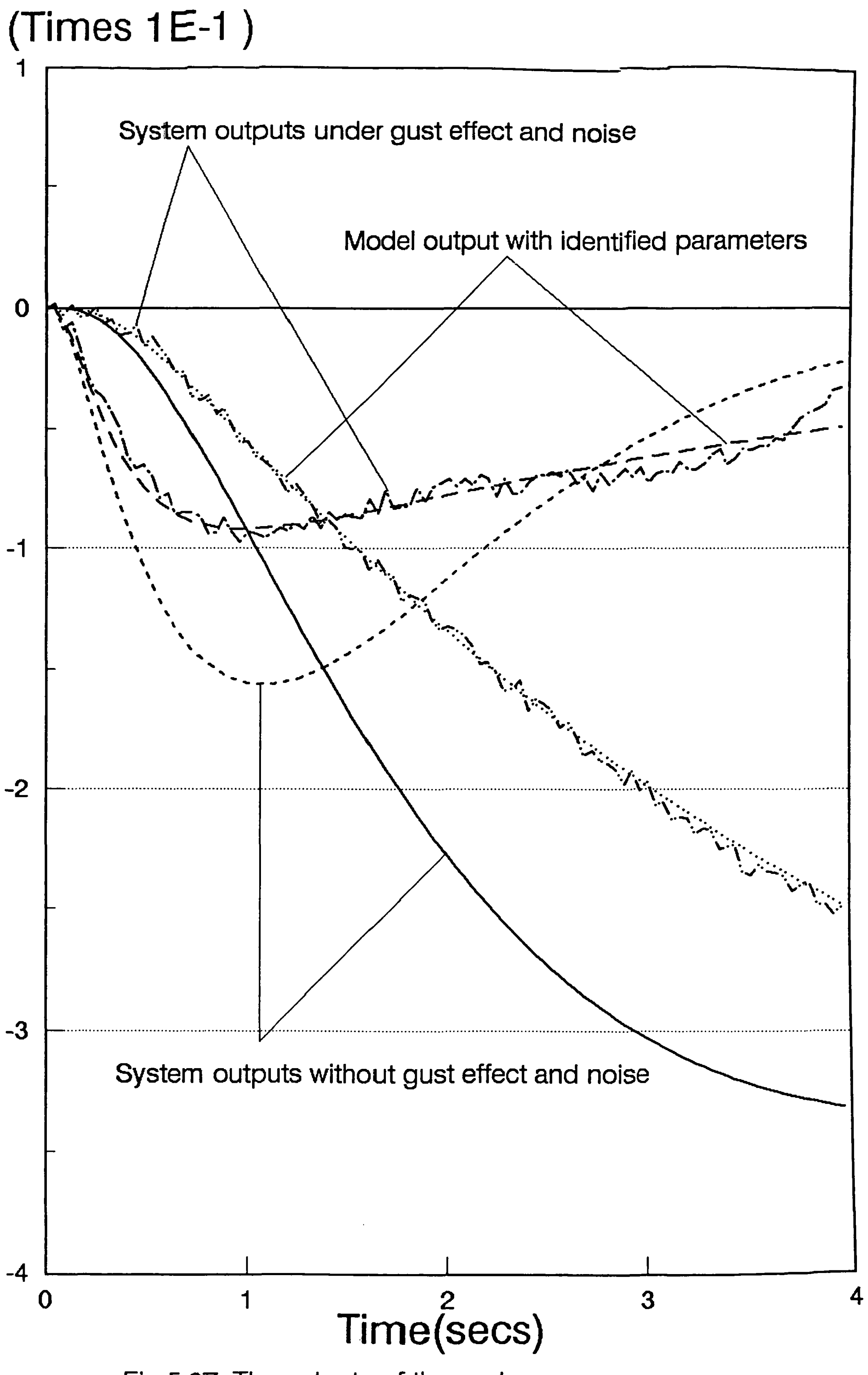


Fig.5.67. The outputs of the systems.
$\alpha_g = 0.1*(1+0.5*\text{random func})$ and observations
$s_i = x_i + 0.01*\text{random func}$ were taken for the
comparision

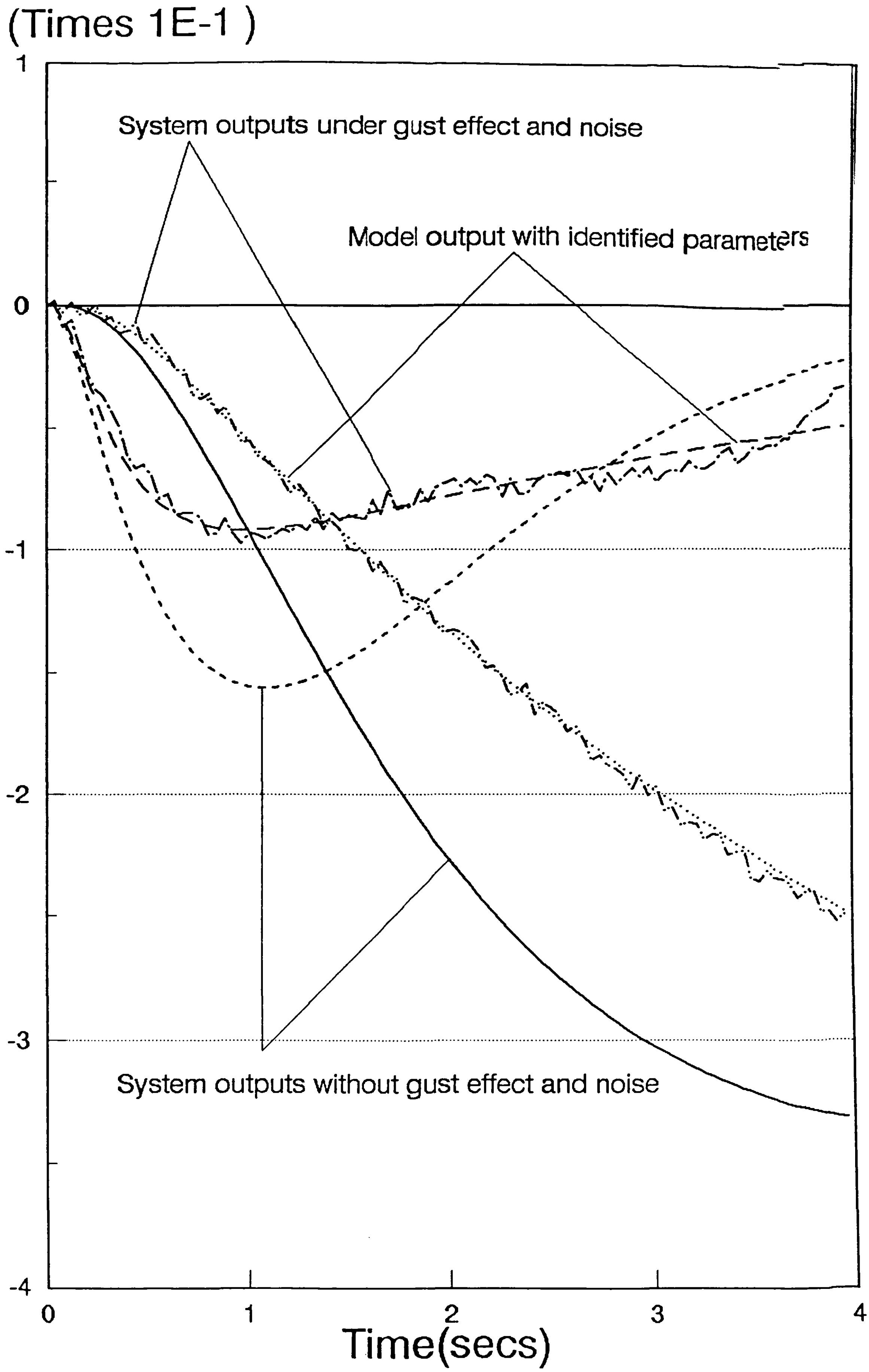


Fig.5.67. The outputs of the systems.
$\alpha_g = 0.1*(1+0.5*\text{random func})$ and observations
$s_i = x_i + 0.01*\text{random func}$ were taken for the
comparision

CHAPTER 6. THE HARDWARE OF THE IDENTIFICATION

6.1. INTRODUCTION

The block diagram of the system identification has been given in Fig.5.1.. In this chapter, the hardware of each block hardware is discussed. The flight modelling block is divided into sub-blocks as shown in Fig.6.1.. The processors' architecture are also discussed .

6.2. Personal Computer

A COMPAQ personal computer has been used for the modelling of the aircraft as well as for communicating with the identification board. It uses Intel 80386 microprocessor that has got 32-bit architecture and 16-MHz processor speed. It is also compatible with 8-MHz 80286 hardware and software. A 16-bit expansion bus has also been built for the P.C. to provide an interface fully compatible with I/O designed for an existing 8-MHz 80286 based system.

Intel 80386 microprocessor has 32 register resources in the following categories [42].

. General Purpose Registers

. Segment Registers

. Instruction Pointer and Flags

. Control Registers

. System Address Registers

. Debug Registers

. Test Registers

All of the 80386 base architecture registers ,which include the general data and address registers, flag registers and instruction pointer, are shown in Fig.6.2.. The other types of registers control, system address, debug and test are primarily used by the system software.

The 80386 processor includes eight general purpose registers of 32 bits to hold data or address quantities. They can be used with 1, 8, 16, 32 and 64 bits. The 32 bit registers are called EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP. They can also be used with the least 16 bits and are named AX, BX, CX, DX, SI, DI, BP, and SP. The least significant 8 bits of 16 bits or the most significant 8 bits of 16 bits can be used separately. The lowest bytes are named AL, BL, CL and DL and higher bytes are named AH, BH, CH and DH, respectively. All of them are illustrated in Fig.6.3.. SI, DI, BP and SP can not be addressed as single byte quantities but 8086 and 80286 programmers are used to this [43]. The usage of the general purpose registers is shown Fig.6.4..

The instruction pointer, which is a 32-bit register, holds the offset of the next instruction to be executed. It can also be used for 16 bit addressing that is named IP. The offset is always relative to the base of the code segment CS.

The flag register is a 32-bit register which is called EFLAGS. The flag bits and bits field are shown in Fig.6.5.. The least significant 16 bits of EFLAGS is named FLAGS, which is used for 8086 and 80286 code. The flag bits meaning are given below.

CF (Carry Flags) : This bit is set when the operation result generates a carry or a borrow. Otherwise CF is zero. It is changed according to all operation code, 8-bit, 16-bit or 32-bit.

PF (Parity Flag) : This flag considers only low order 8 bits of the operation. If the low order 8 bits have an even parity, this flag is set. Otherwise it resets.

AF (Auxiliary Carry Flag) : The auxiliary flag is like a half-carry flag, which is set if the operation resulted in a carry out of bit 3 (addition) or borrow into bit 3 (subtraction). AF is affected by bit 3 only, regardless of overall operand length.

ZF (Zero Flag) : ZF is set, when all bits are zero, otherwise it is reset.

SF (Sign Flag) : If the most significant bit of operation is set, SF is set. It will reflect the state bit of 7, 15, 31 according to the length of operation.

TF (Trap Enable Flag) : This bit enables to single step operation. If this bit is 1, the program executes exactly one step. The single-step continues until this bit becomes zero.

IF (Interrupt Enable Flag) : If this bit is set, external interrupts are recognized otherwise external interrupts are ignored. But (NMI) nonmaskable interrupt can operate independently.

DF (Direction Flag) : This flag determines whether ESI or EDI registers postdecrement (DF=1) or postincrement (DF=0) during the string operation.

OF (Overflow Flag) : This flag is set when the arithmetic operation result is too small or too large which indicates a register length is exceeded

IOPL (Input Output Privilege Level) : This two bit level is peculiar to the Protected mode of operation, and so first appeared on the 80286. It holds the privilege level from 0 to 3, at which your code must be running in order to execute any I/O related instruction.

NT (Nested Flag) : This flag applies to Protected mode. It is used for multitasking operation.

RF (Resume Flag) : This flag is related to the debug operation. By setting it, some exceptions can be masked selectively while debugging.

VM (Virtual 8086 Mode Flag) : When this flag is set, the 80386 is essentially converted into high speed 8086 until the bit is cleared again.

For more information see, for example, references [42] to [45].

6.3. The I/O Interface Card

The COMPAQ computer I/O is compatible with an 80286 microprocessor as mentioned in the previous section. I/O is possible via a 16-bit Expansion Bus which is given in Fig.6.6.. The pins' functions are given by Table 6.1. [44]. Table 6.1. indicates that some pins should be driven by 20 mA. In practice, each address or data line of the processors can not give 20 mA. In this case, a drive interface is needed to put between the 16-bit I/O Expansion Bus and the communication interface. This interface also protects the 16-bit I/O Expansion Bus of the processor from I/O device and bus faults. A prototype card has been designed for this purpose. It has been placed between 300h and 31Fh address line. This standard address bus has been designed by the manufacturers [45]. The prototype card circuit diagram is given by Fig.6.7. It has been connected to the 80386 processor via a 16-bit I/O Expansion Bus Connector and also been connected to the communication interface via a D-type connector, which can be seen in Fig.6.8..

6.4. The Communication Interface Card

The communication between the Personal Computer and TMS320C30 Digital Signal Processor Board has been realized by a 16-bit parallel communication. The communication interface card has been designed to make the communication between the different systems, because both systems operate at different speeds and with different signals. A protocol is also necessary for two-way communications. This interface card has been built on the same board as the TMS320C30 DSP. Its circuit diagram is given in Fig.6.9.. The communication interface operation can be explained by the following steps:

Step 1) The data transmitted from the P.C. via the 300H address line is connected to the latch input (IC 74AS652). Simultaneously, 300H address and I/O commands are solved and DINL is generated by the COMP I/O (PAL PLS153), which is programmed as in Fig.6.21.. The DINL signal is provided before the data bus line. On the other hand, the latch I.C. is triggered by raising edge. In this case, DINL transfers the 3-state to latch. Therefore DINL is delayed for the suit transfer. These signals are given in Fig.6.10..

Step 2) The P.C. sends data via the 304H address line in order to point the communication direction from P.C. to the identification board. The COMP I/O recognizes this data and modifies the XF0 through D-flip flop (IC 74F74).

Step 3) When data is to be read from the latch by the TMS320C30, XF0 is checked firstly. If data is ready to read, XF0 has already been set in *step 2.* TMS320C30 reads data from the 804000H address line. At the same time, TMS I/O (PAL PLS153), which has been programmed as in Fig.6.22., generates DINEN and data appears on TMS320C30 data line.

These three steps are used for the data transfer from COMPAQ to TMS320C30. The other direction transfer can be made with the same method by using DOUTL, $\overline{\text{DOEN}}$ and XF1 and IC 74F244. DOUTL, which can be seen in Fig.6.20.. Acknowledgement signals are assumed to be $\overline{\text{CFEN}}$ and CFDL. After the transfer, flags are cleared by $\overline{\text{CFCLR}}$ and $\overline{\text{CFDCR}}$.

6.5. TMS320C30 Digital Signal Processor

The identification board has been built with only TMS320C30 Digital Signal Processor without an external memory. TMS320C30 is a high performance CMOS 32-bit DSP. It contains a 32-bit floating-point Central Processing Unit, 2K 32-bit on-chip RAM, 64 32-bit instruction cache, 2 serial ports, 2 timers and Direct Memory Addressing Unit (DMA). Its speed is up to 33 MHz. A 4K 32-bit on-chip ROM option is available [46], but off-chip (external) EPROM has been used on the identification board.

6.5.1. Central Processing Unit (CPU)

The CPU consists of the Arithmetic Logic Unit (ALU), two Auxiliary Register Arithmetic Units (ARAUs), a multiplier and 28 registers.

The ALU performs single-cycle operations on a 32-bit integer, 32-bit logical and 40-bit floating point data, including single-cycle integer and floating point conversions. ALU also includes a barrel shifter which is used to shift up to 32 bits left or right in one cycle.

ARAUs can generate two addresses in a single-cycle. They can also be used for arithmetical and logical operations between auxiliary registers. They operate in parallel with the ALU and the multiplier. They support different addressing modes.

The multiplier performs multiplication integer and floating points values. It can multiply 24-bit integer and 32-bit floating points values in a single-cycle. The results of the multiplier are in 32-bit integer or 40-bit floating point format.

The TMS320C30 includes 28 registers for addressing, data, control and stack purposes, which are given in Fig.6.11.. Eight of them (R0-R7), which are named extended precision registers, are capable of storing and operations. They are 40 bits registers, which can be used for 32-bit integer or 40-bit floating point operations. The bits 39-32 are not changed in integer operation which is either signed or unsigned.

Auxiliary registers (AR0-AR7), which are 32-bit registers, are used for addressing by the CPU and modified by the ARAUs. They can be used for different purposes, such as a loop counter. They generate 24-bit address, but they can be used as a 32-bit general purpose register; for example, an extended precision register by multiplier.

The 32-bit index registers (IR0, IR1) are used by the ARAUs for the address indexing. The 32-bit block size register BK is also used by ARAUs for circular addressing to specify the data block size.

The system stack pointer SP is a 32-bit register which points the top of the system stack. The status register ST shows the state of CPU. Fig.6.12. illustrates the bit field and bit names which are explained in Table 6.2..

CPU/DMA interrupt enable register IE determines which CPU or DMA external interrupts can be recognized or ignored. If any bit is 1, this external interrupt will be recognized otherwise it will be ignored. All external interrupts can be masked. IE bit names and bit fields are shown in Fig.6.13. and explained in Table 6.3..

The CPU interrupt flag IF indicates which interrupt is set. The IF bit is set to 1 when an interrupt occurs. Register bit names and functions are given by Fig.6.14. and Table 6.4..

I/O flag register IOF controls the external pins XF0 and XF1. This register and the pin configuration are shown in Fig.6.15. and Table 6.5.

Repeat Counter register RC is a 32-bit register and contains the repeat number of the block operation. 32-bit repeat starting address register RS shows the block operation starting address. The block operation ending address is shown by 32-bit repeat end address register RE.

Program Counter PC is a 32-bit register containing the address of the next instruction to be fetched.

6.5.2. Memory Map

The TMS320C30 can address 16 M memory space. The usages of memory are shown in Fig. 6.16.. First 192 locations are for reset, interrupt vectors, trap vectors and a reserved place. Reserved memory space should not be read and written, otherwise the TMS320C30 may be halted and required a system reset to restart.

6.5.3. The TMS320C30 Circuit

The TMS320C30 circuit has been designed according to external interface categories. These categories can be illustrated by Fig.6.17.. The EPROM has been connected to the primary bus line (C0H-0FFFH). The reading signals of the EPROM device are shown in Fig.6.19.. TMS I/O, which generates data latch and clear flag, is connected to the expansion bus addresses 804000H-804004H. XF0 and XF1, which are used to point to the direction of communication and are tied to the interface communication board. The expansion bus circuit diagram has already been given with the communication circuit diagram in Fig.6.9.. The primary bus circuit diagram is given by Fig.6.18. All memory and device are suitable to work with zero wait state.

6.6. Conclusion

The communication between two microprocessors, and the identification board design have been given in this chapter. Also the processors' architectures have been given briefly. Two different interfaces are used to communicate between two processors. Because the 16-bit I/O Expansion Bus Connector of the 80386 requires a drive interface which has already been built. The purpose of the other, which is called the communication interface, is to achieve the communication between the processors. The PAL devices have been programmed to code and decode the signals. It has to be considered that the COMP I/O device decodes the COMPAQ's address line and produces the latch enable signal before the COMPAQ's data.

Signal Name	Description
IOCHK	This input signal is used to signal the CPU about parity or other serious errors on expansion memory boards plugged into expansion bus. This signal should be driven low by an open-collector type output capable of sinking 20 mA when an uncorrectable system error occurs.
SD0-SD7	These bidirectional signals are the low 8 bits of the system data bus.
SD8-SD15	These bidirectional signals are the high 8 bits of the system data bus.
BUSRDY	This input signal lengthens a bus cycle from its standard time when an expansion board respond quickly enough. It should be pulled low by an open-collector type device as soon as a slow addressed device is selected and held low until the device has responded. This line should not be held low for more than 2.5 microseconds (μs). This line should be driven by an open-collector device capable of sinking 20 mA.
AEN	This output signal when inactive (low) indicates that the CPU or other bus master has control of the bus. When active, the DMA controller has control of the bus. This signal is often used to disable devices that must not respond during a DMA cycle.
SA0-SA19	These bidirectional signal address memory or I/O devices within the system. They are the low order 20 bits of the 24-bit address line. They are enabled while BALE is high and are latched with BALE's falling edge.
RESDRV	This output signal resets the hardware during power on or power failure.
IRQ3-IRQ15	They are interrupt line that can be recognized when a line goes from a low to high state.
DRQ0-DRQ7	The input signals are used to request a DMA service.
DAK0-DAK7	These output signals are DMA acknowledge signal.

Table 6.1. The 16-Bit Expansion Bus Signals (Continued)

NOWS	This input signal indicates the No Wait System. This pin must be pulled low before the falling edge of BCLK. It should be driven by an open-collector device capable of sinking 20 mA.
SMWTC	This output signal is Standart Memory Write that is active (low) for an address between 000000h and 0FFFFFh.
SMRDC	This output signal is Standart Memory Read that is active (low) for an address between 000000h and 0FFFFFh.
IOWC	When this output signal (I/O Write) is low, data is assumed to accept by an I/O device.
IORC	When this output signal (I/O Read) is low, data is assumed to send to data bus by an I/O device.
REFRESH	This output signal indicates (when low) a refresh cycle in progress.
BCLK	It is 8 Mhz clock signal which is synchronize with main processor clock.
T/C	This output signal indicates that terminal count of a DMA operation has been reached. It should be decoded with appropriate DAK line.
BALE	This output signal (when high) indicates that a valid address is present on the LAxx address line. This line is always high when a DMA or bus master operation is occurring.
OSC	This is a clock signal for timing operation. Its frequency is 14.31818 Mhz with a 50 percent duty cycle.
SBHE	This is System Bus High Enable signal which indicates 16-Bit data transfer.
LA17-LA23	They are Latchable Address signals which decode memory according to 0 or 1 wait states. They are valid only BALE is high.
MRDC	This Memory Read signal is low, when a memory device is to send data to data bus with any address of the entire adress space of the system.

Table 6.1. The 16-Bit Expansion Bus Signals (Continued)

MRDC	This Memory Write signal is low, when a memory device is to accept data from data bus with any address of the entire adress space of the system.
M16-	This input signal (Memory is 16 bits) notifies the system that the addressed memory is capable of transferring 16 bits of data at once. When this line is made active during a memory read or write, the standart, 1-wait-state memory cycle is run. This line should be derived from the LAxx address lines. It should be driven by an open-collector device capable of sinking 20 mA.
IO16-	This input signal (I/O is 16 bits) notifies the system that the addressed I/O device is capable of transferring 16 bits of data at once. When this line is made active during an I/O read or write, the standart, 1-wait-state I/O cycle is run. It should be driven by an open-collector device capable of sinking 20 mA.
GRAB	This input signal indicates that a board-mounted bus master is controlling the bus.
GND	These lines are connected to the system DC ground. The maximum current allowe on any single contact is 1.5 A.
+5 VDC -5 VDC +12 VDC -12 VDC	These lines are connected to the essential power supplies.

Table 6.1. The 16-Bit Expansion Bus Signals (Concluded)

BIT	NAME	FUNCTION
0	C	Carry flag
1	V	Overflow flag
2	Z	Zero flag
3	N	Negative flag
4	UF	Floating-point underflow fl..g
5	LV	Latched overflow flag
6	LUF	Latched floating point underflow flag
7	OVM	Overflow mode flag. This flag affected only the integer operation. When this flag is zero, overflow is normal. When OVM=1, integer results overflowing are set to the most positive 32-bit two's complement number (7FFFFFFFh) or the most negative two's complement number according the result's sign.
8	RM	Repeat mode flag. If this flag is set, the PC is modified reaeat operations.
9	Reserved	Read as 0
10	CF	Cache Freeze.
11	CE	Cache Enable
12	CC	Cache Clear
13	GIE	Global Interrupt enable. If the GIE=1, the CPU responds to an enabled interrupt. If the GIE=0, the CPU does not respond to an enabled interrupt.
14	Reserved	Read as 0.
15	Reserved	Read as 0.
16 31	Reserved . . . Reserved	Value undefined.

Table.6.2. Status Register Bits Summary.

BIT	NAME	FUNCTION
0	EINTO	Enable external interrupt 0 (CPU)
1	EINT1	Enable external interrupt 1 (CPU)
2	EINT2	Enable external interrupt 2 (CPU)
3	EINT3	Enable external interrupt 3 (CPU)
4	EXINTO	Enable serial port 0 transmit interrupt (CPU)
5	ERINTO	Enable serial port 0 receive interrupt (CPU)
6	EXINT1	Enable serial port 1 transmit interrupt (CPU)
7	ERINT1	Enable serial port 1 receive interrupt (CPU)
8	ETINTO	Enable timer 0 interrupt (CPU)
9	ETINT1	Enable timer 1 interrupt (CPU)
10	EDINT	Enable DMA controller interrupt (CPU)
11-15	Reserved	Value undefined
16	EINTO	Enable external interrupt 0 (DMA)
17	EINT1	Enable external interrupt 1 (DMA)
18	EINT2	Enable external interrupt 2 (DMA)
19	EINT3	Enable external interrupt 3 (DMA)
20	EXINTO	Enable serial port 0 transmit interrupt (DMA)
21	ERINTO	Enable serial port 0 receive interrupt (DMA)
22	EXINT1	Enable serial port 1 transmit interrupt (DMA)
23	ERINT1	Enable serial port 1 receive interrupt (DMA)
24	ETINTO	Enable timer 0 interrupt (DMA)
25	ETINT1	Enable timer 1 interrupt (DMA)
26	EDINT	Enable DMA controller interrupt (DMA)
27-31	Reserved	Value undefined

Table 6.3. IE Register Bits

BIT	NAME	FUNCTION
0	INT0	External interrupt 0 flag
1	INT1	External interrupt 1 flag
2	INT2	External interrupt 2 flag
3	INT3	External interrupt 3 flag
4	XINT0	Serial port 0 transmit interrupt flag
5	RINT0	Serial port 0 receive interrupt flag
6	XINT1	Serial port 1 transmit interrupt flag
7	RINT1	Serial port 1 receive interrupt flag
8	TINT0	Timer 0 interrupt flag
9	TINT1	Timer 1 interrupt flag
10	DINT	DMA channel interrupt flag
11-31	Reserved	Value undefined

Table 6.4. IF Register Bits

BIT	NAME	FUNCTION
0	Reserved	Read as 0
1	$\overline{I}$/OXF0	If $\overline{I}$/OXF0 = 0, XF0 is configured as a general input pin. If $\overline{I}$/OXF0 = 0, XF0 is configured as a general output pin.
2	OUTXF0	Data output on XF0
3	INXF0	Data input on XF0. A write has no effect.
4	Reserved	Read as 0
5	$\overline{I}$/OXF1	If $\overline{I}$/OXF1 = 0, XF1 is configured as a general input pin. If $\overline{I}$/OXF1 = 1, XF1 is configured as a general output pin.
6	OUTXF1	Data output on XF1
7	INXF1	Data input on XF1. A write has no effect.
8-31	Reserved	Read as 0

Table 6.5. IOF Register Bits

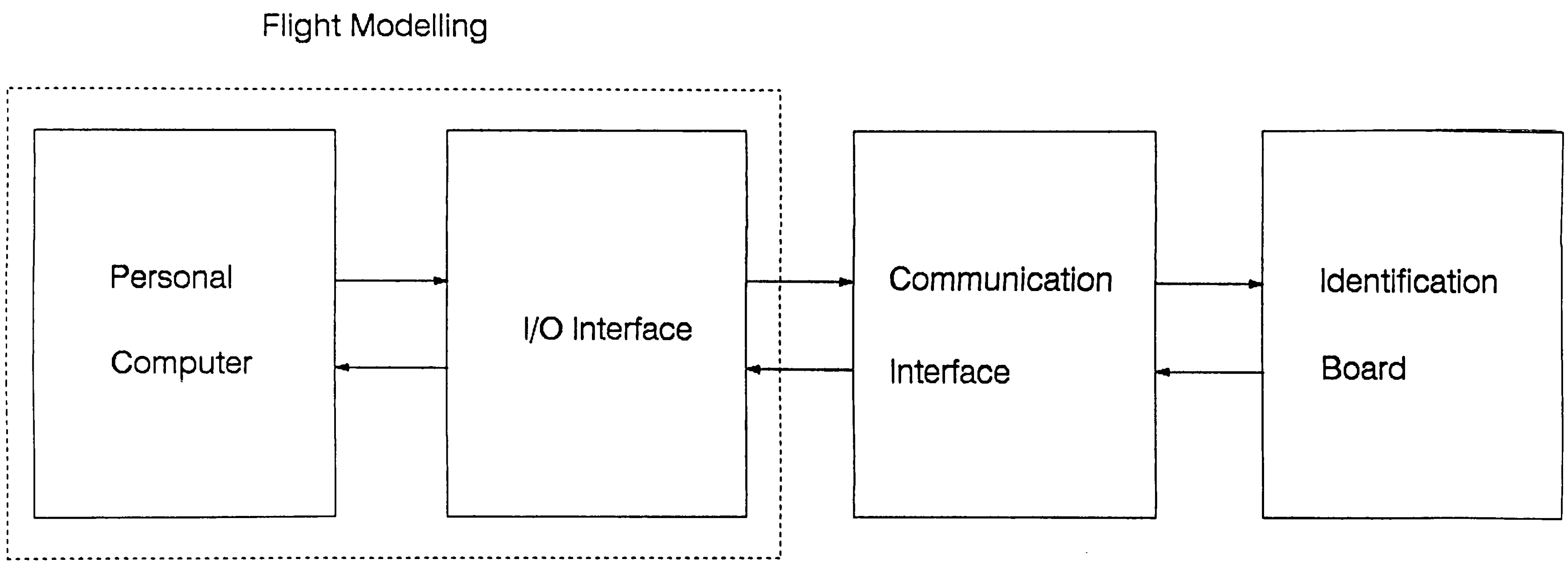


Fig.6.1. The Block Diagram of The Identification

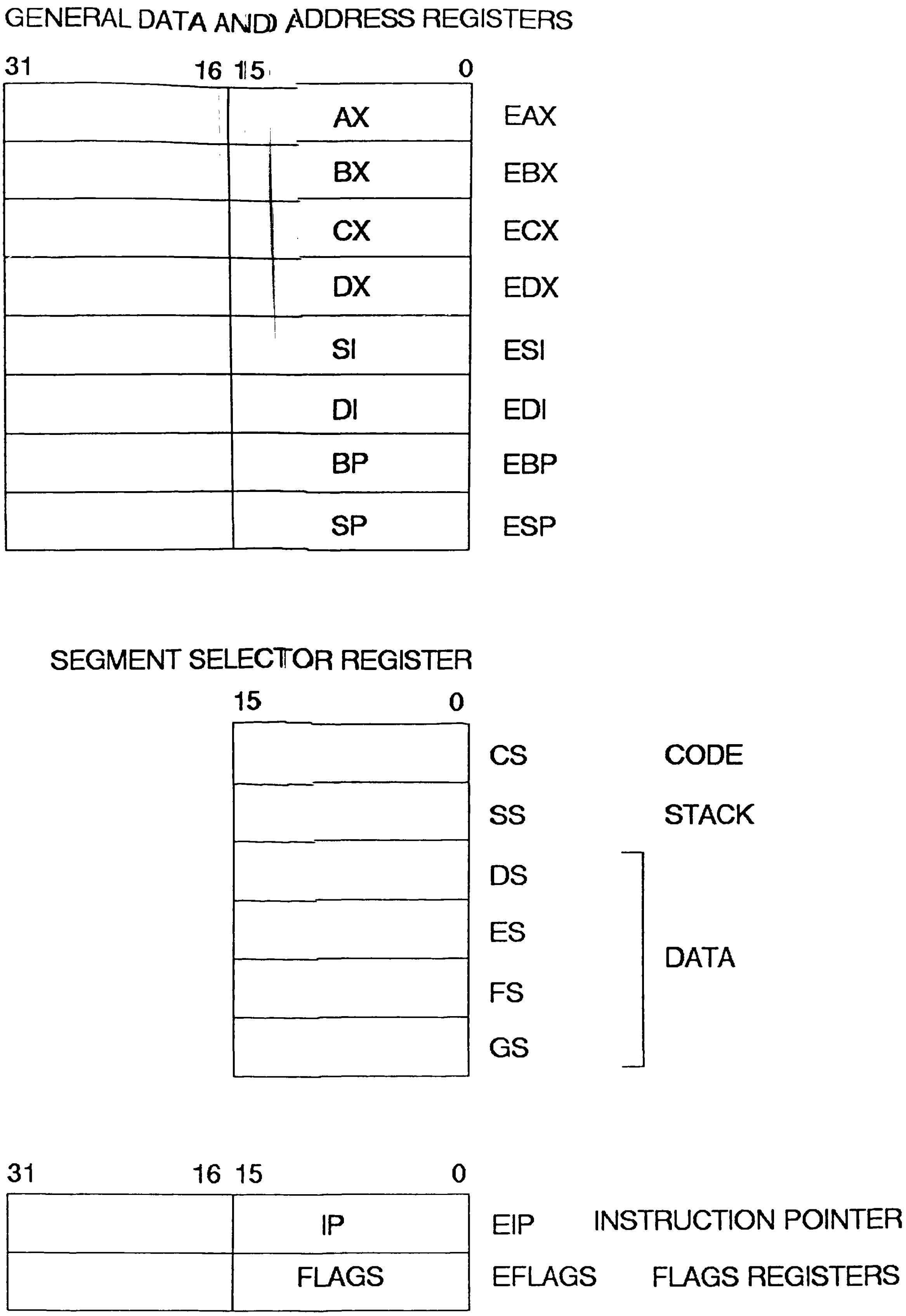


Fig.6.2.80386 Base Architecture Registers

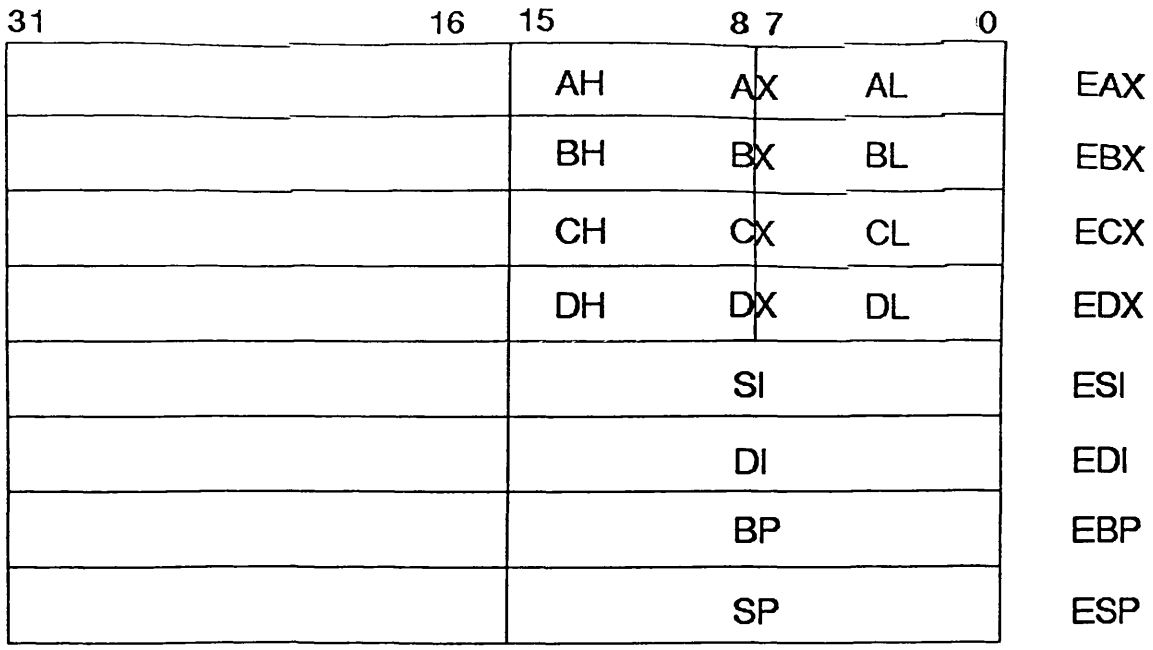


Fig.6.3. General Purpose Registers

Registers	EAX	EBX	ECX	EDX	ESI	EDI	EBP	ESP
General storage	✓	✓	✓	✓	✓	✓	✓	
String operations	✓							
Loop counter			✓					
I/O address				✓				
Multiply	✓	✓	✓	✓	✓	✓	✓	✓
Divide (dividend)	✓							
Divide (remainder)				✓				
Base register	✓	✓	✓	✓	✓	✓	✓	✓
Index register	✓	✓	✓	✓	✓	✓	✓	
XLAT pointer		✓						
I/O data	✓							
String source					✓			
String destination						✓		

Fig.6.4. Register usage

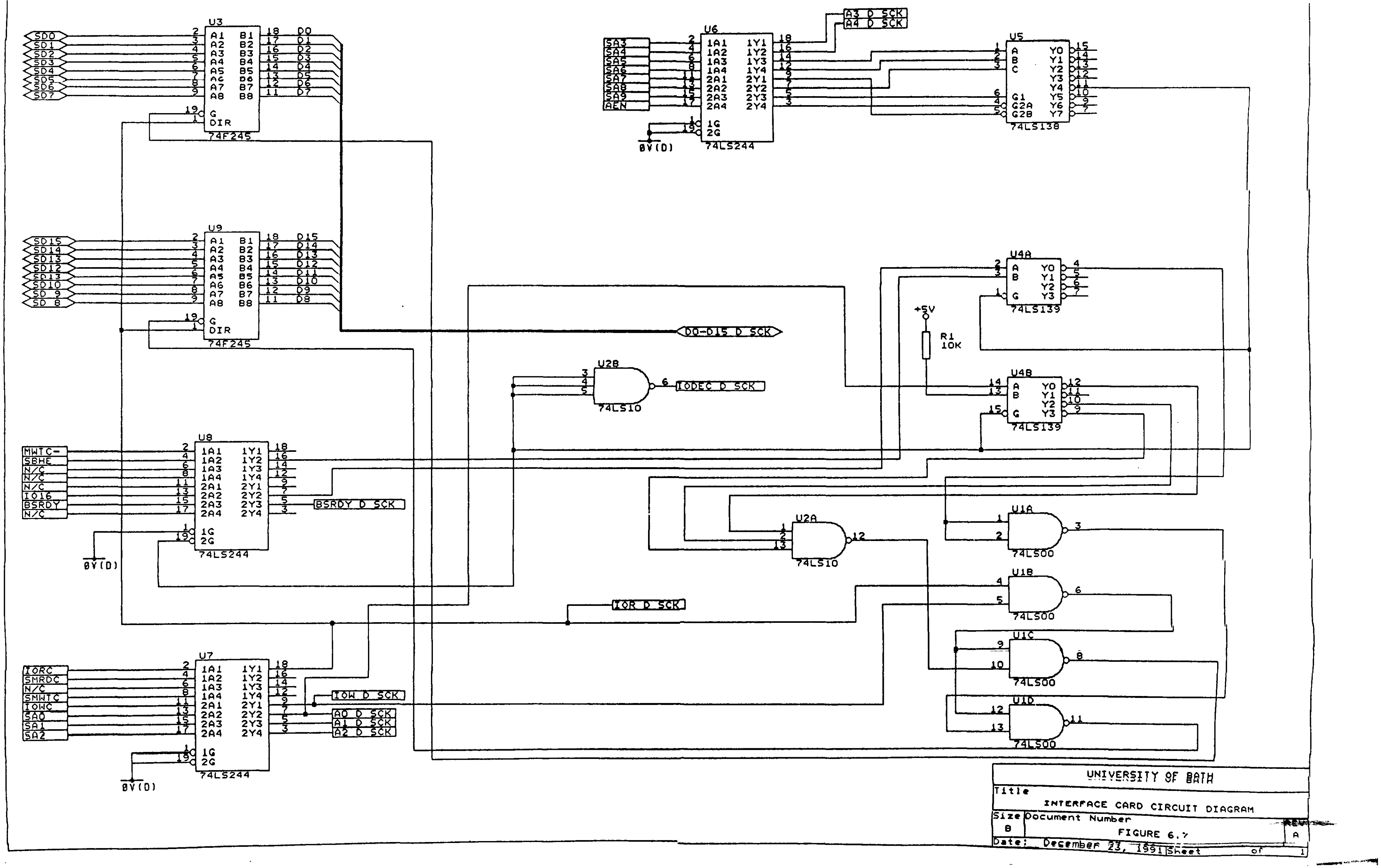
31–18	17	16	15	14	13–12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED FOR INTEL	VF	RF	0	NF	IOPF	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

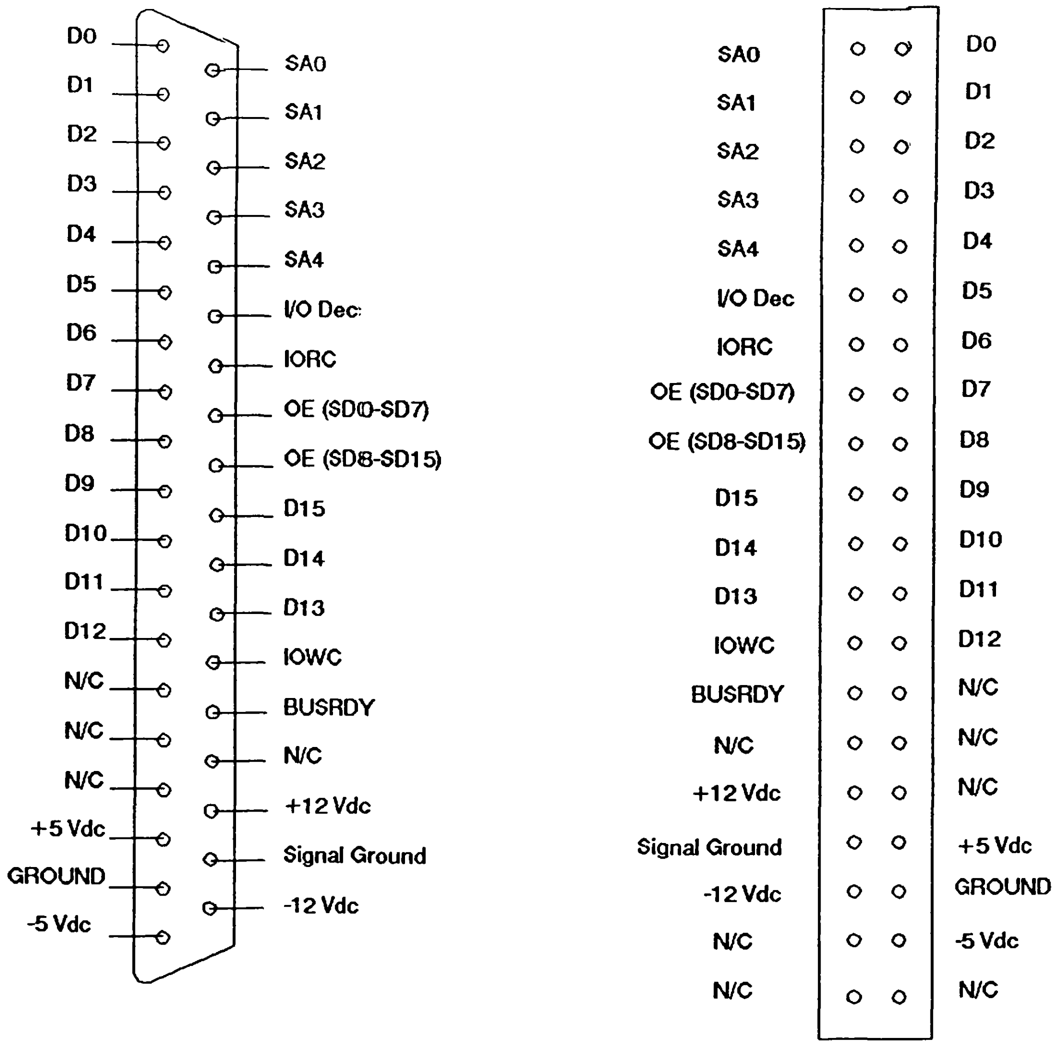
Fig.6.5. Flag Register

Signal	Pin	Pin	Signal
M16-	D01	C01	SBHE-
IO16-	D02	C02	LA23
IRQ10	D03	C03	LA22
IRQ11	D04	C04	LA21
IRQ12	D05	C05	LA20
IRQ15	D06	C06	LA19
IRQ14	D07	C07	LA18
DACK0-	D08	C08	LA17
DRQ0	D09	C09	MRDC-
DACK5-	D10	C10	MWTC-
DRQ5	D11	C11	SD8
DACK6-	D12	C12	SD9
DRQ6	D13	C13	SD10
DACK7-	D14	C14	SD11
DRQ7	D15	C15	SD12
+5 Vdc	D16	C16	SD13
GRAB-	D17	C17	SD14
SIGNAL GROUND	D18	C18	SD15

Signal			Signal
GROUND	B01	A01	IOCHK
RESDRV	B02	A02	SD7
+5 Vdc	B03	A03	SD6
IRQ9	B04	A04	SD5
-5Vdc	B05	A05	SD4
DRQ2	B06	A06	SD3
-12 Vdc	B07	A07	SD2
NOWS-	B08	A08	SD1
+12 Vdc	B09	A09	SD0
GROUND	B10	A10	BUSRDY
SMWTC-	B11	A11	AEN
SMRDC-	B12	A12	SA19
IOWC-	B13	A13	SA18
IORC	B14	A14	SA17
DAK3-	B15	A15	SA16
DRQ3	B16	A16	SA15
DAK1-	B17	A17	SA14
DRQ1	B18	A18	SA13
REFRESH	B19	A19	SA12
BCLK	B20	A20	SA11
IRQ7	B21	A21	SA10
IRQ6	B22	A22	SA9
IRQ5	B23	A23	SA8
IRQ4	B24	A24	SA7
IRQ3	B25	A25	SA6
DAK2-	B26	A26	SA5
T/C	B27	A27	SA4
BALE	B28	A28	SA3
+5 Vdc	B29	A29	SA2
OSC	B30	A30	SA1
SIGNAL GROUND	B31	A31	SA0

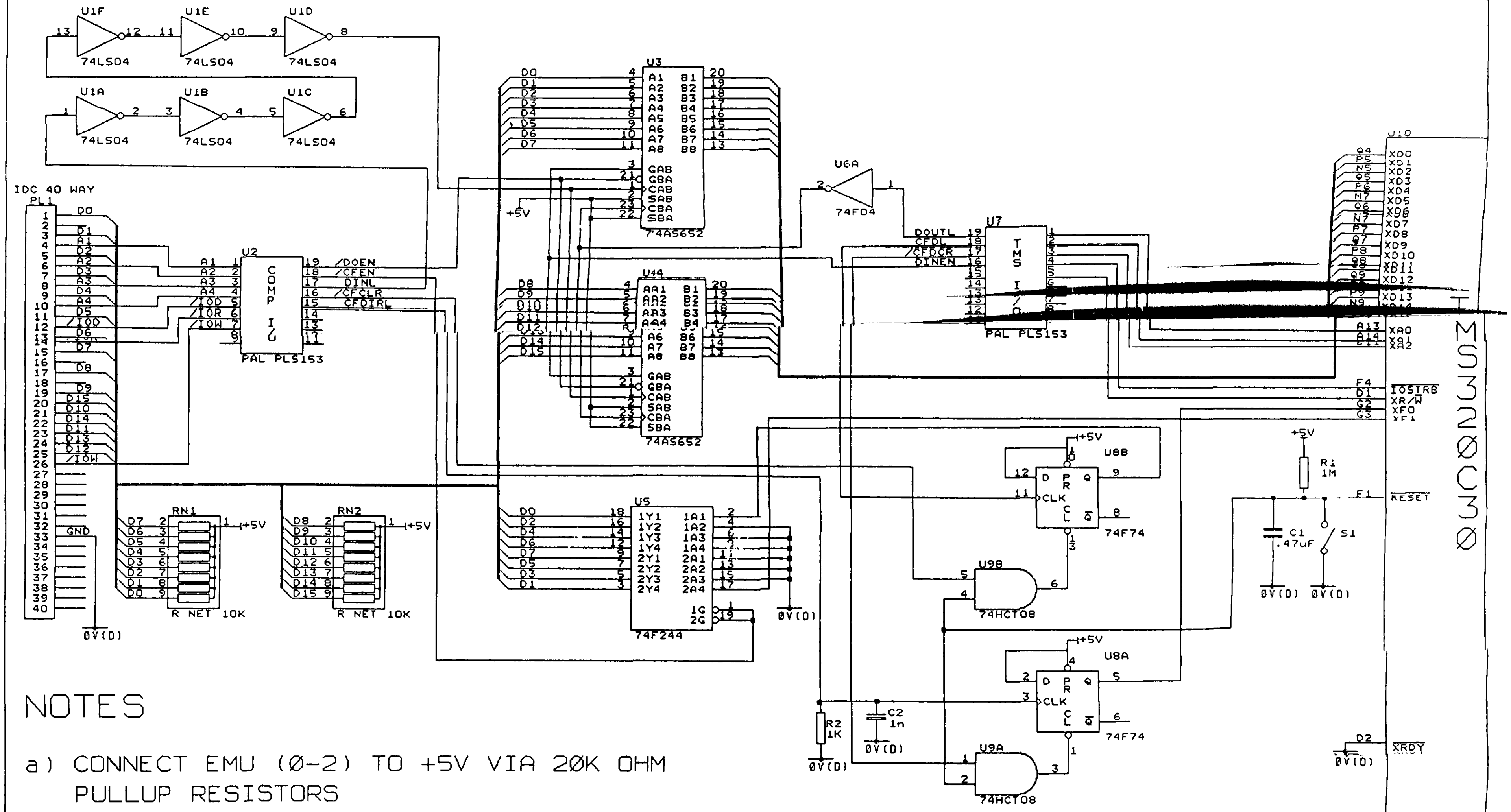
Fig.6.6. 16-Bit Expansion Bus Connector


U3
74F245
SD0
SD1
SD2
SD3
SD4
SD5
SD6
SD7
D0
D1
D2
D3
D4
D5
D6
D7
U9
74F245
SD15
SD14
SD13
SD12
SD11
SD10
SD 9
SD 8
D15
D14
D13
D12
D11
D10
D9
D8
D0-D15 D SCK
U6
74LS244
SA3
SA4
SA5
SA6
SA7
SA8
SA9
AEN
A3 D SCK
A4 D SCK
0V(D)
U5
74LS138
U4A
74LS139
U4B
74LS139
+5V
R1
10K
U2B
74LS10
IODEC D SCK
U8
74LS244
MWTC-
SBHE
N/C
IO16
BSRDY
BSRDY D SCK
U2A
74LS10
U1A
74LS00
U1B
74LS00
U1C
74LS00
U1D
74LS00
IOR D SCK
U7
74LS244
IORC
SMRDC
SMWTC
IOWC
SA0
SA1
SA2
IOW D SCK
A0 D SCK
A1 D SCK
A2 D SCK
UNIVERSITY OF BATH
Title
INTERFACE CARD CIRCUIT DIAGRAM
Size B
Document Number
FIGURE 6.7
REV A
Date: December 23, 1991
Sheet 1 of 1



COMPAQ I/O D-type connector   Communication Interface connector

Fig.6.8. The Interface Connectors Pins



NOTES

a) CONNECT EMU (Ø-2) TO +5V VIA 20K OHM PULLUP RESISTORS

b) CONNECT RSV(Ø-1Ø) & EMU 4 DIRECTLY TO +5V

c) MC/MP SHOULD BE TIED LOW TO PLACE TMS320C30 IN MICROPROCESSOR MODE

UNIVERSITY OF BATH		
Title: TMS320C30 EXPANSION BUS AND INTERFACE		
Size: B	Document Number: FIGURE 6.9	REV: A
Date: December 23, 1991	Sheet	of 2

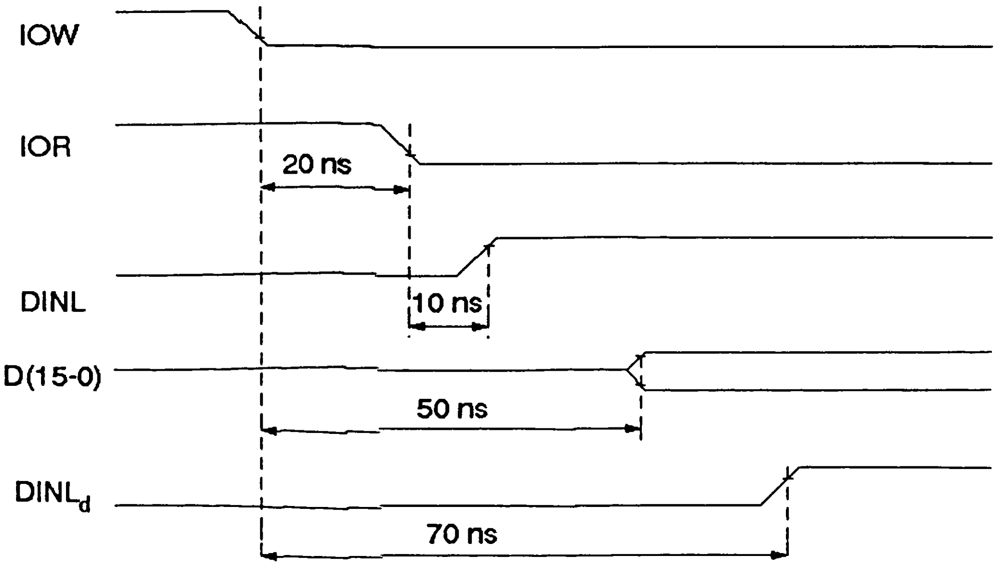


Fig.6.10. Transfer Signals from COMPAQ to Latch

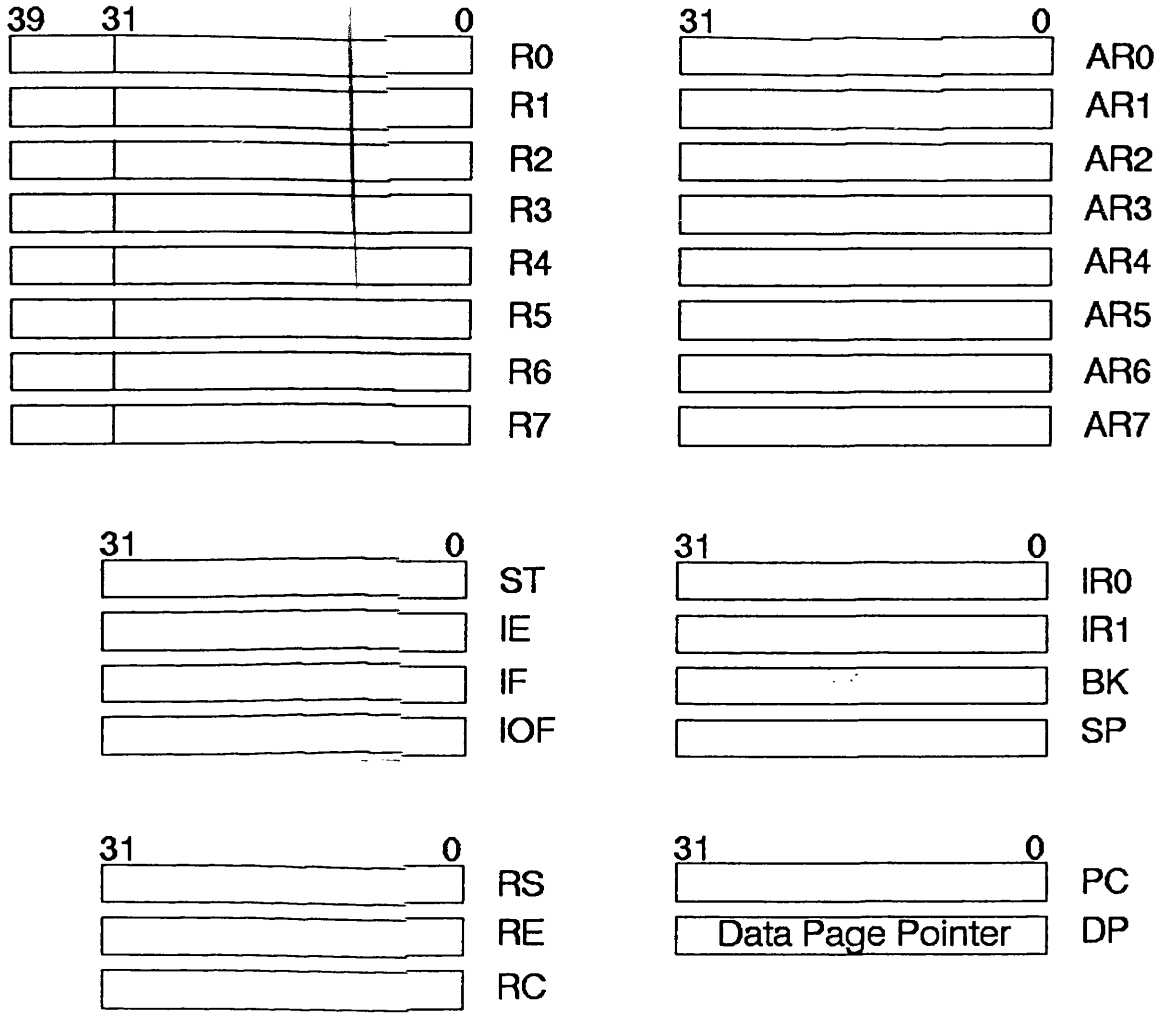


Fig.6.11. CPU Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	GIE	CC	CE	CF	xx	RM	OVM	LUF	LV	UF	N	Z	V	C
		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = reserved bit
R = read, W = write

Fig.6.12. Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	EDINT1 (DMA)	ETINT1 (DMA)	ETINT0 (DMA)	ERINT1 (DMA)	EXINT1 (DMA)	ERINT0 (DMA)	EXINT0 (DMA)	EINT3 (DMA)	EINT2 (DMA)	EINT1 (DMA)	EINT0 (DMA)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	xx	xx	xx	EDINT1 (CPU)	ETINT1 (CPU)	ETINT0 (CPU)	ERINT1 (CPU)	EXINT1 (CPU)	ERINT0 (CPU)	EXINT0 (CPU)	EINT3 (CPU)	EINT2 (CPU)	EINT1 (CPU)	EINT0 (CPU)
		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = reserved bit, read as 0
R = read, W = write

Fig.6.13. CPU/DMA Interrupt Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	xx	xx	xx	DINT1	TINT1	TINT0	RINT1	XINT1	RINT0	XINT0	INT3	INT2	INT1	INT0
		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = reserved bit, read as 0

R = read, W = write

Fig.6.14. CPU Interrupt Flag Register IF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	xx	xx	xx	xx	xx	xx	INXF1	OUTXF1	I/OXF1	xx	INXF0	OUTXF0	I/OXF0	xx
		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = reserved bit, read as 0

R = read, W = write

Fig.6.15. I/O Flag Register IOF

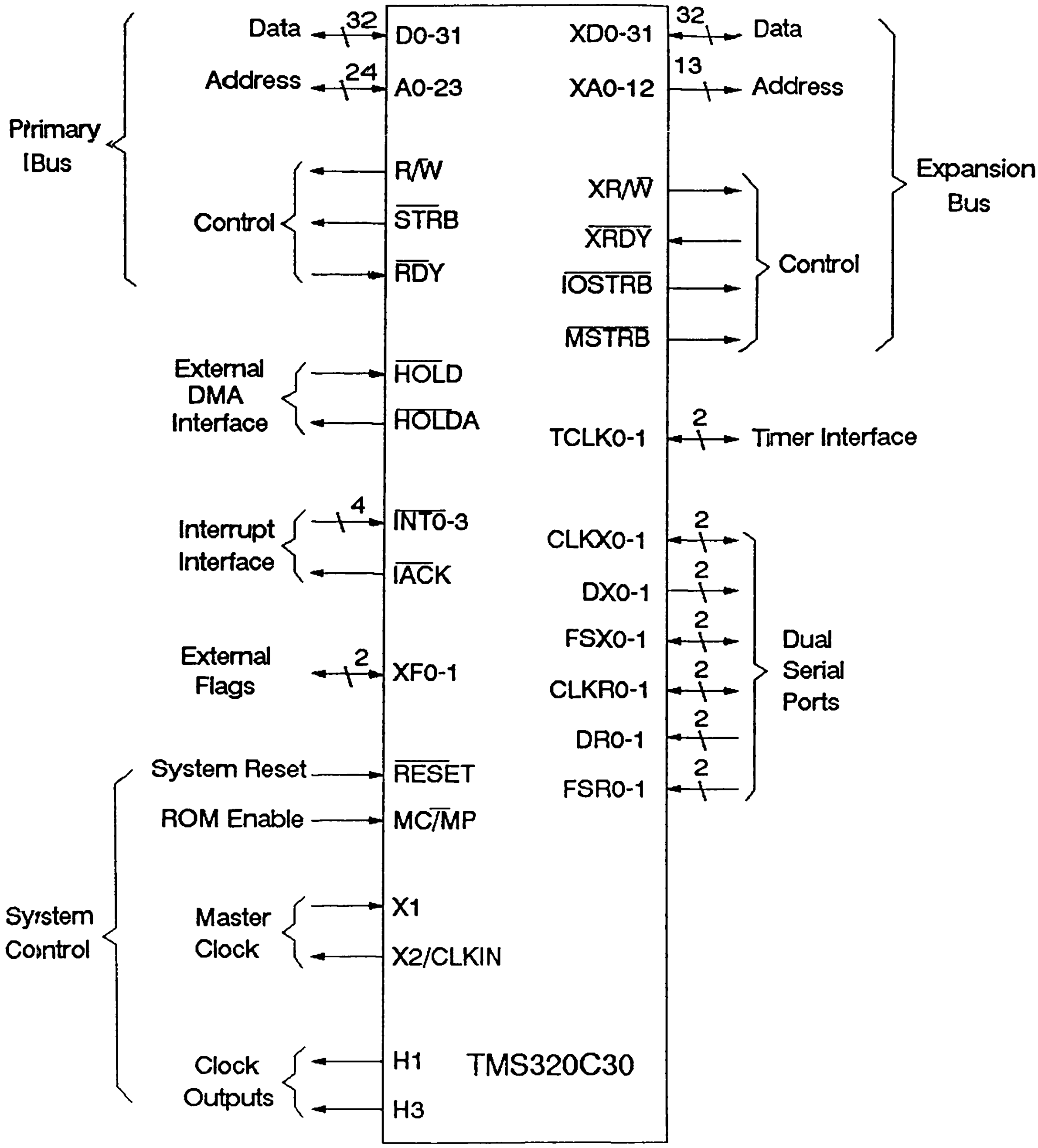
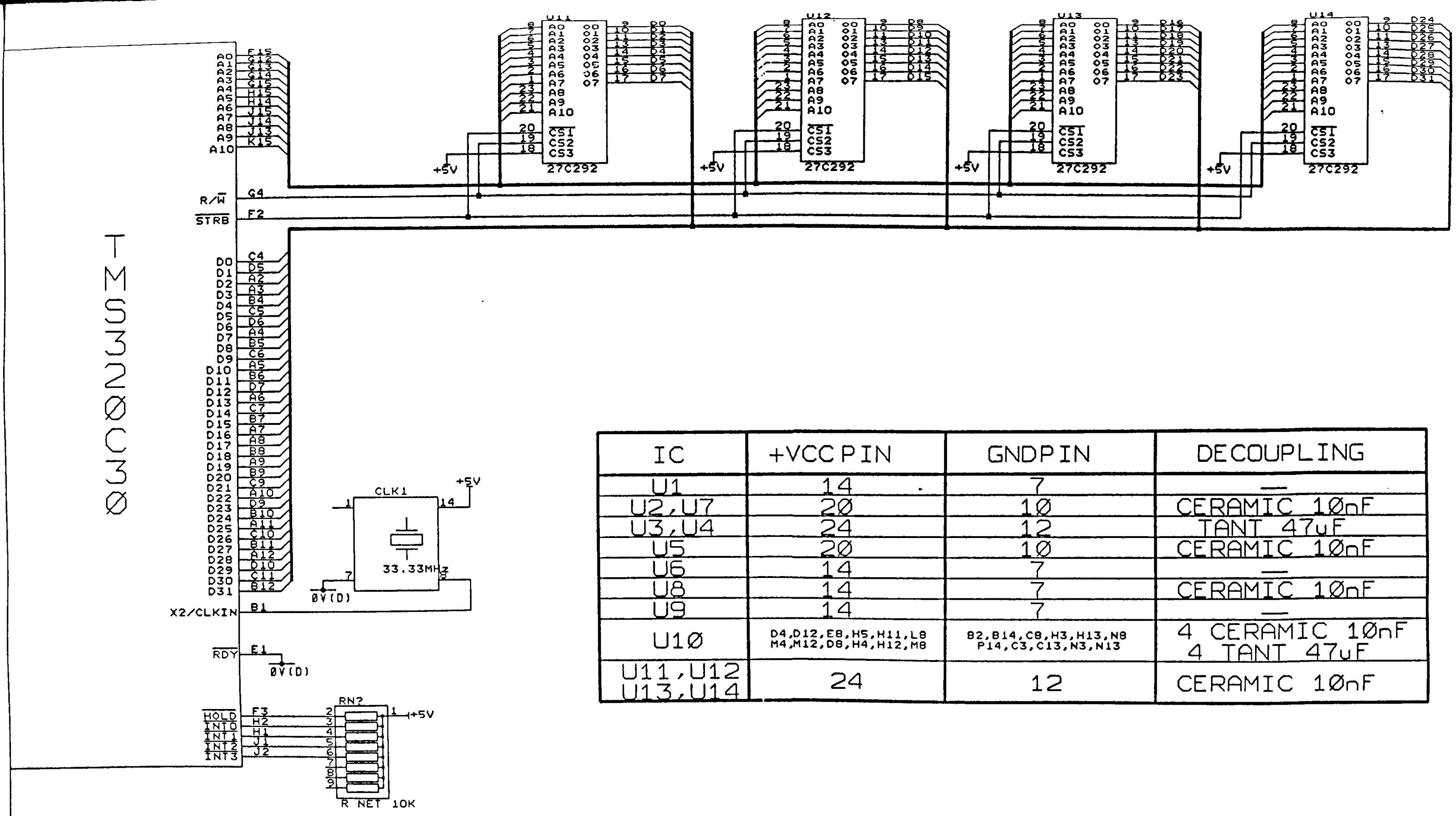


Fig.6.17. External Interfaces on the TMS320C30



IC	+VCC PIN	GNDPIN	DECOUPLING
U1	14	7	—
U2,U7	20	10	CERAMIC 10nF
U3,U4	24	12	TANT 47uF
U5	20	10	CERAMIC 10nF
U6	14	7	—
U8	14	7	CERAMIC 10nF
U9	14	7	—
U10	D4,D12,E8,H5,H11,L8 M4,M12,D8,H4,H12,M8	B2,B14,C8,H3,H13,N8 P14,C3,C13,N3,N13	4 CERAMIC 10nF 4 TANT 47uF
U11,U12 U13,U14	24	12	CERAMIC 10nF

UNIVERSITY OF BATH

Title: TMS320C30 PRIMARY BUS

Size: B | Document Number: FIGURE 6.18 | REV: A

Date: December 23, 1991 | Sheet of

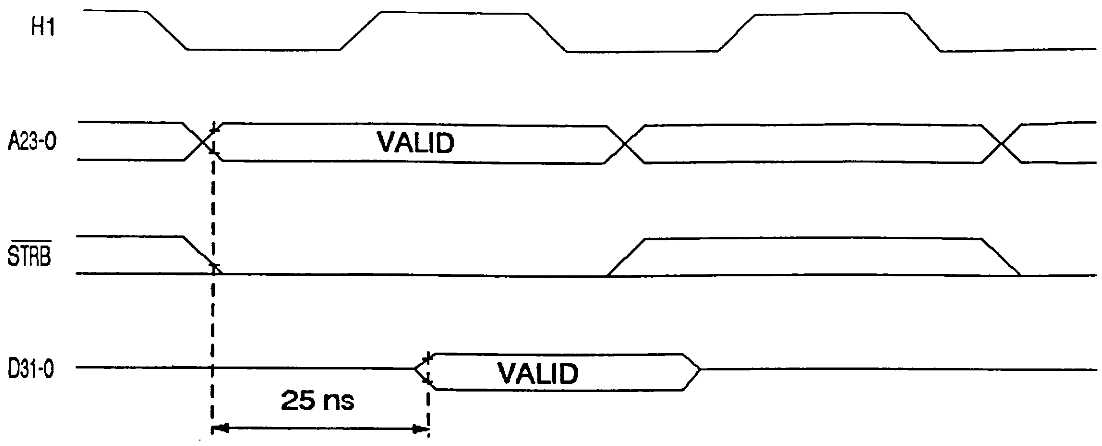


Fig.6.19. The Primary Bus Read Operation Timing

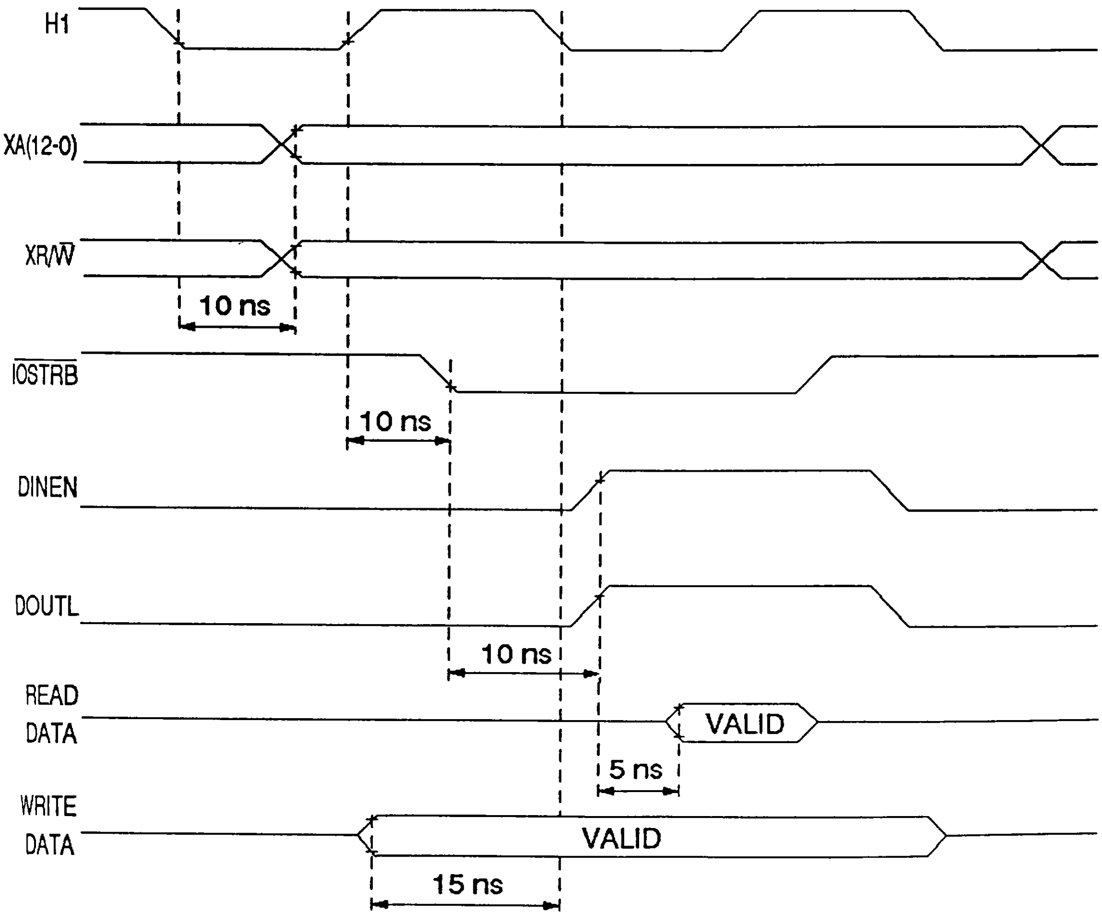
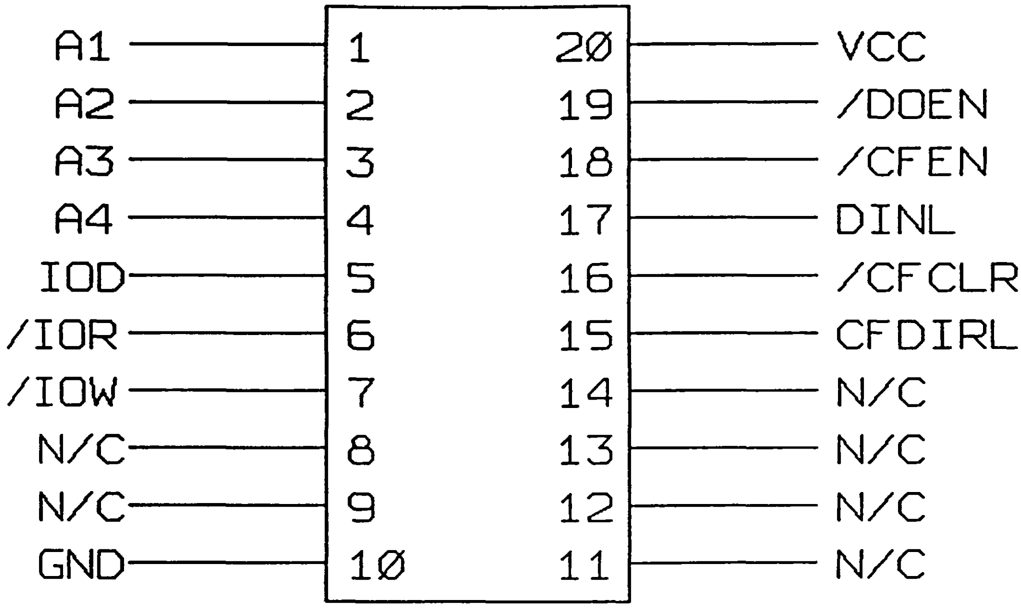


Fig.6.20. Expansion Bus Interface to Latch

COMPIO PAL



LOGIC EQUATIONS

/DOEN=/(/A1 * /A2 * /A3 * /A4 * IOD * IOR)

/CFEN=/(A1 * /A2 * /A3 * /A4 * IOD * IOR)

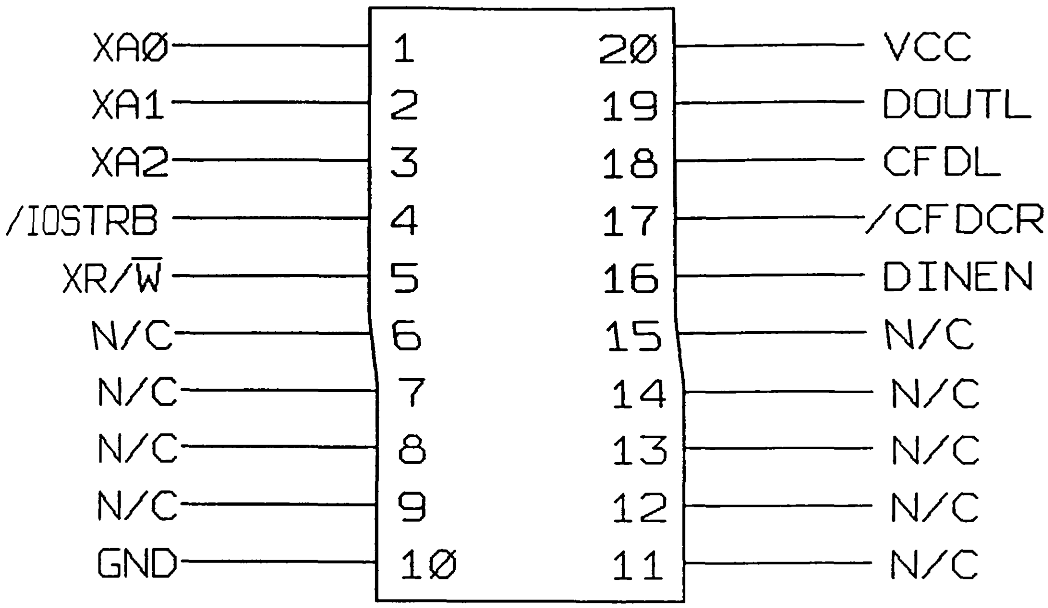
DINL=/A1 * /A2 * /A3 * /A4 * IOD * IOW

/CFCLR=/(A1 * /A2 * /A3 * /A4 * IOD * IOW)

CFDIRL= /A1 * A2 * /A3 * /A4 * IOD * IOW

UNIVERSITY OF BATH		
Title COMPIO VERSION 1.0 PAL		
Size A	Document Number FIGURE 6.21	REV A
Date: January 16, 1992	Sheet	of

TMSIO PAL



LOGIC EQUATIONS

DOUTL=/XAØ * /XA1 * /XA2 * IOSTRB * /XR/$\overline{W}$

CFDL =XAØ * /XA1 * /XA2 * IOSTRB * /XR/$\overline{W}$

CFDCR=/(/XAØ * XA1 * /XA2 * IOSTRB */XR/$\overline{W}$)

DINEN=/XAØ * /XA1 * /XA2 * IOSTRB * XR/$\overline{W}$

UNIVERSITY OF BATH		
Title TMSIO VERSION 1.0 PAL		
Size A	Document Number FIGURE 6.22	REV A
Date: December 23, 1991	Sheet 1 of	

CHAPTER 7. THE SOFTWARE OF THE IDENTIFICATION

7.1. Introduction

Two different processors work for the application of the identification as discussed in previous chapters. Their softwares are different as well as their hardwares, which were explained in the previous chapter. Programs run independently except during the communication. The flow charts of the programs are given in Fig.7.1.. They represent all different options, which have been given in chapter 5. They are explained in detail in this chapter.

7.2. Personal Computer Software.

In the P.C. flow chart blocks 1, 3, 5 and 6 are operated with a high level language FORTRAN 77. Blocks 2 and 4 are processed using assembly language, directly, because, addresses of the I/O port cannot be accessed by standard FORTRAN 77 . The assembly code can be called from a high level language as a subroutine. So, the flight modelling with a high level language and the communication with low level language have been achieved in the P.C.. Block 6 has been realized due to Eq.5.7. by fourth order Runge-Kutta integration routine. Blocks 2 and 4 can be done according to a protocol, which will be given in the next section. The Runge-Kutta algorithm is written in FORTRAN 77 . The assembly language codes relating to the read-write are given in appendix A.1.1. and A.1.2.. It has been mentioned that the P.C. uses 80386 processor, but its 16-bit I/O expansion bus is compatible with the 80286 as discussed the section 6.2..

N value of the block 1 has been described due to the kind of parameter value. Its value has been taken to be equal to 3 for the noise-free system. It has been increased with noise level. The results of identification for N=30 and N=60 were given in chapter 5. Before the Nth step there is no response from the identification board, therefore there is no reading. However, before the Nth step, the program writes the results into a file where all parameters are equal to 1. The flow chart 7.1. can be applied with another approach that P.C. reads data from the ready data file instead of calculating each step values, Because, the data creation and communication in 40 msec, which is the step time, may not be possible. That is if the COMPAQ is to slow in similarity the aircraft.

7.3. Communication Protocol

The communication between the 80386 and TMS320C30 is performed via the latch circuit and flags as discussed before. This communication can only be conducted in one direction at any time. The external flag of TMS320C30 (XF1) determines the direction of the communication. Both directions protocols are given below:

```
  COMPAQ         -->        TMS320C30
             (XF1=0)
Write 0300H,data
Write 0304H,'1'
                            Read XF0
                            Read   804000h,data
                            Write  804002h,clear flag
Set flag 1                  Write  804001h
Write 0302h,clear flag
return start
```

```
 TMS320C30       -->       COMPAQ
             (XF1=1)
                           Read  0302h
                           BIT 1 = 0    COMPAQ --> C30
                           BIT 1 = 1    C30 --> COMPAQ
Write 804000h,data
(load latch)
Write 804001               Set flag 1
                           (checked by read 0302h
                             BIT 0 )
                           Read 0300h,data
                           Write 0302h
                           (clear flag)
                           Write 0304h
                           (set C30 flag)
Read XF0
When 'set' data has
been read read by compaq
write 804002h
(clear flag)
```

When data is written on the 300h address line by COMPAQ, the COMPIO PAL decoder, which was shown in the previous chapter, produces the latch trigger signal. So data is written the latch circuits. Then the COMPAQ sends any data to 302h address line, which causes the XF0 external bit of TMS320C30 to be high. After XF0=1, the TMS320C30 reads data from the latch via the 804000h address line. Then the TMS32030 sends data to 804002h address line to clear XF0 signal. Later, it writes any data to 804002h address line to send an acknowledgement to the COMPAQ. This signal causes the bit 0 of the 302h address of COMPAQ to be equal to 1. When the COMPAQ senses the acknowledgement, it continues to the next operation. However, the COMPAQ checks the XF1 before each operation. The TMS320C30 changes XF1 to start a write operation. The TMS320C30 writes to the 804000h address line to transfer data to the latch circuit. Simultaneously, the TMSIO PAL decoder which was shown in the previous chapter, produces the latch trigger signal. The TMS320C30 writes any data to the 804001h address line to alert the COMPAQ. In this condition, 0003h is shown on the 302h address line of COMPAQ. Bit 0 and bit 1 represent the data ready flag and the direction flag, respectively. The COMPAQ reads data from 300h address line after the data ready flag. Then, it writes data to clear the data ready flag. Finally, it writes the data to send the acknowledgement to the TMS302C30, which causes to XF0=1. Then, the TMS320C30 clears the direction flag via the 804002h address line.

7.4. Read and Write Operation with P.C.

The transfer values from the P.C. to the board have been in real mode. They are in 32-bit floating point format, but I/O bus is only 16 bits . Therefore, 32-bit floating point value is assumed to be

32-bit integer value and transferred as two 16-bit. For this purpose, initially, the starting address of the variable is determined. Subsequently, low significant 16-bit of 32-bit is sent. After the sensing of acknowledgement, it is cleared and then high significant 16-bit of 32-bit is sent. So, the 32-bit data is written or transferred to TMS320C30 board.

Reading of the data also follows the same procedure. For this case the OUT command is replaced by the IN command with appropriate control signals. The Read and Write operation program is given in appendix with GO and COMEP names, respectively.

7.5. TMS320C30 Software

The TMS320C30 software is much more complex than the P.C.'s I/O software. This program controls all the board operations, initialization of the processor, identification operation, I/O operation and the data conversation and acquisitions. Each operation control will be given in the following sub-sections.

7.5.1. The Initialization of the TMS320C30

The processor should be initialized with a reset operation before the start of the execution of the program. When reset is activated, the TMS320C30 DSP goes to reset vector, which is the contents of the memory location 0. Firstly, the starting address of the initialization program should be described. Secondly, memory mapped registers [46] and interrupt structure should be initialized. Then, all memory must be filled with zero and the operation program variables must be defined. In this work, the initialization has been programmed so that the processor reaches the operation program via the primary bus and the I/O via the expansion bus. The processor uses only the on-chip memory for storage. The interrupts have not been considered because of the communication protocol. The initialization program is given in appendix A.1.3..

The reset vector is described in initialization as follow

```
        .sect     "init"      ; Named section
RESET   .word     INIT        ; RS- loads address INIT to PC
```

This description is not enough in order to point the vector address name. The address of the initialization program should be described with ROM address in the linker command file. The variables address, the stack space and data space should be written in the linker command file. An example is given below

```
/**********************************************************************
/*          SAMPLE COMMAND FILE FOR LINKING C30 PROGRAMS
/*
/* File Name: EDS.CMD
/**********************************************************************

EDS.OBJ

/* SPECIFY THE SYSTEM MEMORY MAP */

MEMORY
{
   VECS:    org = 0h        len = 040h       /* INTERRUPT VECTORS */
   ROM:     org = 0C0h      len = 07FFF40h   /* PROGRAM CODE      */
   IOM:     org = 0800000h len = 02000h      /* -MSTRB I/O        */
   IO:      org = 0804000h len = 02000h      /* -IOSTRB I/O       */
   RAM0:    org = 0809800h len = 0400h       /* RAM BLOCK 0       */
   RAM1:    org = 0809C00h len = 0300h       /* RAM BLOCK 1       */
   STACK:   org = 0809F00h len = 0100h       /* SYSTEM STACK      */
}




/* SPECIFY THE SECTIONS ALLOCATION INTO MEMORY */

SECTIONS
{
   vectors:  {} > VECS      /* INTERRUPT VECTORS                 */
   stk_init: {} > STACK     /* INITIAL STACK POINTER             */
   .text:    {} > ROM       /* CODE                              */
   .data:    {} > RAM1      /* ROM-RESIDENT TABLES & CONSTANTS   */
   .bss:     {} > RAM0      /* BSS DATA                          */
}
```

The variables, which are used by the program, are put into suitable locations of the memory. This operation is performed by the initialization program and the linker command file. At first, their location in the memory are declared . In the second part, which has been described in the RESET declaration by INIT section, the contents of the variables (initial values) are filled with the initialization program. The on-chip RAM of the processor is considered as memory which may not be sufficient unless the code is in an optimal form. For example, when N=60 is selected, the address space of data must be enlarged from 400h to 600h. Thus, the resident table and constants must be squeezed in the 100h memory space. Its linker command file is given in appendix A.1.4. The memory usage of the N=60 identification routine is also given in appendix A.1.5..

7.5.2. I/O Program

The processor can recognize the data by the XF0 external flag. It can be controlled by the IOF register. The data is assumed to be integer as in the other processor. Therefore, they are transferred to the temporary memories. A typical reading program is given below:

```
READ    LDI  @PRLIO,AR1    ; Load the device address to AR1
        LDI  @PRLIOO,AR2   ; Load the temporary address to AR2
        LDI  20H,IOF       ; Configure the external flags
        LDI  5,RC          ; Load the data number
        RPTB DV            ; Start to read
XF0     LDI  IOF,R1        ; Check the IOF register
        AND  8,R1          ;
        BZ   XF0           ; Wait until data is ready
        LDI  *AR1,R2       ; Read data
        STI  R2,*AR2++(1)  ; Store data
        STI  R0,*+AR1(2)   ; Clear flag
DV      STI  R0,*+AR1(1)   ; Send acknowledgement and wait new data
        RETS               ; Return to main program.
```

Data length is 16-bit during the transfer. It will be converted to

32-bit floating point TMS format. The name of this program is DMAA in the main program, which is given in appendix A.1.13.

Writing by the processor is similar to reading. The 32-bit floating point values are converted to 16-bit integer values and stored in other temporary memories before writing. This program sends only two 16-bit integer data. Its source assembly form is given below:

```
WRITE   LDI  @PRLIO,AR1     ; Load the device address to AR1
        LDI  @OUTPUT,AR2    ; Load the data address to AR2
        LDI  60H,IOF        ; Configure the external flags
        LDI  *AR2++(1),R2   ; Load data to register
        STI   R2,*AR1       ; Send data( low 16-bit) to latch
        STI  R1,*+AR1(1)    ; Send the flag to P.C.
CV      LDI  IOF,R1         ; Check acknowledgement
        AND  8,R1           ;
        BZ   CV             ; Wait acknowledgement
        STI  R0,*+AR1(2)    ; Clear flag
        LDI  *AR2++(1),R2   ;
        STI   R2,*AR1       ; Send high 16-bit
        STI  R1,*+AR1(1)    ;
CV1     LDI  IOF,R1         ;
        AND  8,R1           ;
        BZ   CV1            ;
        STI  R0,*+AR1(2)    ;
        LDI  R0,IOF         ;
        RETS                ; Return to main program
```

7.5.2.1. Floating Point Conversion

The TMS320C30 floating point format is different from P.C. floating point format as mentioned before. The TMS320C30 floating point format is:

8	1	23
e	*s*	*f*

The first 8 bits correspond to the exponent, which is expressed in two's complement format. The following one bit is for the sign of mantissa and the 23 bits are for mantissa itself. Actually, the sign bit includes the information for the mantissa, therefore the mantissa is represented by 24 bits.

$2^{e^*}(01.f)$ if s=0

$2^{e^*}(10.f)$ if s=1

0 if e=-128.

For example, 1 is described by 00000000h and -1 is described by FF000000h.

The P.C. uses the IEEE floating point format, which is as:

1	8	23
s	*e*	*f*

The first bit of data is sign bit, the following 8-bit is exponent and the remaining 23-bit is the mantissa. In this format, the mantissa is represented by 24-bit. Because the integer part of mantissa is assumed to be 1. 23 -bit represents only fractional part. So floating point number is:

$$s * 2^{e-127} * (1.f)$$

where s represents only the sign of the mantissa. The exponent value is shifted with 127 to be actual value. In this case minimum exponent is 0, maximum exponent is 7Fh.

IEEE to TMS320C30 floating point format conversion and TM320C30 to IEEE floating point format conversion assemble source programs are given in appendix A.1.6. and A.1.6. by TMSC and CMPQ names.

7.5.3. The Identification Program

The identification operation consists of complex matrix operations. Therefore, all matrix operations in the assemble source should be defined. The matrix addition and multiplication, the inverse of the floating point number, the integration and the inverse of the matrix are given with detail, in the following sub-sections.

7.5.3.1. Matrix Addition and Subtraction

It is very well known that only matrices of the same dimension can be added or subtracted. The assemble source program is a low level program, therefore checking of the dimension is made by the programmer. A matrix addition can be represented as

$$C(i,j) = A(i,j) + B(i,j) \quad i = 1,2,\ldots,P \ , \ j = 1,2,\ldots,N$$

$$\begin{bmatrix} a(1,1) & a(1,2) & \ldots & a(1,N) \\ a(2,1) & a(2,2) & \ldots & a(2,N) \\ \vdots & & & \vdots \\ a(P,1) & a(P,2) & \ldots & a(P,N) \end{bmatrix} + \begin{bmatrix} b(1,1) & b(1,2) & \ldots & b(1,N) \\ b(2,1) & b(2,2) & \ldots & b(2,N) \\ \vdots & & & \vdots \\ b(P,1) & b(P,2) & \ldots & b(P,N) \end{bmatrix} =$$

$$\begin{bmatrix} c(1,1) & c(1,2) & \ldots & c(1,N) \\ c(2,1) & c(2,2) & \ldots & c(2,N) \\ \vdots & & & \vdots \\ c(P,1) & c(P,2) & \ldots & c(P,N) \end{bmatrix}$$

We must define the address of each element of matrices in the programming. If we assume the starting addresses of *xx*, *yy* and *zz* for the matrices A, B and C, respectively, the addresses other elements are as follows:

xx	$\leftrightarrow$ a(1,1),	yy	$\leftrightarrow$ b(1,1)
xx + (N-1)	$\leftrightarrow$ a(1,N),	yy + (N-1)	$\leftrightarrow$ b(1,N)
xx + N	$\leftrightarrow$ a(2,1),	yy + N	$\leftrightarrow$ b(2,1)
xx + N*(i-1)	$\leftrightarrow$ a(i,1),	yy + N*(i-1)	$\leftrightarrow$ b(i,1)
xx + N*(i-1) + (j-1)	$\leftrightarrow$ a(i,j),	yy + N*(i-1) + (j-1)	$\leftrightarrow$ b(i,j)
xx + (P*N)-1	$\leftrightarrow$ a(P,N),	yy + (P*N)-1	$\leftrightarrow$ b(P,N)

zz	$\leftrightarrow$ c(1,1)
zz + (N-1)	$\leftrightarrow$ c(1,N)
zz + N	$\leftrightarrow$ c(2,1)
zz + N*(i-1)	$\leftrightarrow$ c(i,1)
zz + N*(i-1) + (j-1)	$\leftrightarrow$ c(i,j)
zz + (P*N)-1	$\leftrightarrow$ c(P,N)

c(i,j) = a(i,j) + b(i,j) and their addresses are zz + N*(i-1) +(j-1), xx + N*(i-1) +(j-1) and yy + N*(i-1) +(j-1), respectively. It is easily seen that all addresses are identical. They can be reached from the starting addresses by the same number of shifting. Last addresses are found with shifting (P*N)-1. In this case, each element of C matrix can be calculated and placed in their correct location. The related assemble program is given in appendix A.1.8.. It can be shown that each element of matrices is used only one time during the operation. Therefore, if A or B matrix is not needed any more, the memory space for the C matrix is not required. This is an advantage for the restricted memory usage. This facility has been used in the main program.

The matrix addition program was used for the matrix subtraction. The Subtraction command is used instead of addition command via checking the control flag, which is MINUS variable in the program.

7.5.3.2. Matrix Multiplication

Matrix addition and required memory arrangements have been shown in the previous section. But multiplication is more complex than addition. A matrix multiplication is defined as:

$$A(P,N) * B(N,R) = C(P,R)$$

$$c(i,j) = \sum_{k=1}^{N} a(i,k)*b(k,j)$$

N times scalar multiplications are necessary to obtain each element of the resultant matrix. Calculating an element of the resultant matrix is explained by an example as follows:

Step 1) Set $c(i,j)$ to zero

Step 2) Find the starting addresses of elements of matrices A and B, which are the first element of ith line of A and the first element of jth colon of B.

Step 3) Multiply $a(i,k)$ by $b(k,j)$ and add result to $c(i,j)$.

Step 4) Increase the memory address of the element of A matrix by one. Increase the memory address of the element of B by R.

Step 5) Go to step 2. Continue this operation until Nth step.

These steps are applied for all elements of the resultant matrix. Its assemble source program is given in appendix A.1.9..

7.5.3.3. The Inverse of Floating Point Number

The TMS320C30 processor does not have a division command. The integer division operation can be performed by using the Subtract Integer Conditionally command. A software for the inverse of the floating point number is developed in reference [46]. This program is based on the following iterative algorithm. At the *i*th iteration, the estimate $x(i)$ of $1/v$ is computed from v, and the previous estimate of $x(i-1)$ according to the formula:

$$x(i) = x(i-1) * [\ 2.0 - v * x(i-1)\ ]$$

An initial estimate $x(0)$ is required to start the operation. $v=a*2^{e}$ is given a good initial estimate in [46]:

$$x(0) = 1.0 * 2^{-e-1}$$

In this algorithm, the accuracy of $2^{-23}=1.192e-7$ can be achieved wih 5 iterations. 10 times iterations are applied in this study.

The full assemble code is given in appendix A.1.10..

7.5.3.4. The Integration Algorithm

The particular system and the homogeneous systems modelling need an integration operation. The Runge-Kutta algorithm, which is well known and is most widely used, is programmed for integration in this study. This algorithm calculates the next values of the variable of of the differential equation from the previous values. The algorithm is given as:

$$\frac{d}{dt} x(t) = f(x, t)$$

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(x_n, t_n)$$

$$k_2 = hf(x_n + 0.5k_1,\ t_n + 0.5h)$$

$$k_3 = hf(x_n + 0.5k_2,\ t_n + 0.5h)$$

$$k_4 = hf(x_n + k_3,\ t_n + h)$$

All parameters are assumed to be constant between two consecutive steps. The integration program was written as a subroutine. The system function was considered as:

$$\dot{x} = Ax + Bu$$

$$\dot{x} = Ax + C \qquad\qquad C = Bu$$

The beginning addresses of matrices x, A and C were stored in the variables VAR, PAR and CONT before calling the integration subroutine. In the first step, the values of $\dot{x}$ vector are obtained by calling the matrix multiplication and matrix addition subroutines. k_1 vector, which is called by MK1, is obtained by multiplying $\dot{x}$ vector with the step time. Then, MK1 is divided by 2 and added to the variable vector. In the following steps, MK2, MK3, MK4 are solved. The next step variable vector is found as

$$x_{N+1} = x_N + 0.1666 * (MK1 + 2*MK2 + 2*MK3 + MK4)$$

This subroutine is called for three different systems which are the response to the of actual system input, the particular system and the homogeneous systems. The assemble program of all this operation is given in appendix A.1.11.

7.5.3.5. The Matrix Inversion

Matrix inverse is done with using the Gauss-Jordan elimination algorithm. The *nxn* unity matrix is added to original matrix right side. So, the matrix becomes to *n* x *2n* matrix. When the left *nxn* part of the matrix is converted to identity matrix via elimination, the right *nxn* part of the matrix becomes the inverse of the original matrix [47]. The Gauss -Jordan scheme is given by following routine:

Step 1) i is selected 1 to start.

Step 2) Rows are interchanged to make the value of a_{ii} the largest magnitude of any coefficient in the ith column. New row is divided by a_{ii}.

Step 3) All other values of the ith column are made zero by subtracting a_{ji} / a_{ii} times the ith row from the jth row. j is changed from 1 to n except i.

Step 4) Increase the i and go to step 2.

Step 5) Continue this operation until i reaches to N.

In step 2, the rows that are before ith row, are not considered to interchange.

The n x $2n$ space is needed to use this algorithm. In the programming, ith column is changed with ith column of the identity matrix after step 3 as the identity matrix is not considered to interchange. Only I_{ii} is 1 and other elements are zero in the identity matrix. Therefore, the space of the unity matrix is not needed to share in the memory. Finally, the columns are interchanged according to the rows' interchanging. So, the inverse operation can be achieved by nxn space.

The inverse of the matrix A can be solved with an assembly source program. In the first step, some auxiliary variable vectors which are IPIVOT(N) and INDEX(N,2) are described for control and index. Their values are set to zero. IPIVOT(I) describes whether the ith row has already been interchanged. INDEX(I,1) and INDEX(I,2) describe the maximum coefficient's row and column numbers in for *i*th column. The maximum coefficient of *i*th column is found without using interchanged rows. The maximum coefficient's row number and the value of *i* are noted to INDEX(I,1) and INDEX(I,2). IPIVOT(I) value is increased by one. The row which includes the maximum coefficient in the *i*th column is interchanged with the *i*th row. In this case, a(i,i) becomes the maximum coefficient. a(i,i) is stored to PIVOT variable. In here, *i*th column is thought as *i*th column of the identity matrix. Therefore, a(i,i) is taken to equal 1. Then all elements of row are divided by PIVOT. In the next step, the *i*th element of *j*th row a(j,i) is stored to a temporary address and a(j,i) is taken to equal zero, because all other elements of *i*th column of identity matrix are zero. Then, the temporary value (the old value of a(j,i)) times *i*th row is subtracted from *j*th row. It corresponds to *Step 3 of* Gauss elimination. This operation is performed for all rows except *i*th rows. Thus, one column operation is finished and the next column is started. The maximum coefficient is found by starting from (i+1)th row. This elimination is performed for all columns. The resultant matrix is the inverse of the original matrix but it still needs to be interchanged, because, some rows have been interchanged. Interchangings were not considered, when the *i*th column was thought as the *i*th column of the identity matrix. INDEX vector determined which rows are interchanged. The columns are interchanged according the rows interchanging. Hence, the inverse of matrix is obtained with nxn space. Its assembly source program is given in appendix A.1.12.

7.5.3.6. Flow Chart of Identification Program

The flow chart of the operation is given in Fig.7.1.. Initially, the input data is read. Then the control input is delayed via integration program. Subsequently, the homogeneous systems' control inputs are obtained from the input data. Then, the particular and the homogeneous systems outputs are found for the next step by using the Runge-Kutta subroutine. Program returns to the input. The same routine is repeated until the third step. Then the homogeneous outputs are collected in the general homogeneous system matrix $h(t_i)$ as seen below.

$$\begin{bmatrix} h_{11}(t) & h_{21}(t) & h_{31}(t) & h_{41}(t) & h_{51}(t) \\ h_{21}(t) & h_{22}(t) & h_{32}(t) & h_{42}(t) & h_{52}(t) \\ h_{11}(t+1) & h_{21}(t+1) & h_{31}(t+1) & h_{41}(t+1) & h_{51}(t+1) \\ h_{21}(t+1) & h_{22}(t+1) & h_{32}(t+1) & h_{42}(t+1) & h_{52}(t+1) \\ h_{21}(t+2) & h_{22}(t+2) & h_{32}(t+2) & h_{42}(t+2) & h_{52}(t+2) \end{bmatrix}$$

The transpose of the homogeneous system $[h(t_i)]^T$ are obtained. Then the last three step input data are collected in BTI matrix.

$$
\mathrm{BTI} = \begin{bmatrix} b_1(t) \\ b_2(t) \\ b_1(t+1) \\ b_2(t+1) \\ b_2(t+2) \end{bmatrix}
$$

The particular outputs are collected in PTI matrix as BTI.

$$
\mathrm{PTI} = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_1(t+1) \\ p_2(t+1) \\ p_2(t+2) \end{bmatrix}
$$

Eq(5.26) is rewritten to guide to program:

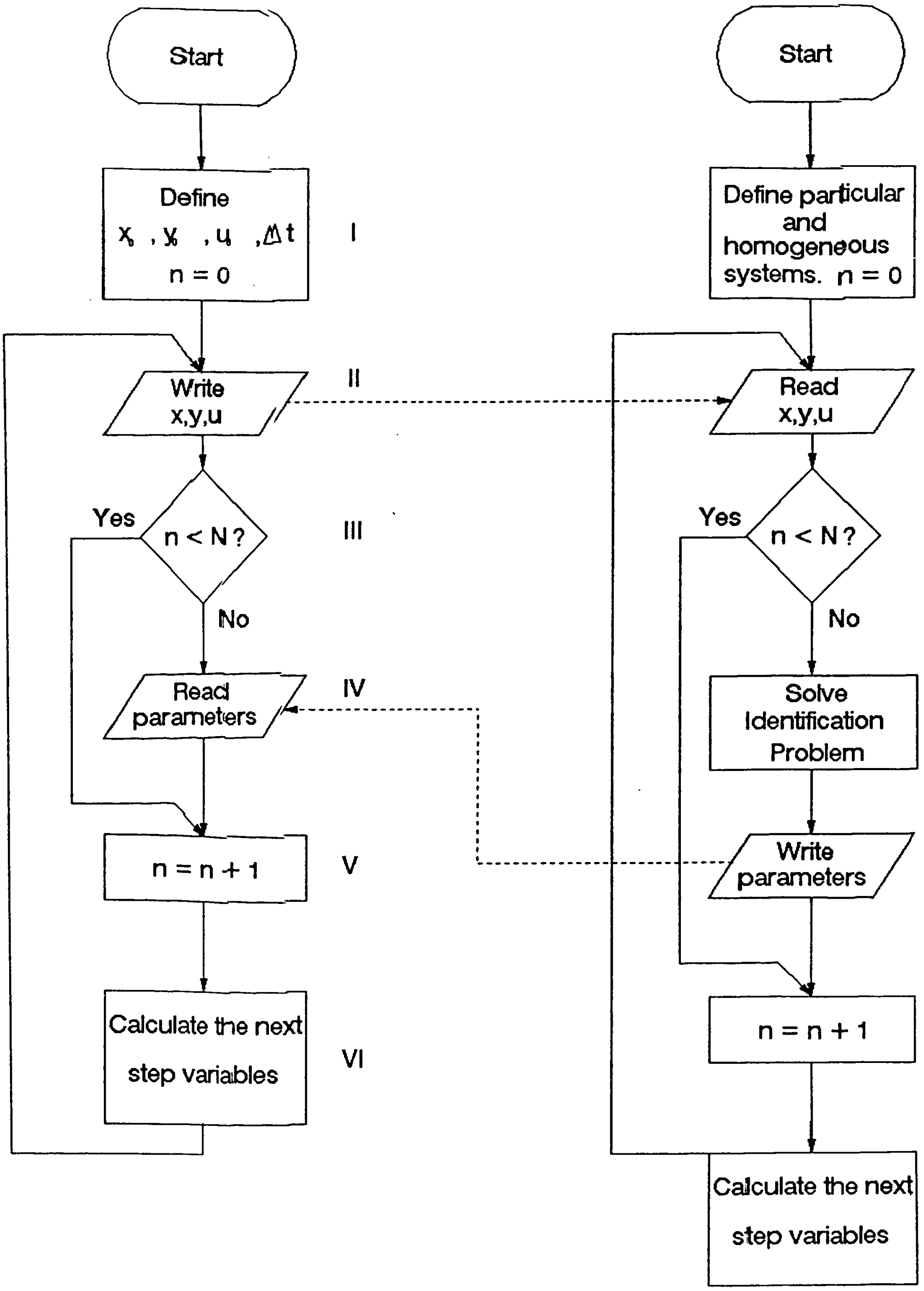
$$
[\ c\ ]_{k+1} = \Big(\sum_{i=1}^{N} [h(t_i)]^T \, [\ h(t_i)\ ] \Big)^{-1} *
$$

$$
\sum_{i=1}^{N} [\ h(t_i)]^T \{ [b(t_i)] - [p(t_i)] \}
$$

The value of first parenthesis of the right side of equation is placed in a matrix whose starting address is described by the SPAYDA. The starting address of the second parenthesis is described by SUMPAY. Both parenthesis initial values are already set with the initialization program. New values of $\{\ [h(t_i)]^T\ [h(t_i)]\ \}$ and $[h(t_i)]^T\ \{\ [b(t_i)]-[p(t_i)]\ \}$ are calculated and added to SPAYDA and SUMPAY. It can be seen that the matrix inverse and multiplication of the parenthesis are not done before Nth the observation. After the Nth step identification is realized and the identified parameters are replaced in the particular system matrix. All particular outputs and all homogeneous outputs and Eq(5.26) are calculated for the same input. Thus, two iterations are completed. Then, the preparation of the next step is initiated. All inputs are shifted back for the new inputs to be Nth. All identification results are converted to COMPAQ format and sent to the COMPAQ. The program returns to read new data. This main program assembly code is given in appendix A.1.13.

7.6. Conclusion

The identification operation was programmed in the assembly code of TMS320C30 DSP. It occupies less than 1.5k program memory and uses less than 2k RAM. The floating point precision is maintained throughout. Even the floating point inversion is iterated 10 times and 40-bit extended registers are operated. In addition, the integration is solved by a fourth order method. This program can iterate two times for 40 step averaging in a 40 msec sampling period. All assembly source program has been originally developed for this work. Only, the floating point inversion program has been taken from reference [46].



P.C. Flow Chart

TMS320C30 Flow Chart

Fig.7.1. Flow Charts of the Processors Programs

CHAPTER 8. CONCLUSION AND SUGGESTION FOR FURTHER WORK

8.1. Conclusion

The on-line identification of the continuous model of an aircraft by a direct method has been studied in a real-time simulation. Dettailed information and conclusions have been presented for each chapter, but the overall conclusions are as follows.

In chapter 2, a review and literature survey of well known on-line identification methods were presented for the continuous model. Indirect methods and the transfer from the discrete-model to the continuous model were explained in detail. The disadvantage of transfer methods from the discrete model to the continuous model was given by an example. The quasilinearization of Newton-Raphson method were presented as the direct identification methods.

In chapter 3, the dynamics of an aircraft were explained to show which parameters dominate the aircraft system. The complexity of the aircraft motion was simplified by ignoring second order effect.The necessity of auto control of the aircraft has been reviewed using stability criteria.

In chapter 4, the model reference adaptive control and its stability have been given briefly with respect to the controller design. Self tuning regulator design was also presented with different application methods. Both control methods were reviewed for the non-minimum phase systems and unstable systems. It was found that the model reference controller was inapplicable, but the self tuning method was applicable and also suitable to optimize the controller, because the regulator design could be implemented without zero cancellation in the self tuning system was given this opportunity. Newly developed spectral factorization methods were introduced. An example of the non-minimum phase aircraft system control was given.

In chapters 5, 6 and 7 the direct continuous model identification of the aircraft dynamics was implemented as a real-time simulation.

In chapter 5, the Newton-Raphson method was used as an identification method. The equations of the longitudinal motion of the aircraft were redefined in this chapter. The time-constant parameters and the time-varying parameters were identified in the noise-free systems. Then, the parameters were estimated with noise on the measured responses. In the next step, atmospheric turbulence and affects were included in the equations of the aircraft dynamics. The identification algorithm was applied to the gust affected aircraft motion. The effect of the measurement noise and atmospheric turbulence on the parameter identification were reviewed for different amplitudes. The ratio of the noise to the signal was changed from 0 to 0.3 . The gust was represented with a random function with specific amplitudes, which were given in Table.5.1.. The gust is not a controllable input, therefore identification was carried out between the controllable input and gust affected output. Thus, the parameters were not the actual parameters of the system, but they could represent a system according to system input. However, the actual parameter could be solved from equations, which were given in this chapter.

The identification method used in this work successfully estimated the necessary parameters within the transient period. This provides an opportunity to adapt the controller of the attack angle of the aircraft. The use of a fourth order numeric integration method instead of the linear integration filter increased the accuracy of the result. Even under high noise and turbulence affects, this method can be used to identify the system in 60 samples. When this method is compared with other estimation methods, which were given in chapter 2, very good performance was obtained with the limited samples and high noise level.

In chapter 6, the identification board, the model computer and the interfaces between them were given with circuit diagram. The control signals of the communication were given with time diagram. Two PAL were programmed to obtain the control signals with minimum delay. 32-bit data was transferred as 2x16-bit by an asynchrone parallel communication method because the computer could communicate with 16-bit.

CHAPTER 8. CONCLUSION AND SUGGESTION FOR FURTHER WORK

8.1. Conclusion

The on-line identification of the continuous model of an aircraft by a direct method has been studied in a real-time simulation. Detailed information and conclusions have been presented for each chapter, but the overall conclusions are as follows.

In chapter 2, a review and literature survey of well known on-line identification methods were presented for the continuous model. Indirect methods and the transfer from the discrete-model to the continuous model were explained in detail. The disadvantage of transfer methods from the discrete model to the continuous model was given by an example. The quasilinearization of Newton-Raphson method were presented as the direct identification methods.

In chapter 3, the dynamics of an aircraft were explained to show which parameters dominate the aircraft system. The complexity of the aircraft motion was simplified by ignoring second order effect.The necessity of auto control of the aircraft has been reviewed using stability criteria.

In chapter 4, the model reference adaptive control and its stability have been given briefly with respect to the controller design. Self tuning regulator design was also presented with different application methods. Both control methods were reviewed for the non-minimum phase systems and unstable systems. It was found that the model reference controller was inapplicable, but the self tuning method was applicable and also suitable to optimize the controller, because the regulator design could be implemented without zero cancellation in the self tuning system was given this opportunity. Newly developed spectral factorization methods were introduced. An example of the non-minimum phase aircraft system control was given.

In chapters 5, 6 and 7 the direct continuous model identification of the aircraft dynamics was implemented as a real-time simulation.

In chapter 5, the Newton-Raphson method was used as an identification method. The equations of the longitudinal motion of the aircraft were redefined in this chapter. The time-constant parameters and the time-varying parameters were identified in the noise-free systems. Then, the parameters were estimated with noise on the measured responses. In the next step, atmospheric turbulence and affects were included in the equations of the aircraft dynamics. The identification algorithm was applied to the gust affected aircraft motion. The effect of the measurement noise and atmospheric turbulence on the parameter identification were reviewed for different amplitudes. The ratio of the noise to the signal was changed from 0 to 0.3 . The gust was represented with a random function with specific amplitudes, which were given in Table.5.1.. The gust is not a controllable input, therefore identification was carried out between the controllable input and gust affected output. Thus, the parameters were not the actual parameters of the system, but they could represent a system according to system input. However, the actual parameter could be solved from equations, which were given in this chapter.

The identification method used in this work successfully estimated the necessary parameters within the transient period. This provides an opportunity to adapt the controller of the attack angle of the aircraft. The use of a fourth order numeric integration method instead of the linear integration filter increased the accuracy of the result. Even under high noise and turbulence affects, this method can be used to identify the system in 60 samples. When this method is compared with other estimation methods, which were given in chapter 2, very good performance was obtained with the limited samples and high noise level.

In chapter 6, the identification board, the model computer and the interfaces between them were given with circuit diagram. The control signals of the communication were given with time diagram. Two PAL were programmed to obtain the control signals with minimum delay. 32-bit data was transferred as 2x16-bit by an asynchrone parallel communication method because the computer could communicate with 16-bit.

In chapter 7, the identification board was programmed in assemble code of the TMS320C30 DSP. The operations of the identification algorithm in the assemble code were developed. The source program is given in appendix. The model computer input and output were programmed in assembly code of the 80286 microprocessor. All the necessary processor software was developed by the author..

The on-line continuous time model identification attempted in this thesis is intended to apply to aircraft systems where it is desired to control auto-landing. It has been achieved as a real time simulation for different environment including high level measurement noise and different atmospheric turbulence conditions. In this study, all the data used was for a RAVEN 201 pilotless aircraft. Therefore the results are suitable for real applications and the implementation may used for the auto-landing of the RAVEN 201 aircraft.

8.2. Suggestion for Further Work

The possibility of on-line identification of an aircraft was shown in this work. A parallel communication link and its interface were used between the identification board and the model computer. As an alternative a serial communication system for the application could be developed. The TMS320C30 is suitable for both serial communication and modem connection.

It has been shown how the identification time is dependent on the measurement noise level. On the other hand, early estimation of the parameter is necessary in order to apply it to an adaptive control system. Therefore, the identification time should be as short as possible to increase the sensitivity of the measurement system.

Assembly routines given here can be used for the adaptive control of an aircraft in future studies..

ACKNOWLEDGEMENTS

I would like to thank the following people;

Mr. J.K.M.MacCORMAC for his supervision and encouragement during my studies at the Bath University.

My friends, Ramazan Tasaltun and Safak Aksoy for their help in English.

My wife for her patience during the my studies.

I would also like to thank "ULUDAG UNIVERSITY" for their financial support to this study.

20- SAGE, A.P. Optimum Systems Control, Prentice-Hall, Englewood Cliffs, n. j. , 1968.

21- DETCHMENDY, D.M. and SHRIDAR, R. "On the Experimental Determination of the Dynamical Characteristics of Physical Systems," Proc. National Electronics Conf. , pp.522-530, Chicago, 1965.

22- MacCORMAC, J.K.M. "The Use Of Hybrid Computation In An On-line Identification Scheme," IFAC,Budapest,1968,

23-. KALABA, R. AND SPINGARN, K. "On The Rate of Convergence of The Quasilinearization Method," IEEE Trans. Aut. Control Vol.10, pp.798-799,1983.

24- ISERMANN, R., BAUR, U., BAMBERGER, W., KNEPPO, P. and SIEBERT, H. "Comparision of Six On-Line Identification and Parameter EStimation Methods," Automatica, Vol.10, pp.81-103, 1974.

25- SARIDIS, G.N., "Comparision of Six On-Line Identification Algorithms," Automatica, Vol.10, pp.69-79, 1974.

26- BABISTER, A.W., Aircraft Dynamic Stability and Response, Pergamon Press Ltd., U.K., 1980.

35- CLARKE, D.W. , KANJILAL, P.D. and MOHTADI, C. " A Generalized LQG Approach to Self-Tuning Control ", Part I., Int. J. Control , 1985, Vol.41, No.6, pp 1509-1523.

36- ÅSTRÖM, K.J. and WITTENMARK, B. "LQG Self Tuner", IFAC Symposium on Adaptive Control, 1983, Sanfransisco, U.S.A.

37- GRIMBLE, M.J. , "Controllers for LQG Self-Tuning Applications with Coloured Measurment Noise and Dynamic Costing", IEE Procedings, 1986, Vol.133, Pt.D, No.1, pp 19-29

38- HUNT, K.J. , GRIMBLE, M.J. and JONES, R.W., IFAC 10th Triennial World Congress, Munich, FRG, 1987, pp 169-174

39- GOODWIN, G.C. and MAYNE, D.Q. ," A Parameter Estimation Perspective of Continuous Time Model Reference Adaptive Control " Automatica, Vol23., 1987, pp.57-70

40- ROSENBERG, K., Private Communication, 1991

41- ETKIN, B., Dynamics of Atmospheric Flight, John Wiley & Sons, Inc.,1972

42- INTEL, 80386 High Performance 32-Bit Microprocessor with

REFERENCES :

1- WHITAKER, H.P. , YAMRON, J. and KEZER, A., Design of Model Referance Adaptive Control System, Report R-164, 1958, Instrumentation Laboratory, M.I.T, Cambrigde

2- ÅSTRÖM, K.J. and WITTENMARK, B. "On Self Tuning Control," Automatica, Vol.9, 1973, pp.185-199.

3- ISERMANN, R., "Process Fault Detection Based on Modelling and Estimation Method - A Survey," Automatica, Vol.20, No.4, pp.387-404, 1984

4- SINHA, N.K. and KUSZTA, B. " On-line Identification of Discrete-Time Systems," in Modelling And Identification of Dynamic System. Van Nostrad Reinhold Comp. England (1983).

5- ASTRÖM, K.J and WITTENMARK, B., Adaptive Control, Addison Wesley Publishing, U.S.A., 1989.

6- CLARKE, D.W. "Generalized Least-Square Estimation of The Parameters of A Dynamic Model," Preprints of The First IFAC Symposium On Identification. Prague, Czechoslovakia, (1967).

7- HASTING-JAMES, R. and SAGE, M.W. "Recursive Generalized Least

APPENDIX 1

Square Procedure For On-line Identification Of Process Parameters," Proc. IEE 116, pp. 2057-2062, 1969.

8- ÅSTRÖM, K.J. and BOHLIN, N.T. "Numerical Identification Of Linear Dynamic Systems From Normal Operating Records," Proc. IFAC Symposium On Self-Adaptive Control Systems, 96-110 Teddington, England, 1965.

9- GERTLER, J. and BANYASIZ, C. "A Recursive (On-line) Maximum Likelihood Identification Method," IEEE Trans. Aut. Control, Vol. AC-19, pp. 816-820, 1974.

10- WONG, K.Y. and POLAK, E. " Identification Of Linear Discrete Time Systems Using The Instrumental Variable Methods," IEEE Trans. Aut. Control, Vol. AC-12, 707, 1967.

11- YOUNG, P.C. " An Instrumental Variable Method For Real-Time Identification Of A Noisy Process," Automatica vol.6, pp. 271-287, 1970.

12- YOUNG, P.C. Lectures On The Parameter Estimation, Summer School: Theory And Practice Of Systems Modelling and Identification. Toulouse, 1972.

13- BUDIN, M.A. " Minimal Realization Of Discrete Linear Systems From Input-Output Observation," IEEE Trans. Aut. Control, Vol.

27- BLAKELOCK, J.H., Automatic Control of Aircraft and Missiles, John Wiley & Sons, Inc., U.S.A., 1965.

28- ASTROM, K.J. and WITTENMARK, B., Computer Controlled Control Systems, 1984

29- PARKS, P.C. "Lyapunov Redesign of Model Reference Adaptive Control Systems", IEEE Trans. Auto. Cont. , Vol. AC-11, No 3, 1966, pp 362-367.

30- LANDAU, Y.D. Adaptive Control : The Model Reference Approach , Markel Decker, Inc., 1979.

31- MONOPOLI, R.V. "Model Reference Adaptive Control with An Augment Error Signal", IEEE Trans. Auto. Cont. , 1974, pp 474-483

32- ELLIOT, H. "Direct Adaptive Pole Placement with Application to Nonminimum Phase Systems", IEEE Trans. Auto. Cont. , 1982 , pp 720-722.

33- CLARKE, D.W. and GAWTHROP, P.J. "Self Tuninig Controller", IEE Proc., 1975, Vol.122, No.9, pp 929-934.

34- CLARKE, D.W. "Self Tuning Control of Nonminimum Phase System", Automatica, Vol. 20, 1985, No. 5, pp. 501-517

Integrated Memory Management, Intel Corp.,1986

43- TURLEY, J.L., Advanced 80386 Programming Techniques, Mc Graw-Hill, Berkeley, California, 1988.

44- COMPAQ DESKPRO 386 Technical Reference Guide, COMPAQ Computer Corp.,1986.

45- MUELLER, J. and WANG, W., Microsoft Macro Assembler 5.1: Programming in the 80386 Environment, Windcrest Books, 1990.

46- Third-Generation TMS320 User's Guide, Texas Instrument Corp., 1988

47- GERALD, C.F. and WHEATLEY, P.O., Applied Numeric Analysis, Addision-Wesley Publishing Comp., 1989.

A.1.1.

```
;
;GO - Data values to parallel link at 300H
;
.286C
;
F_GO        STRUC                       ;
PARMLINE    DD      ?                   ;address word for start of data
F_GO        ENDS                        ;
;
STACK    SEGMENT PARA    'STACK'        ;
         DB      64 DUP(?)              ;
STACK    ENDS                           ;
;
D_GO        SEGMENT 'DATA'              ;
         DB      'GO'                   ;
SP_SAVE  DW      0                      ;
         DD      GO                     ;
         DD      0                      ;
D_GO     ENDS                           ;
;
C_GO     SEGMENT 'CODE'                 ;
         ASSUME  CS:C_GO,DS:D_GO
         DW      SEG     D_GO           ;
GO PROC     FAR
         PUBLIC  GO
         MOV     AX,D_GO
         MOV     DS,AX
         MOV     SP_SAVE,SP
         MOV     AX,0300H
         MOV     DX,AX                  ;save the output port to DX
         SUB     AX,AX
         PUSH    DS                     ;save DS onto stack
         LDS     SI,ES:PARMLINE[BX] ;load address of first  argument
                                        ;into DS:SI
         MOV     AX,[SI]                ;load the first part of data
         OUT     DX,AX                  ;send data to output port
         MOV     DX,0304H
         OUT     DX,AX                  ;write 304h for flag
         MOV     DX,0302H
GB:      IN      AL,DX
         AND     AL,01                  ;check acknowledgement
         JZ      GB                     ;if no, wait it
         OUT     DX,AX                  ;if yes clear the flags
         MOV     DX,0300H
         ADD     SI,0002H
         MOV     AX,[SI]                ;load the second part of data
         OUT     DX,AX                  ;send data to output port
         MOV     DX,0304H
         OUT     DX,AX                  ;write 304h for flag
         MOV     DX,0302H
         NOP
         NOP
```

(Continued)

```
GB1:    IN      AL,DX
        AND     AL,01           ;check acknowledgement
        JZ      GB1             ;if no, wait it
        OUT     DX,AX           ;if yes clear the flags

        POP     DS              ;restore DS
        RET                     ;return to fortran
;
GO ENDP
C_GO        ENDS
        END
```

(Concluded)

A.1.2.

```
;COMEP - inputs data values from parallel link at 300H
;
.286C
;
F_COMEP         STRUC               ;
PARMLINE         DD      ?          ;address word for start of data
F_COMEP         ENDS                ;
;
STACK   SEGMENT PARA    'STACK'     ;
        DB      64 DUP(?)           ;
STACK   ENDS                        ;
;
D_COMEP         SEGMENT 'DATA'      ;
        DB       'COMEP'            ;
SP_SAVE DW      0                   ;
        DD      COMEP               ;
        DD      0                   ;
MASK    DB      03H                 ;
D_COMEP         ENDS                ;
;
C_COMEP         SEGMENT 'CODE'      ;
        ASSUME  CS:C_COMEP,DS:D_COMEP
        DW      SEG     D_COMEP     ;
COMEP  PROC     FAR                 ;
        PUBLIC  COMEP               ;
        MOV     AX,D_COMEP          ;
        MOV     DS,AX               ;
        MOV     SP_SAVE,SP          ;
;
;ES:BX has vector to the parameter list control block FRAME
;
        MOV     AX,0302H            ;load address of software  handshake
                                    ;port into AX
        MOV     DX,AX               ;copy AX into DX
        SUB     AX,AX               ;get zero in AX
        PUSH    DS                  ;save DS onto stack
        LDS      SI,ES:PARMLINE[BX] ;load address of first  argument
                                    ;into DS:SI
HJ:     IN      AL,DX
        CMP     AL,03               ;is data ready ?
        JNZ     HJ                  ;if no, wait it
        MOV     DX,0300H
        IN      AX,DX               ;input data value from port 0300H
        MOV     DX,0302H
        OUT     DX,AX               ;clear flag
        MOV     DX,0304H
        OUT     DX,AX               ;send acknowledgement
        MOV     [SI],AX             ;save the low part of data
        MOV     DX,0302H
HJ1:    IN      AL,DX
        CMP     AL,03               ;is data ready ?
        JNZ     HJ1                 ;if no, wait it
```

(Continued)

```
        MOV     DX,0300H
        IN      AX,DX               ;input data value from port 0300H
        MOV     DX,0302H
        OUT     DX,AX               ;clear flag
        MOV     DX,0304H
        OUT     DX,AX               ;send acknowledgement
        ADD     SI,0002H
        MOV     [SI],AX             ;save the high part of data
        POP     DS                  ;restore DS
        RET                         ;return to fortran
;
COMEP   ENDP                        ;
C_COMEP         ENDS                ;
        END                         ;
```

(Concluded)

A.1.3.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
* THIS IS INITIALIZATION PROGRAM FOR IDENTIFICATION.                *
* STEP NUMBER VARIABLE STEPN CAN BE VARIED DUE TO DESIRED VALUE      *
* RAM BLOCK 1 DIVIDED THREE PART. FIRST 200H MEMORY SPACE IS ADDED *
* TO RAM  BLOCK 0, 100H MEMORY IS FOR VARIABLES, 100H MEMORY IS FOR*
* STACK.                                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **


         .option X
         .global BEGIN,INIT
         .sect    "init"       ; Named section
RESET    .word    INIT         ; RS- loads address INIT to PC
         .data
MASK     .word    0FFFFFFFFH
BLK0     .word    0809800H     ; Beginning address of RAM block 0
BLK1     .word    0809C00H     ; Beginning address of RAM block 1
STCK     .word    0809F00H     ; Beginning of stack
CTRL     .word    0808000H     ; Pointer for peripheral-bus memory
*                              ; map
NEGONE     .word  0FFFFFFH     ; Minus value
N          .word  0000002H     ; N, P, R are dimension variable.
P          .word  0000002H     ;
R          .word  0000001H     ;
*
* * * * * * * TEMPORARY VARIABLES * * * * * * * *   * * *
*
I          .word  0000000H
J          .word  0000000H
K          .word  0000000H
L          .word  0000000H
L1         .word  0000000H
INDEX      .word  080981DH
IROW       .word  0000000H
ICOL       .word  0000000H
DEGROW     .word  0809830H
NA         .word  0000000H
DEG        .word  0000000H
DEG1       .word  0000000H
DEG2       .word  0000000H
NIPIVO     .word  0809813H
NINDEX     .word  080981DH
NDEGRO     .word  0809830H
NPIV       .word  0809809H
NAMAX      .word  0809852H
D          .word  0000000H
THUN       .word  0000000H
NTT        .word  0000000H
DEGSPC     .word  0000000H
TT         .word  0000000H
MINUS      .word  0000000H
NN         .word  0000000H
NM1        .word  0000000H
```

(Continued)

```
NM2      .word   0000000H
NM3      .word   0000000H
NPR      .word   0000000H
NR       .word   0000000H
NLI      .word   0000000H
KR       .word   0000000H
TR       .word   0000000H
MNR      .word   0000000H
T        .word   0000000H
PN       .word   0000000H
NP       .word   0000000H
AMAX     .word   0809852H      ; AMAX is for matrix inverse
M5       .word   0809853H      ; M5 is  temporary matrix
M6       .word   0809887H      ; M6 is temporary matrix
M7       .word   08098BBH      ;  M7 is for power supply
MK1      .word   08098C0H      ; All MK* are for integration
MK2      .word   08098C4H
MK3      .word   08098C8H
MK4      .word   08098CCH
XSMAT    .word   0809900H      ; XSMAT is observed values.
TIME     .word   0809A30H      ; TIME is step size.
PMAT     .word   0809A02H      ; PMAT is particular matrix.
XPMAT    .word    0809A08H     ; XPMAT is particular system
*                              ; output matrix.
UP       .word   0809A06H      ; Particular system control vector
XHMAT    .word   0809A50H      ; Homogeneous system output vector
UHMAT    .word   0809A52H      ; Homogeneous system control vector
ITAD     .word   0809BB0H      ; Actual system control input
ITA      .word   0809BD8H      ; Delayed input.
AD       .word   0809AD0H      ; AD = The general homogeneous matrix
PAR      .word   0000000H
VAR      .word   0000000H
CONT     .word   0000000H
PRLIO    .word   0000000H      ; The communication port address.
PRLIOO   .word   0000000H      ; Temporary input memory.
ERRM     .float 0.001000
PMA      .word   08098D0H
HO       .word   0000000H
OUTPUT   .word   0808082H
CHECK    .word   0000000H
MISTA    .word   0000000H
LAST     .word   0000000H
INPUT    .word   0000000H
TIME1    .float 0.989200
STEP     .float 0.040000
TEMPM    .word   0809D80H
SABIT    .word   0809800H      ; The constants of the conversion.
SABIT1   .word   0809802H      ; The constants of the conversion.
BTI      .word   0809950H      ; The sum vector of the observations
PTI      .word   0809955H      ; The sum vector of the particular
*                              ; outputs.
XHM      .word   080995AH      ; The sum vector of the homogeneous
*                              ; outputs
```

(Continued)

```
XHMT        .word  0809973H      ; Transpose of the XHM
SUMPAY      .word  080998CH      ; ∑ hi (bi - pi )
SPAYDA      .word  0809DA0H      ; ∑ hi hi^T
STEPN       .word  0000000H      ; Step number
FIRSAT      .word  0000000H
SAY         .word  0000000H
SAY1        .word  0000000H
TCDD        .word  0000000H
*
*
                 .text
INIT     LDI  80H,DP             ; Point the DP register to page 0
         LDI  1800H,ST           ; Clear and enable cache, and disable
*                                ; OVM
         LDI   0809H,AR1
         LSH   4,AR1
         ADDI  8,AR1
         LSH   8,AR1
         STI   AR1,@BLKO         ; BLKO includes the beginning address
*                                ; of RAM  block 0
         LDI   600H,R6
         ADDI  R6,AR1
         STI   AR1,@BLK1         ; BLK1 includes the  beginning  address
*                                ; of RAM1  block 1
        LDI   AR1,R5
        ADDI  100H,R5
        STI   R5,@STCK           ; STCK points the address of stack
*                                ; pointer
         SUBI  100H,R5
         LDI   -1,R7
         STI   R7,@MASK          ; MASK includes FFFFFFFFH
         RPTS  4
         SUBI  R6,R5
         STI   R5,@CTRL          ; CTRL includes the beginning address
*                                ; of peripheral bus memory map
         LDI  @MASK,IE           ; Unmask all interrupts
         LDI  @BLKO,ARO          ; ARO points to block 0
         LDI  @BLK1,AR1          ; AR1 points to block 1
         ADDI 5,AR1
         LDF  0.0,RO             ; Zero register RO
         RPTS 1023               ; Repeat 1024 times ...
         STF  RO,*ARO++(1)       ; Zero out location in RAM block 0
         RPTS 511
         STF  RO,*ARO++(1)       ; Zero out location in RAM block 1.
         RPTS 506
         STI  RO,*AR1++(1)       ; Zero out location in RAM block1.
*                                ; repeat is not 1024 because we must
*                                ; protect sembols' value
*
         LDI   @CTRL,ARO         ; LOAD in ARO the pointer to control
*                                ;    register
```

(Continued)

```
        STI    R0,*+AR0(0)        ; Init DMA control
        STI    R0,*+AR0(32)       ; Init timer 0 control
        STI    R0,*+AR0(48)       ; Init timer 1 control
        STI    R0,*+AR0(64)       ; Init serial 0 global control
        STI    R0,*+AR0(66)       ; Init serial 0 xmt control
        STI    R0,*+AR0(67)       ; Init serial 0 rcv control
        STI    R0,*+AR0(68)       ; Init serial 0 timer control
        STI    R0,*+AR0(80)       ; Init serial 1 global control
        STI    R0,*+AR0(82)       ; Init serial 0 xmt control
        STI    R0,*+AR0(83)       ; Init serial 0 rcv control
        STI    R0,*+AR0(84)       ; Init serial 0 timer control
        STI    R0,*+AR0(96)       ; Init parallel interface control
        STI    R0,*+AR0(100)      ; Init I/O interface control
*
        LDI    @STCK,SP           ; Initialize the stack pointer
        OR     2000H,ST           ; Global interrupt enable
*
        BR     BEGIN              ; Branch to beginnig of the
                                  ;application.

*
* This part of initialization places the values in the variable.
*
          .text
          .global BEGIN
BEGIN     .set $
       LDI    @BLK0,AR0
       LDI    @BLK1,AR1
       LDI    @MASK,R1
       STI    R1,*+AR1(5)         ; NEGONE = FFFFFFFFH
       LDI    2,R1
       STI    R1,*+AR1(6)         ; N = 2
       STI    R1,*+AR1(7)         ; P = 2
       LDI    1,R1
       STI    R1,*+AR1(8)         ; R = 1
       STI    AR0,*+AR1(80)       ; SABIT = 809800H
       ADDI   2,AR0
       STI    AR0,*+AR1(81)       ; SABIT1 = 809802H
       ADDI   7,AR0
       STI    AR0,*+AR1(25)       ; NPIV = 809809H
       ADDI   10,AR0
       STI    AR0,*+AR1(22)       ; NIPIVO = 809813H
       ADDI   10,AR0
       STI    AR0,*+AR1(23)       ; NINDEX = 80981DH
       STI    AR0,*+AR1(14)       ;  INDEX = 80981DH
       ADDI   19,AR0
       STI    AR0,*+AR1(24)       ; NDEGRO = 809830H
       STI    AR0,*+AR1(17)       ;  DEGRO = 809830H
       ADDI   34,AR0
       STI    AR0,*+AR1(26)       ; NN = NAMAX
       STI    AR0,*+AR1(46)
       ADDI   1,AR0
       STI    AR0,*+AR1(47)       ; M5 = 809853H
```

(Continued)

```
ADDI   R2,R1
STF    R1,*+AR1(77)       ; P(1,2) = 0.9893
LDF    0.04,R1
LDI    070AH,R2
LSH    8,R2
ADDI   R2,R1
STF    R1,*+AR1(78)       ; STEP = 0.04 sec
LDI    @CTRL,R2
SUBI   4000H,R2
STI    R2,*+AR1(67)       ; PRLIO = 804000H
ADDI   5DD0H,R2
STI    R2,*+AR1(68)       ; PRLIIO = 809DD0H
ADDI   10H,R2
STI    R2,*+AR1(72)       ; OUTPUT = 809DE0H
LDI    @TIME,AR1
LDF    @STEP,R1
STF    R1,*AR1            ; TIME = 0.04
LDI    @SABIT,AR2
LDI    0FFH,R2
LSH    24,R2
STI    R2,*AR2            ; FF000000H
LDI    80H,R1
LSH    16,R1
STI    R1,*+AR2(1)        ; 00800000H
STI    R1,*+AR2(3)
SUBI   1,R1
STI    R1,*+AR2(4)        ; 007FFFFFH
LSH    8,R1
ADDI   0FFH,R1
STI    R1,*+AR2(2)        ; 7FFFFFFFH
STI    R0,@TT             ; TT = 0
STI    R0,@LAST           ; 0
BR     MAINPROGRAM
```

(Concluded)

```
ADDI  34H,AR0
STI   AR0,*+AR1(48)     ; M6 = 809887H
ADDI  34H,AR0
STI   AR0,*+AR1(49)     ; M7 = 8098BBH
ADDI  5,AR0
STI   AR0,*+AR1(50)     ; MK1 = 8098C0H
ADDI  4,AR0
STI   AR0,*+AR1(51)     ; MK2 = 8098C4H
ADDI  4,AR0
STI   AR0,*+AR1(52)     ; MK3 = 8098C8H
ADDI  4,AR0
STI   AR0,*+AR1(53)     ; MK4 = 8098CCH
ADDI  4,AR0
STI   AR0,*+AR1(70)     ; PMA = 8098D0H
ADDI  30H,AR0
STI   AR0,*+AR1(54)     ; XSMAT = 809900H
ADDI  0C8H,AR0
STI   AR0,*+AR1(82)     ; BTI = 8099C8H
ADDI  5,AR0
STI   AR0,*+AR1(83)     ; PTI = 8099CDH
ADDI  5,AR0
STI   AR0,*+AR1(84)     ; XHM = 8099D2H
ADDI  19H,AR0
STI   AR0,*+AR1(85)     ; XHMT = 8099EBH
ADDI  19H,AR0
STI   AR0,*+AR1(86)     ; SUMPAY = 809A04H
ADDI  19H,AR0
STI   AR0,*+AR1(55)     ; TIME = 809A1DH
ADDI  1H,AR0
STI   AR0,*+AR1(56)     ; PMAT = 809A1EH
ADDI  4,AR0
STI   AR0,*+AR1(58)     ; UP = 809A22H
ADDI  2,AR0
STI   AR0,*+AR1(57)     ; XPMAT = 809A24H
ADDI  0C8H,AR0
STI   AR0,*+AR1(59)     ; XHMAT = 809AEC
ADDI  2,AR0
STI   AR0,*+AR1(60)     ; UHMAT = 809AEE
ADDI  114H,AR0
STI   AR0,*+AR1(61)     ; ITAD = 809C00H
ADDI  0C8H,AR0
STI   AR0,*+AR1(62)     ; ITA = 809CC8H
ADDI  0C8H,AR0
STI   AR0,*+AR1(79)     ; TEMPM = 809D90H
ADDI  10H,AR0
STI   AR0,*+AR1(87)     ; SPAYDA = 809DB0H
LDI   3CH,R5
STI   R5,*+AR1(88)      ; STEPN = 3CH
ADDI  30H,AR0
STI   AR0,*+AR1(63)     ; AD = 809DECH
LDF   0.9892,R1
LDI   0C36H,R2
LSH   8,R2
```

(Continued)

A.1.4.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
/*          COMMAND FILE FOR LINKING C30 PROGRAMS                  */
/*                                                                 */
/* File Name: GES.CMD                                              */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **/
GES.OBJ          /* GES - System Identification Pogram             */

/* SPECIFY THE SYSTEM MEMORY MAP */

MEMORY
{
   VECS:    org = 0h         len = 040h       /* INTERRUPT VECTORS */
   ROM:     org = 0C0h       len = 07FFF40h   /* PROGRAM CODE      */
   IOM:     org = 0800000h   len = 02000h     /* -MSTRB I/O        */
   IO:      org = 0804000h   len = 02000h     /* -IOSTRB I/O       */
   RAM0:    org = 0809800h   len = 0400h      /* RAM BLOCK 0       */
   RAM1:    org = 0809E00h   len = 0100h      /* RAM BLOCK 1       */
   STACK:   org = 0809F00h   len = 0100h      /* SYSTEM STACK      */
}

/* SPECIFY THE SECTIONS ALLOCATION INTO MEMORY */

SECTIONS
{
   vectors:   {} > VECS      /* INTERRUPT VECTORS                  */
   stk_init:  {} > STACK     /* INITIAL STACK POINTER              */
   .text:     {} > ROM       /* CODE                               */
   .data:     {} > RAM1      /* ROM-RESIDENT TABLES & CONSTANTS    */
   .bss:      {} > RAM0      /* BSS DATA                           */
}
```

A.11.5.

```
* * * * * * * 804000H * * * * * * * * * *

804000H         PRLIO

* * * * * * * 809800H * * * * * * * * * *

809800H-01H     SABIT
809802H-04H     SABIT1
809813H-18H     IPIVOT
80981DH-27H     INDEX
809830H-34H     NDEGRO
809852H         AMAX
809853H-58H     M5
809887H-8DH     M6
8098BBH-BFH     M7
8098C0H-C3H     MK1
8098C4H-C7H     MK2
8098C8H-CBH     MK3
8098CCH-CFH     MK4
8098D0H-D1H     PMA

* * * * * * * 809900H * * * * * * * * * *

809900H-C7H     XSMAT
8099C8H-CCH     BTI   ( XS )
8099CDH-D1H     PTI   ( XP )
8099D2H-EAH     XHM   ( XH )
8099EBH-A03H     XHMT  ( XHT )
809A04H-1CH     SUMPAY(SIGMA(BI=PI))
809A1DH         TIME
809A1EH-21H     PMAT
809A22H-23H     UP
809A24H-EBH     XPMAT

* * * * * * * 809A00H * * * * * * * * * *

809AECH-BFFH     XHMAT & UHMAT
809C00H-C7H     ITAD
809CC8H-D8FH     ITA
809D90H-AFH     TEMPM
809DB0H-CFH     SPAYDA
809DD0H-DFH     PRLIIO
809DE0H-EFH     OUTPUT

* * * * * * * 809E00H * * * * * * * * * * *

809E00-56H      VARIABLES

* * * * * * * 809F00H * * * * * * * * * * *

      S T A C K
```

A.1.6.

```
* THIS PROGRAM CONVERTS THE FLOATING-POINT NUMBER  FROM  TMS320C30
* FORMAT TO IEEE FORMAT.
* ( AR1(0) ) = FF000000H
* ( AR1(1) ) = 00800000H
* ( R5 )     = DATA

TMSC   LDI    *+AR1(1),R2
       LSH    8,R2
       SUBI   R5,R2         ; Is data zero ?
       BZ     Z0            ; If yes, go to end of the program.
       AND3   R5,*AR1,R1    ; Obtain the exponent of TMS320C30 format
       ASH    -24,R1
       LDI     R1,R2
       AND     1,R2
       ADDI    1,R1
       ASH    -1,R1
       SUBI     1,R1
       ADDI    40H,R1
       LSH     24,R1        ; IEEE exponent
       PUSH    R5
       POPF    R5           ; Is the floating point value negative ?
       BN      T0           ; If yes, go to T0
       CMPI    0,R2         ; Is the exponent odd ?
       BZ      T1           ; If yes, go to T1.
       LSH    -8,R5         ; Mantissa is found.
       OR      R1,R5        ; Add exponent.
       RETS                 ; Conversion is  completed,  Go  back  to
                            ; main program.
T1     LSH     -8,R5
       OR      *+AR1(1),R5  ; Put 1 to the 23th bit.
       OR       R1,R5       ; Add exponent.
       RETS                 ;Conversion is  completed,  Go  back  to
                            ; main program.
T0     NEGI    R5
       LSH    -8,R5
       OR      R1,R5
       CMPI    0,R2
       BZ     T3
       LDI    80H,R2
       LSH     24,R2
       OR     R2,R5
       RETS                 ; Conversion is  completed,  Go  back  to
                            ; main program.
T3     OR      *+AR1(1),R5
       OR       R1,R5
       LDI    80H,R2
       LSH     24,R2
       OR     R2,R5
       RETS                 ; Conversion is  completed,  Go  back  to
                            ; main program.
Z0     LDI    0,R5
       RETS                 ; There is no need to comversion. Go back
                            ; to main program.
```

A.1.7.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS PROGRAM CONVERTS THE FLOATING-POINT NUMBER FROM IEEE FORMAT*
* TO TMS320C30 FORMAT.                                            *
* ( AR2(0) = FF000000H                                            *
* ( AR2(1) = 00800000H                                            *
* ( AR1(0) = 7FFFFFFFH                                            *
* ( AR1(1) = 00800000H                                            *
* ( AR1(2) = 007FFFFFH                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CMPQ    CMPI  0,R5            ; Is data zero ?
        BZ    Z3              ; If yes, go to Z3.
        AND3  R5,*AR1,R1
        AND3  R5,*+AR1(1),R2
        LSH   -24,R1
        SUBI  40H,R1
        ADDI  1,R1
        LSH   1,R1
        CMPI  0,R2
        BNZ   GO
        SUBI  1,R1
GO      LSH   24,R1           : Exponent is converted.
        LDI   *+AR1(2),R3
        AND3  R5,R3,R2        ; Mantissa is obtained.
        CMPI  0,R5            ; Is mantissa negative ?
        BN    GK              ; If yes, go to GK.
        OR3   R1,R2,R5        ; Add exponent to mantissa.
        RETS                  ; Conversion is  completed.  Go  back  to
                              ; main program.
GK      PUSH  R2
        POPF  R2
        NEGI  R2,R2           ; Add sign to mantissa.
        BNZ   KRL
        SUBI  1,R1
        AND   *AR2,R1
        LDI   80H,R2
        LSH   24,R2
KRL     LSH   -8,R2           ; Exponent is obtained.
        OR3   R1,R2,R5        ; Add exponent to mantissa.
        RETS                  ; Conversion is  completed.  Go  back  to
                              ; main program.

Z3      LDI   *+AR1(1),R1
        LSH   8,R1
        LDI   R1,R5
        RETS                  ; Conversion is  completed.  Go  back  to
                              ; main program.
```

A.1.8.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS PROGRAM CAN MAKE THE ADDITION OF TWO MATRIX AND THE         *
* SUBRACTION OF TWO MATRIX.                                        *
* THEIR DIMENSION IS (PxN).                                        *
* NM1 INCLUDES FIRST MATRIX STARTING ADDRESS.                      *
* NM2 INCLUDES SECOND MATRIX STARTING ADDRESS.                     *
* NM3 INCLUDES SUM MATRIX STARTING ADDRESS.                        *
* MINUS POINTS EITHER THE ADDITION OR THE SUBTRACTION.             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MAD     LDI     @P,R2           ;
        MPYI    @N,R2           ;
        SUBI    1,R2            ;
        STI     R2,@PN          ; It is the maximum shifting value.
        LDI     @NM1,AR1        ; Load starting address of A matrix
        LDI     @NM2,AR2        ; Load starting address of B matrix
        LDI     @NM3,AR3        ; Load starting address of C matrix
        LDI     @PN,RC
        LDI     @MINUS,R1
        BNZ     NEGA
        RPTB    A1
        LDF     *AR1++(1),R1
        ADDF    *AR2++(1),R1
A1      STF     R1,*AR3++
        RETSU
NEGA    RPTB    A2
        LDF     *AR1++(1),R1
        SUBF    *AR2++(1),R1
A2      STF     R1,*AR3++
        RETSU
```

A.1.9.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS IS THE MATRIX MULTIPLICATION PROGRAM                        *
* NM1 POINTS THE STARTING ADDRESS OF A MATRIX (PXN)                *
* NM2 POINTS THE STARTING ADDRESS OF B MATRIX (NXR)                *
* NM3 POINTS THE STARTING ADDRESS OF C MATRIX (PXR)                *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

MAT   LDI  @N,R1
      STI  R1,@DEG1          ; DEG1=N
      LDI  @R,R1
      STI  R1,@NR            ; NR=R
      SUBI 1,R1
      STI  R1,@KR            ; KR=R-1
      LDI  @P,R1
      STI  R1,@NPR           ; NPR=P
      LDI  @N,R1
      MPYI @R,R1
      SUBI 1,R1
      STI  R1,@DEG           ; DEG=N*R-1
      STI  R1,@MNR           ; MNR=N*R-1
      LDI  @NR,IR0
      LDI  @NM1,AR1          ; Load starting address of A matrix
      LDI  @NM2,AR2          ; Load starting address of B matrix
      LDI  @NM3,AR3          ; Load starting address of C matrix
      LDI  @N,R1
      STI  R1,@NN            ; NN=N
NEW   LDF  0.0,R5            ; c_ij = 0
BAS   MPYF3  *AR1++(1),*AR2++(IR0),R4
      ADDF   R4,R5           ; c_ij = Σ a_ik * b_kj
      LDI    @NN,R1
      SUBI   1,R1
      STI    R1,@NN
      BP     BAS
      STF    R5,*AR3++(1)
      SUBI   @MNR,AR2
      LDI    @N,R1
      STI    R1,@NN
      SUBI   @DEG1,AR1
      LDI    @NR,R1
      SUBI   1,R1
      STI    R1,@NR
      BP     NEW             ; Go to start another c value.
      ADDI   @R,R1
      STI    R1,@NR
      LDI    @NM2,AR2
      ADDI   @DEG1,AR1
      LDI    @NPR,R1
      SUBI   1,R1
      STI    R1,@NPR
      BP     NEW
      RETSU                  ; Go back to main program.
```

A.1.10.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THE  INVERSE OF A FLOATING-POINT NUMBER                          *
* THE FLOATING POINT-NUMBER v IS SRORED IN R3.  AFTER THE          *
* COMPUTATION IS COMPLETED, 1/v IS ALSO STORED IN R3.              *
* ARGUMENT ASSIGMENTS:                                             *
* ARGUMENT | FUNCTION                                              *
* ---------|-----------------                                      *
* R3       | v = NUMBER TO FIND THE RECIPROCAL OF (UPON THE CALL) *
* R3       | 1/v (UPON THE RETURN)                                 *
*                                                                  *
*  REGISTER USED AS INPUT : R3                                     *
*  REGISTERS MODIFIED : R3, R1, R2, R6                             *
*  REGISTER CONTAINING RESULT : R3                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

INVF  LDF   R3,R6          ; v is saved for later
      ABSF  R3             ; The algorithm uses v = |v|
      PUSHF R3
      POP   R1
      ASH   -24,R1         ; The 8 LSBs of R1 contain  the  exponent
                           ; of v.
      NEGI  R1
      SUBI  1,R1
      ASH   24,R1
      PUSH  R1
      POPF  R1             ; R1 = x[0] = 1.0 * 2^(-e-1)
*
      MPYF  R1,R3,R2       ; R2 = v * x[0]
      SUBRF 2.0,R2         ; R2 = 2.0 - v * x[0]
      MPYF  R2,R1          ; R1 = x[1] = x[0] * ( 2.0 - v * x[0] )

*
      MPYF  R1,R3,R2       ; R2 = v * x[1]
      SUBRF 2.0,R2         ; R2 = 2.0 - v * x[1]
      MPYF  R2,R1          ; R1 = x[2] = x[1] * ( 2.0 - v * x[1] )

      MPYF  R1,R3,R2
      SUBRF 2.0,R2
      MPYF  R2,R1

      MPYF  R1,R3,R2
      SUBRF 2.0,R2
      MPYF  R2,R1

      MPYF  R1,R3,R2
      SUBRF 2.0,R2
      MPYF  R2,R1

      MPYF  R1,R3,R2
      SUBRF 2.0,R2
      MPYF  R2,R1

      MPYF  R1,R3,R2
      SUBRF 2.0,R2
      MPYF  R2,R1
```

(Continued)

```
        MPYF   R1,R3,R2
        SUBRF  2.0,R2
        MPYF   R2,R1
*
        MPYF   R1,R3,R2
        SUBRF  2.0,R2
        MPYF   R2,R1            ; R1 = x[9] = x[8] * ( 2.0 - v * x[8] )
*
        RND    R1               ; This minimizes error in the LSBs
*
        MPYF   R1,R3,R2
        SUBRF  1.0,R2
        MPYF   R1,R2            ; R1 = x[9] * ( 1.0 - v * x[9] )
        ADDF   R2,R1            ; R1 = x[10] = x[9]*(1.0 - v*x[0])+ x[9]
*
        RND    R1,R3            ; Round since this is follow by a MPYF
*
        NEGF   R3,R2
        LDF    R6,R6            ; This set condition flags.
        LDFN   R2,R3            ; If v<0 , then R3 = -R3
        RETS                    ; Return to main program.
```

(Concluded)

A.1.11.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS PROGRAM SOLVES THE INTEGRATION WITH USING THE FOURTH ORDER *
* RUNGE-KUTTA ALGORITHM.                                           *
* THE VARIABLES PAR, VAR, CONT INCLUDE THE STARTING ADDRESSES  OF *
* THE PARAMETER VECTOR, THE STATE VARIABLE VECTOR AND THE CONTROL *
* VECTOR, RESPECTIVELY.                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
RUNGE  LDI   @PAR,R1
       STI   R1,@NM1
       LDI   @VAR,R2
       STI   R2,@NM2
       LDI   @CONT,R3
       LDI   @M5,R5
       STI   R5,@NM3
       PUSH  R1
       PUSH  R2
       PUSH  R3
       PUSH  R5
       CALL  MAT              ; Find A * Xn
       POP   R5
       POP   R3
       POP   R2
       STI   R3,@NM1
       STI   R5,@NM2
       STI   R5,@NM3
       LDI   @R,R4
       STI   R4,@N
       PUSH  R2
       PUSH  R3
       PUSH  R5
       CALL  MAD              ; Find [( A * Xn ) + ( B * Un )]
       POP   R5
       POP   R3
       POP   R2
       LDI   @TIME,AR3
       LDI   R5,AR1
       LDI   @MK1,AR2
       LDI   1,RC
       RPTB  HI
       LDF   *AR1++(1),R4
       MPYF  *AR3,R4
HI     STF   R4,*AR2++(1)     ; MK1 includes k1 of algorithm
       LDI   @MK1,AR1
       LDI   @M6,AR2
       LDI   1,RC
       RPTB  H1
       LDF   *AR1++(1),R4
       MPYF  0.5,R4
H1     STF   R4,*AR2++(1)
       STI   R2,@NM1
       LDI   @M6,R4
       STI   R4,@NM2
```

(Continued)

```
        LDI   @M7,R4
        STI   R4,@NN3
        PUSH  R2
        PUSH  R3
        PUSH  R5
        CALL  MAD              ; X = Xn + ( 0.5 * K1)
        POP   R5
        POP   R3
        POP   R2
        POP   R1
        STI   R1,@NM1
        STI   R5,@NM3
        LDI   @M7,R4
        STI   R4,@NM2
        LDI   @P,R4
        STI   R4,@N
        PUSH  R1
        PUSH  R2
        PUSH  R3
        PUSH  R5
        CALL  MAT              ; Find  A * X
        POP   R5
        POP   R3
        POP   R2
        STI   R3,@NM1
        STI   R5,@NM2
        STI   R5,@NM3
        LDI   @R,R4
        STI   R4,@N
        PUSH  R2
        CALL  MAD              ; Find [( A * X) + ( B * Un)]
        POP   R2
        LDI   @TIME,AR3
        LDI   R5,AR1
        LDI   @MK2,AR2
        LDI   1,RC
        RPTB  HI1
        LDF   *AR1++(1),R4
        MPYF  *AR3,R4
HI1     STF   R4,*AR2++(1)     ; K2= [( A * X) + ( B * Un)] * h
        LDI   @MK2,AR1
        LDI   @M6,AR2
        LDI   1,RC
        RPTB  H11
        LDF   *AR1++(1),R4
        MPYF  0.5,R4
H11     STF   R4,*AR2++(1)
        STI   R2,@NM1
        LDI   @M6,R4
        STI   R4,@NM2
        LDI   @M7,R4
```

(Continued)

```
        STI     R4,@NM3
        PUSH    R2
        CALL    MAD             ; X = X  + 0.5 * K
                                       n          2
        POP     R2
        POP     R1
        STI     R1,@NM1
        STI     R5,@NM3
        LDI     @M7,R4
        STI     R4,@NM2
        LDI     @P,R4
        STI     R4,@N
        PUSH    R1
        PUSH    R2
        PUSH    R3
        PUSH    R5
        CALL    MAT             ; Find  A * X
        POP     R5
        POP     R3
        POP     R2
        STI     R3,@NM1
        STI     R5,@NM2
        STI     R5,@NM3
        LDI     @R,R4
        STI     R4,@N
        PUSH    R2
        CALL    MAD             ; Find [( A * X) + ( B * U )]
                                                          n
        POP     R2
        LDI     @TIME,AR3
        LDI     R5,AR1
        LDI     @MK3,AR2
        LDI     1,RC
        RPTB    HI2
        LDF     *AR1++(1),R4
        MPYF    *AR3,R4
HI2     STF     R4,*AR2++(1)    ; K =0 [( A * X) + ( B * U )] * h
                                   3                      n
        LDI     @MK3,AR1
        LDI     @M6,AR2
        LDI     1,RC
        RPTB    H12
        LDF     *AR1++(1),R4
H12     STF     R4,*AR2++(1)
        STI     R2,@NM1
        LDI     @M6,R4
        STI     R4,@NM2
        LDI     @M7,R4
        STI     R4,@NM3
        PUSH    R2
        CALL    MAD             ; X = X  + K
                                       n    3
        POP     R2
        POP     R1
        STI     R1,@NM1
```

(Continued)

```
        STI     R5,@NM3
        LDI     @M7,R4
        STI     R4,@NM2
        LDI     @P,R4
        STI     R4,@N
        PUSH    R1
        PUSH    R2
        PUSH    R3
        PUSH    R5
        CALL    MAT             ; Find ( A * Xn)
        POP     R5
        POP     R3
        POP     R2
        POP     R1
        STI     R3,@NM1
        STI     R5,@NM2
        STI     R5,@NM3
        LDI     @R,R4
        STI     R4,@N
        PUSH    R2
        CALL    MAD             ;  Find [( A * X) + ( B * Un)]
        POP     R2
        LDI     @TIME,AR3
        LDI     R5,AR1
        LDI     @MK4,AR2
        LDI     1,RC
        RPTB    HI3
        LDF     *AR1++(1),R4
        MPYF    *AR3,R4
HI3     STF     R4,*AR2++(1)    ; K4 = [( A * X) + ( B * Un)] * h
        LDI     R5,AR1
        LDI     @MK1,AR2
        LDI     @MK2,AR3
        LDI     @MK3,AR4
        LDI     @MK4,AR5
        LDI     1,RC
        RPTB    HI4
        LDF     *AR2++(1),R1
        LDF     *AR3++(1),R6
        MPYF    2,R6
        LDF     *AR4++(1),R3
        MPYF    2,R3
        LDF     *AR5++(1),R4
        ADDF     R6,R1
        ADDF     R4,R3
        ADDF     R3,R1
        MPYF     0.1666666,R1
HI4     STF      R1,*AR1++(1) ; K = 0.1666 * ( K1 + 2*K2 + 2*K3 + K4 )
        STI      R2,@NM1
        LDI      @INPUT,R6
        BNZ      W1
```

(Continued)

```
        LDI     @HO,R6
        BZ      WRN
        MPYI    14H,R6          ; This is for the homogeneous systems
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD             ; X    = X  + K
                                   n+1    n
        LDI     @P,R1
        STI     R1,@N
        RETSU                   ; Go back to main program
W1      LDI     1,R6            ; This is for the particular system
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD             ; X    = X  + K
                                   n+1    n
        STI     R0,@INPUT
        RETSU                   ; Go back to main program

WRN     ADDI    R7,R2           ; This is for the input system
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD             ; X    = X  + K
                                   n+1    n
        LDI     @P,R1
        STI     R1,@N
        RETSU                   ; Go back to main program
                                                            (Concluded)
```

/A.1.12.

```
** * * .* * * * * * ( * * * * * * * * * * * * * * * * * * * * * * *
** THIS PROGRAM IS FOR THE MATRIX INVERSE                              *
** AD is the starting address of matrix A(NxN)                         *
** * * * * * * * * * : * * * * * * * * * * * * * * * * * * * * * * * *
]INVER   LDI    @AD,R1
         STI    R1,@IA          ; Store the starting address  of  the
                                ; matrix
         LDI    @INDEX,R1
         STI    R1,@IINDEX      ; Store the starting address  of  the
                                ; index
         LDF    0.0,R0
         LDI    @NIPIVO,AR1
         RPTS   @N              ; The dimension of the matrix is nxn
         STI    R0,*AR1++(1)    ; All IPIVOT = 0
         LDI    1,R1
         STI    R1,@I           ; I = 1
\YIRMI   LDI    @NA,AR1         ; ( AR1 ) = NA
         LDI    @N,R1
         STI    R1,@IP
         STI    R1,@DEGSPC
         LDI    @N,R2
         MPYI3  R2,R2,R4
         SUBI   1,R4
         STI    R4,@NN          ; NN=N**2-1
\YUZ     LDI    @DEGSPC,AR0
         LDI    1,R1
         STI    R1,@J
         LDI    @N,R1
         STI    R1,@NR
         LDI    @NAMAX,AR2
         STF    R0,*AR2          ; AMAX = 0
IDON     LDI    @J,R1
         SUBI   1,R1
         ADDI   @NIPIVO,R1
         STI    R1,@DEG
         LDI    @DEG,AR5
         LDI    *AR5,R1
         CMPI   1,R1            ; Has this row been checked ?
         BN     ADON            ; If no, go to check it.
         BNZ    ICONT
         BR     JCONT           ; If yes, go to next row.
/ADON    LDI    @NAMAX,AR2
         LDF    *AR2,R6         ; Load AMAX to the R6
         LDF    *AR1,R7         ; Load A(J,I) to R7
         ABSF   R6
         ABSF   R7
         CMPF   R6,R7           ; Check [abs( A(J,I) ) - abs(AMAX)]
         BN     JCONT           ; If AMAX is bigger, then go to next row.
(CHANGE  LDI    @I,R1           ; If A(J,I) is bigger, then change AMAX
         STI    R1,@ICOL        ; with A(J,I) and store I to ICOL and J
         LDI    @J,R1           ; to IROW
         STI    R1,@IROW
         LDF    *AR1,R1
```

(Continued)

```
        STF     R1,*AR2
JCONT   ADDI    AR0,AR1
        LDI     @J,R1
        ADDI    1,R1
        STI     R1,@J
        LDI     @NR,R1
        SUBI    1,R1
        STI     R1,@NR          ; Have all rows been checked ?
        BP      DON             ; If no, go to next row.

CAIPIV  LDI     @ICOL,R1        ;
        SUBI    1,R1
        ADDI    @NIPIVO,R1
        STI     R1,@DEG
        LDI     @DEG,AR4
        LDI     *AR4,R1
        ADDI    1,R1
        STI     R1,*AR4         ; Write this row has been checked.
        LDI     @ICOL,R1
        SUBI    @IROW,R1        ; Is J equal I ?
        BZ      CAINDE          ; If yes, go to write to index.
SATIR   LDI     @IROW,R1        ; If no, check the next row.
        SUBI    1,R1
        STI     R1,@DEG
        LDI     @N,R2
        MPYI    R2,R1
        ADDI    @NA,R1
        STI     R1,@DEG
        LDI     @DEG,AR3
        LDI     @NDEGRO,AR4
        SUBI    1,R2
        STI     R2,@DEG1
        LDI     @DEG1,RC
        RPTB    KRT
        LDF     *AR3++(1),R1
KRT     STF     R1,*AR4++(1)
        LDI     @ICOL,R1
        SUBI    1,R1
        STI     R1,@DEG2
        LDI     @DEG2,R2
        MPYI    @N,R2
        ADDI    @NA,R2
        STI     R2,@DEG2
        LDI     @DEG2,AR3
        LDI     @DEG,AR4
        LDI     @DEG1,RC
        RPTB    KRT1
        LDF     *AR3++(1),R1
KRT1    STF     R1,*AR4++(1)
        LDI     @NDEGRO,AR3
        LDI     @DEG2,AR4
        LDI     @DEG1,RC
        RPTB    KRT2
```

(Continued)

```
        LDF   *AR3++(1),R1
]KRT2   STF   R1,*AR4++(1)   ; The row, which has maximum coefficient,
                             ; has been interchanged with ith row.
(CAINDE LDI   @NINDEX,AR4    ; Calculate the index numbers.
        LDI   @IROW,R1
        STI   R1,*AR4++(1)   ; INDEX(I,1) = IROW
        LDI   @ICOL,R2
        STI   R2,*AR4++(1)   ; INDEX(I,2) = ICOL
        STI   AR4,@NINDEX
        SUBI  1,R2
        STI   R2,@DEG
        LDI   @N,R3
        MPYI3 R2,R3,R1
        ADDI  R2,R1
        ADDI  @NA,R1
        STI   R1,@DEG
        LDI   @DEG,AR3
        LDI   @NPIV,AR4
        LDF   *AR3,R1        ; R1 = A(ICOL,ICOL)
        STF   R1,*AR4
        LDF   1.0,R1
        STF   R1,*AR3        ; A(ICOL,ICOL) = 1.
        LDF   *AR4,R3        ; PIVOT = A(ICOL,ICOL)
        BZ    FT             ; If PIVOT=0, then go to end.
        CALL  INVF           ; Calculate 1/PIVOT
        LDI   1,R1
        STI   R1,@L          ; L = 1
        LDI   @ICOL R1
        SUBI  1,R1
        MPYI  @N,R1
        ADDI  @NA,R1
        STI   R1,@DEG
        LDI   @DEG,AR3
INEWCO  LDF   *AR3,R2        ; R2 = A(ICOL,L)
        MPYF3 R2,R3,R4       ; R4 = A(ICOL,L) / PIVOT
        STF   R4,*AR3++(1)
        LDI   @L,R1
        ADDI  1,R1
        STI   R1,@L
        CMPI  @N,R1
        BP    ANY
        BR    NEWCOL
/ANY    LDI   1,R1
        STI   R1,@L1         ; L1 = 1
/ANADON LDI   @L1,R1
        CMPI  @ICOL,R1       ; Is this row ICOLth row ?
        BZ    FIVEN1         ; If yes, go to next row.
        SUBI  1,R1
        LDI   @N,R2
        MPYI  R2,R1
        ADDI  @ICOL,R1
        SUBI  1,R1
        ADDI  @NA,R1
```

(Continued)

```
        STI   R1,@DEG1
        LDI   @DEG1,AR3
        LDF   *AR3,R1
        STF   R1,@NTT         ; T = A(L1,ICOL)
        STF   R0,*AR3
        LDI   1,R1
        STI   R1,@L           ; L=1
        LDI   @ICOL,R1
        SUBI  1,R1
        STI   R1,@DEG1
        LDI   @L1,R1
        SUBI  1,R1
        STI   R1,@DEG2
        LDI   @N,R2
        MPYI  @DEG1,R2
        ADDI  @L,R2
        ADDI  @NA,R2
        SUBI  1,R2
        STI   R2,@DEG1
        MPYI  @N,R1
        ADDI  @L,R1
        SUBI  1,R1
        ADDI  @NA,R1
        STI   R1,@DEG2
        LDF   @NTT,R2
        LDI   @DEG1,AR3
        LDI   @DEG2,AR4
DONGER  MPYF3 *AR3++(1),R2,R4 ; R4 = A(ICOL,L) * T
        LDF    *AR4,R1        ; R1 = A(L1,L)
        SUBF   R4,R1          ; R1 = A(L1,L) - A(ICOL,L) * T
        STF    R1,*AR4++(1)
        LDI    @L,R1
        ADDI   1,R1
        STI    R1,@L
        CMPI   @N,R1
        BLS    DONGER
FIVENN  LDI    @L1,R1
        ADDI   1,R1
        STI    R1,@L1
        CMPI   @N,R1
        BLS    ANADON         ;
ICONT   LDI    @I,R1
        ADDI   1,R1
        STI    R1,@I          ; Increase the row number
        LDI    @NP,R1
        SUBI   1,R1
        STI    R1,@NP
        BZ     SON            ; Have all rows been eliminated ?
        SUBI   @NN,AR1        ; If no, start the next row.
        BR     YUZ
SON     LDI    @N,R1          ; If yes, check the column with INDEX
        SUBI   1,R1
        STI    R1,@DEG2
```

(Continued)

```
        STI    R1,@NTT
        LDI    1,R1
        STI    R1,@I
NNEWL   LDI    @N,R1
        SUBI   @I,R1
        ADDI    1,R1
        STI    R1,@L
        SUBI   1,R1
        MPYI   2,R1
        ADDI   @INDEX,R1
        STI    R1,@DEG1
        LDI    @DEG1,AR1
        LDI    *AR1++(1),R1
        SUBI   *AR1--(1),R1 ; Is INDEX(L,1) equal to INDEX(L,2)
        BZ     NZERO        ; If yes, check to next index.
        LDI    *AR1++(1),R1 ; If no, change the column.
        STI    R1,@IROW     ; IROW = INDEX(L,1)
        LDI    *AR1++(1),R1
        STI    R1,@ICOL     ; ICOL = INDEX(L,2)
        LDI    1,R1
        STI    R1,@J
CHCOL   LDI    @J,R1
        SUBI   1,R1
        STI    R1,@DEG1
        LDI    @N,R7
        MPYI3  R7,R1,R4
        ADDI   @IROW,R4
        SUBI   1,R4
        ADDI   @NA,R4
        STI    R4,@DEG1
        MPYI3  R7,R1,R4
        ADDI   @ICOL,R4
        SUBI   1,R4
        ADDI   @NA,R4
        STI    R4,@DEG2
        LDI   @NDEGRO,AR2
        LDI   @DEG1,AR3
        LDI   @DEG2,AR4
        LDF   *AR3,R1
        STF   R1,*AR2       ; NDEGRO = A(J,IROW)
        LDF   *AR4,R1
        STF   R1,*AR3       ; A(J,IROW) = A(J,ICOL)
        LDF   *AR2,R1
        STF   R1,*AR4       ; A(J,ICOL) = A(J,IROW)
        LDI   @J,R1
        ADDI  1,R1
        STI   R1,@J
        CMPI  @N,R1
        BLS   CHCOL
NZERO   LDI   @I,R1
        ADDI  1,R1
        STI   R1,@I
        CMPI  @N,R1
```

(Continued)

```
        BLS  NEWL
        RETS                ; Return the main program
FT      LDI  5,R1
        STI  R1,@MISTA      ; Write " divided by zero "
        RETS                ; Return the main program
```

(Concluded)

A.1.13.

```
             .option X
             .global BEGIN,INIT
             .sect     "init"      ; Named section
RESET        .word     INIT        ; RS- loads address INIT to PC
             .data
MASK         .word  0FFFFFFFFH
BLK0         .word  0809800H     ; Beginning address of RAM blok 0
BLK1         .word  0809C00H     ; Beginning address of RAM blok 1
STCK         .word  0809F00H     ; Beginning of stack
CTRL         .word  0808000H     ; Pointer for peripheral-bus memory
*                                ; map
NEGONE       .word  0FFFFFFH
N            .word  0000002H
P            .word  0000002H
R            .word  0000001H
I            .word  0000000H
J            .word  0000000H
K            .word  0000000H
L            .word  0000000H
L1           .word  0000000H
INDEX        .word  080981DH
IROW         .word  0000000H
ICOL         .word  0000000H
DEGROW       .word  0809830H
NA           .word  0000000H
DEG          .word  0000000H
DEG1         .word  0000000H
DEG2         .word  0000000H
NIPIVO       .word  0809813H
NINDEX       .word  080981DH
NDEGRO       .word  0809830H
NPIV         .word  0809809H
NAMAX        .word  0809852H
D            .word  0000000H
THUN         .word  0000000H
NTT          .word  0000000H
DEGSPC       .word  0000000H
TT           .word  0000000H
MINUS        .word  0000000H
NN           .word  0000000H
NM1          .word  0000000H
NM2          .word  0000000H
NM3          .word  0000000H
NPR          .word  0000000H
NR           .word  0000000H
NLI          .word  0000000H
KR           .word  0000000H
TR           .word  0000000H
MNR          .word  0000000H
T            .word  0000000H
PN           .word  0000000H
NP           .word  0000000H
AMAX         .word  0809852H
```

(Continued)

```
M5          .word  0809853H
M6          .word  0809887H
M7          .word  08098BBH
MK1         .word  08098C0H
MK2         .word  08098C4H
MK3         .word  08098C8H
MK4         .word  08098CCH
XSMAT       .word  0809900H
TIME        .word  0809A30H
PMAT        .word  0809A02H
XPMAT       .word  0809A08H
UP          .word  0809A06H
XHMAT       .word  0809A50H
UHMAT       .word  0809A52H
ITAD        .word  0809BB0H
ITA         .word  0809BD8H
AD          .word  0809AD0H
PAR         .word  0000000H
VAR         .word  0000000H
CONT        .word  0000000H
PRLIO       .word  0808081H
PRLIOO      .word  0808080H
ERRM        .float 0.001000
PMA         .word  08098D0H
HO          .word  0000000H
OUTPUT      .word  0808082H
CHECK       .word  0000000H
MISTA       .word  0000000H
LAST        .word  0000000H
INPUT       .word  0000000H
TIME1       .float 0.989200
STEP        .float 0.040000
TEMPM       .word  0809D80H
SABIT       .word  0809800H
SABIT1      .word  0809802H
BTI         .word  0809950H
PTI         .word  0809955H
XHM         .word  080995AH
XHMT        .word  0809973H
SUMPAY      .word  080998CH
SPAYDA      .word  0809DA0H
STEPN       .word  0000000H
FIRSAT      .word  0000000H
SAY         .word  0000000H
SAY1        .word  0000000H
TCDD        .word  0000000H
*
*
                  .text
INIT        LDI   80H,DP
            LDI   1800H,ST
            LDI   0809H,AR1
            LSH   4,AR1
```

(Continued)

```
        ADDI   8,AR1
        LSH    8,AR1
        STI    AR1,@BLK0
        LDI    600H,R6
        ADDI   R6,AR1
        STI    AR1,@BLK1
        LDI    AR1,R5
        ADDI   100H,R5
        STI    R5,@STCK
        SUBI   100H,R5
        LDI    -1,R7
        STI    R7,@MASK
        RPTS   4
        SUBI   R6,R5
        STI    R5,@CTRL
        LDI  @MASK,IE
        LDI  @BLK0,AR0
        LDI  @BLK1,AR1
        ADDI 5,AR1
        LDF  0.0,R0
        RPTS 1023
        STF  R0,*AR0++(1)
        RPTS 511
        STF  R0,*AR0++(1)
        RPTS 506
        STI  R0,*AR1++(1)
        LDI    @CTRL,AR0
        STI    R0,*+AR0(0)
        STI    R0,*+AR0(32)
        STI    R0,*+AR0(48)
        STI    R0,*+AR0(64)
        STI    R0,*+AR0(66)
        STI    R0,*+AR0(67)
        STI    R0,*+AR0(68)
        STI    R0,*+AR0(80)
        STI    R0,*+AR0(82)
        STI    R0,*+AR0(83)
        STI    R0,*+AR0(84)
        STI     R0,*+AR0(96)
        STI    R0,*+AR0(100)
        LDI    @STCK,SP
        OR     2000H,ST
        BR     BEGIN


         .text
         .global BEGIN
BEGIN    .set $
        LDI    @BLK0,AR0
        LDI    @BLK1,AR1
        LDI    @MASK,R1
        STI    R1,*+AR1(5)
        LDI    2,R1
```

(Continued)

```
STI    R1,*+AR1(6)
STI    R1,*+AR1(7)
LDI    1,R1
STI    R1,*+AR1(8)
STI    AR0,*+AR1(80)
ADDI   2,AR0
STI    AR0,*+AR1(81)
ADDI   7,AR0
STI    AR0,*+AR1(25)
ADDI   10,AR0
STI    AR0,*+AR1(22)
ADDI   10,AR0
STI    AR0,*+AR1(23)
STI    AR0,*+AR1(14)
ADDI   19,AR0
STI    AR0,*+AR1(24)
STI    AR0,*+AR1(17)
ADDI   34,AR0
STI    AR0,*+AR1(26)
STI    AR0,*+AR1(46)
ADDI   1,AR0
STI    AR0,*+AR1(47)
ADDI   34H,AR0
STI    AR0,*+AR1(48)
ADDI   34H,AR0
STI    AR0,*+AR1(49)
ADDI   5,AR0
STI    AR0,*+AR1(50)
ADDI   4,AR0
STI    AR0,*+AR1(51)
ADDI   4,AR0
STI    AR0,*+AR1(52)
ADDI   4,AR0
STI    AR0,*+AR1(53)
ADDI   4,AR0
STI    AR0,*+AR1(70)
ADDI   30H,AR0
STI    AR0,*+AR1(54)
ADDI   0C8H,AR0
STI    AR0,*+AR1(82)
ADDI   5,AR0
STI    AR0,*+AR1(83)
ADDI   5,AR0
STI    AR0,*+AR1(84)
ADDI   19H,AR0
STI    AR0,*+AR1(85)
ADDI   19H,AR0
STI    AR0,*+AR1(86)
ADDI   19H,AR0
STI    AR0,*+AR1(55)
ADDI   1H,AR0
STI    AR0,*+AR1(56)
ADDI   4,AR0
```

(Continued)

```
STI   AR0,*+AR1(58)
ADDI  2,AR0
STI   AR0,*+AR1(57)
ADDI  0C8H,AR0
STI   AR0,*+AR1(59)
ADDI  2,AR0
STI   AR0,*+AR1(60)
ADDI  114H,AR0
STI   AR0,*+AR1(61)
ADDI  0C8H,AR0
STI   AR0,*+AR1(62)
ADDI  0C8H,AR0
STI   AR0,*+AR1(79)
ADDI  10H,AR0
STI   AR0,*+AR1(87)
LDI   3CH,R5
STI   R5,*+AR1(88)
ADDI  30H,AR0
STI   AR0,*+AR1(63)
LDF   0.9892,R1
LDI   0C36H,R2
LSH   8,R2
ADDI  R2,R1
STF   R1,*+AR1(77)
LDF   0.04,R1
LDI   070AH,R2
LSH   8,R2
ADDI  R2,R1
STF   R1,*+AR1(78)
LDI   @CTRL,R2
SUBI  4000H,R2
STI   R2,*+AR1(67)
ADDI  5DD0H,R2
STI   R2,*+AR1(68)
ADDI  10H,R2
STI   R2,*+AR1(72)
LDI   @TIME,AR1
LDF   @STEP,R1
STF   R1,*AR1
LDI   @SABIT,AR2
LDI   0FFH,R2
LSH   24,R2
STI   R2,*AR2
LDI   80H,R1
LSH   16,R1
STI   R1,*+AR2(1)
STI   R1,*+AR2(3)
SUBI  1,R1
STI   R1,*+AR2(4)
LSH   8,R1
ADDI  0FFH,R1
STI   R1,*+AR2(2)
STI   R0,@TT
```

(Continued)

```
        STI   R0,@LAST
*
*
* MAIN PROGRAM
*
*
ESK     LDI  80H,DP
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* DATA ARE READ AND PLACED                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        CALL DMA
        LDI  @PRLIOO,AR4
        LDI  *AR4++(1),R1
        LSH  16,R1
        LSH  -16,R1
        LDI  *AR4++(1),R2
        LSH  16,R2
        ADDI3 R2,R1,R5
        LDI  @SABIT1,AR1
        LDI  @SABIT,AR2
        CALL CMPQ
        LDI   @SAY1,R1
        BNZ   DEV
        PUSH  R5
        POPF  R5
        CMPF  10.,R5
        BNZ   FIRR
        STI   R5,@SAY1
        STI   R0,@TT
        STI   R0,@SAY
        LDI   @PMAT,AR1
        LDF   1.00,R1
        LDF   @TIME1,R2
        STF   R1,*AR1++(1)
        STF   R2,*AR1++(1)
        RPTS  3
        STF   R1,*AR1++(1)
        LDI   @ITA,AR1
        STF   R0,*AR1
        BR    ESK
FIRR    LDI  @SAY,R1
        BNZ  YETER
        BR   ESK
DEV     LDI  @SAY,R1
        CMPI 10,R1
        BP   YETER
        LDI  @TT,R2
        BNZ   R11
        LDI  @ITAD,AR1
        ADDI R2,AR1
        LDI  @XSMAT,AR3
        LDI  @XPMAT,AR6
        PUSH R5
```

(Continued)

```
        POPF R5
        STF  R5,*AR1
        LDI  *AR4++(1),R1
        LDI  *AR4++(1),R2
        LSH  16,R2
        ADDI3 R1,R2,R5
        LDI  @SABIT1,AR1
        CALL CMPQ
        PUSH R5
        POPF R5
        STF  R5,*AR3++(1)
||      STF  R5,*AR6++(1)
        LDI  *AR4++(1),R1
        LDI  *AR4++(1),R2
        LSH  16,R2
        ADDI3 R1,R2,R5
        CALL CMPQ
        PUSH R5
        POPF R5
        STF  R5,*AR3++(1)
||      STF  R5,*AR6++(1)
        BR   R12
*
*
R11     LDI  @ITAD,AR1
        ADDI R2,AR1
        LDI  @XSMAT,AR3
        ADDI @TT,AR3
        ADDI @TT,AR3
        PUSH R5
        POPF R5
        STF  R5,*AR1
        LDI  *AR4++(1),R1
        LDI  *AR4++(1),R2
        LSH  16,R2
        ADDI3 R1,R2,R5
        LDI  @SABIT1,AR1
        CALL CMPQ
        PUSH R5
        POPF R5
        STF  R5,*AR3++(1)
        LDI  *AR4++(1),R1
        LDI  *AR4++(1),R2
        LSH  16,R2
        ADDI3 R1,R2,R5
        CALL CMPQ
        PUSH R5
        POPF R5
        STF  R5,*AR3++(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THE CONTROL INPUT IS DELAYED                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
R12     LDF  0,R0
```

(Continued)

```
        STI   R0,@CHECK
        STI   R0,@MISTA
        STI   R0,@HO
        LDI   @PMA,AR1
        LDF   -10.0,R1
        STF   R1,*AR1
        STI   AR1,@PAR
        LDI   @ITAD,AR1
        LDI   @TT,R2
        ADDI  R2,AR1
        MPYF3 R1,*AR1,R2
        NEGF  R2,R2
        STF   R2,*AR1
        STI   AR1,@CONT
        LDI   @ITA,R1
        ADDI  @TT,R1
        STI   R1,@VAR
        LDI   1,R7
        STI   R7,@N
        STI   R7,@P
        LDI   R7,R1
        PUSH  R1
        STI   R7,@INPUT
        CALL  RUNGE
        POP   R1
        ASH   1,R1
        STI   R1,@P
        STI   R1,@N
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* CALCULATION OF THE CONTROL INPUTS OF HOMOGENEOUS SYSTEMS             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        LDI   @FIRSAT,R1
        BZ    RETRY
        STI   R0,@FIRSAT
        STI   R0,@TT
        STI   R0,@TCDD
RETRY   LDF   0,R0
        STI   R0,@NTT
        STI   R0,@MISTA
        LDI   @UHMAT,AR0
        LDI   @XSMAT,AR1
        LDI   @TT,R2
        ADDI  R2,AR1
        ADDI  R2,AR1
        LDI   @TCDD,R4
        LDI   14H,R5
        MPYI3 R5,R4,R3
        ADDI  R3,AR0
        LDF   *AR1++(1),R1
        STF   R1,*AR0++(1)        ;       UH11=XS1
        STF   R0,*AR0++(3)        ;       UH12=0
        STF   R0,*AR0++(1)        ;       UH21=0
        STF   R1,*AR0++(3)        ;       UH22=XS1
```
(Continued)

```
        LDF   *AR1++(1),R1
        STF   R0,*AR0++(1)        ;       UH31=0
        STF   R1,*AR0++(3)        ;       UH32=XS2
        LDI   @ITA,AR1
        ADDI  R2,AR1
        LDF   *AR1,R1
        STF   R1,*AR0++(1)        ;       UH41=η
        STF   R0,*AR0++(3)        ;       UH42=0
        STF   R0,*AR0++(1)        ;       UH51=0
        STF   R1,*AR0             ;       UH52=η
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* INTEGRATION OF THE PARTICULAR SYSTEM                            *
*                                                                 *
* * * * * * * * * *  M A T R I X  * * * * * * * * * * * * * * * * *
        STI   R0,@NTT
        STI   R0,@HO
        LDI   @PMAT,R1
        STI   R1,@PAR             ; NM1 = 1000
        LDI   @XPMAT,R1
        LDI   @TT,R2
        LDI   R2,R5
        MPYI  2,R2
        ADDI  R2,R1
        STI   R1,@VAR
        LDI   @UP,R1
        STI   R1,@NM1
        LDI   @ITA,R3
        ADDI  R5,R3
        STI   R3,@NM2
        LDI   @TEMPM,R3
        STI   R3,@NM3
        STI   R3,@CONT
        LDI   @R,R1
        STI   R1,@N
        CALL  MAT
        LDI   @P,R1
        STI   R1,@N
        LDI   2,R7
        CALL  RUNGE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* INTEGRATION OF THE HOMOGENEOUS SYSTEMS                           *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        LDI   1,R2
        STI   R2,@HO
GERI    LDI   @NTT,R2
        LDI   4,R3
        MPYI3 R2,R3,R1
        ADDI   @XHMAT,R1
        LDI    @TCDD,R2
        LDI    14h,R3
```

(Continued)

```
        MPYI   R2,R3,R4
        ADDI   R4,R1
        STI    R1,@VAR
        ADDI   2,R1
        STI    R1,@CONT
        LDI    @NTT,R1
        ADDI   1,R1
        STI    R1,@NTT
        LDI    2,R7
        CALL   RUNGE
        LDI    @NTT,R1
        SUBI   5,R1
        BN     GERI
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* CONTROL FOR THREE STEPS                                         *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        LDI   @TT,R1
        ADDI 1,R1
        STI  R1,@TT
        CMPI  3,R1
        BN    SENBAK
        SUBI @STEPN,R1
        BZ   YT
        CALL  TANICI
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* CONTINUE TO NEXT ITERATION                                      *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SENBAK STI  R1,@TCDD
       LDI  @CHECK,R2
       BZ   DIGER
       BNZ   RETRY
       NOP
       NOP
DIGER  LDI   @SAY,R3
       BZ    ESK
       BNZ   RETRY
YT     CALL TANICI
       CALL IDEN
       LDI  @MISTA,R1
       BNZ  YETER
       LDI  @M6,AR2
       LDI  @PMAT,AR1
       LDF  *AR2++(1),R1
       STF  R1,*AR1++(2)
       LDF  *AR2++(1),R1
       STF  R1,*AR1++(1)
       LDF  *AR2++(1),R1
       STF  R1,*AR1++(1)
       LDI  @UP,AR1
```

(Continued)

```
             LDF   *AR2++(1),R1
       STF  R1,*AR1++(1)
       LDF  *AR2++(1),R1
       STF  R1,*AR1++(1)
YT1    LDI   @CHECK,R1
       ADDI  1,R1
       STI   R1,@CHECK
       CMPI  1,R1
       BP    YETER
       LDI   @SUMPAY,AR0
       LDI   @SPAYDA,AR1
       RPTS  4
       STF   R0,*AR0++(1)
       RPTS  24
       STF   R0,*AR1++(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* PREPERATION FOR NEXT STEP                                       *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LDI  @XSMAT,AR1
       LDI  @XPMAT,AR2
       LDF  *AR1++(1),R1
       STF  R1,*AR2++(1)
       LDF  *AR1++(1),R1
       STF  R1,*AR2++(1)
       LDI  @XHMAT,AR4
       RPTS 12H
       STF  R0,*AR4++(1)
       STI  R0,@TT
       STI  R0,@TCDD
       BR   RETRY
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* SEND THE IDENTIFICATION RESULTS                                 *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
YETER  LDI  @PMAT,AR6
       LDI  OFFFH,R7
       LSH  4,R7
       ADDI OFH,R7
       LDI  R7,R6
       LSH  16,R6
       LDI  5,RC
       RPTB OUT
       LDI  @OUTPUT,AR7
       LDI  @SABIT,AR1
       LDF  *AR6++(1),R5
       PUSHF R5
       POP   R5
```

(Contimued)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                  *
* CONVERT THE RESULTS   TO IEEE FLOATING POINT FORMAT              *
*                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       CALL   TMSC
       AND3   R5,R7,R1
       STI    R1,*AR7++(1)
       AND3   R5,R6,R1
       LSH    -16,R1
       STI    R1,*AR7--(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                  *
* SEND THE RESULTS TO COMPUTER                                      *
*                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

OUT    CALL   DMAO

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                  *
* SHIFT THE DATA FOR NEXT STEP                                     *
*                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LDI   @XSMAT,AR0
       LDI   2,R1
       ADDI3 R1,AR0,AR1
       LDI   @STEPN,R2
       MPYI3 R2,R1,R3
       SUBI 1,R3
       LDI   R3,RC
       RPTB GULUM
       LDF   *AR1++(1),R5
GULUM  STF   R5,*AR0++(1)
       LDI   @ITA,AR0
       LDI   1,R1
       ADDI3 R1,AR0,AR1
       LDI   @STEPN,R3
       SUBI 1,R3
       LDI   R3,RC
       RPTB GULUM2
       LDF   *AR1++(1),R5
GULUM2 STF   R5,*AR0++(1)
       LDI    @SUMPAY,AR0
       LDI    @SPAYDA,AR1
       RPTS   4
       STF    R0,*AR0++(1)
       RPTS   24
       STF    R0,*AR1++(1)
       LDI   @XSMAT,AR1
       LDI   @XPMAT,AR2
       LDF   *AR1++(1),R1
       STF   R1,*AR2++(1)
```

(Continued)

```
       LDF   *AR1++(1),R1
       STF   R1,*AR2++(1)
       LDI   @XHMAT,AR4
       RPTS 12H
       STF   R0,*AR4++(1)
       LDI   @TT,R6
       SUBI 1,R6
       STI   R6,@TT
       LDI   15,R4
       STI   R4,@FIRSAT
       LDI   @SAY,R6
       ADDI 1,R6
       STI   R6,@SAY
       CMPI 20,R6
       BP    BUYET
       BR    ESK
       NOP
       NOP
BUYET  STI   R0,@SAY1
       BR  ESK
DUR    IDLE


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* IDENTIFICATION SUBROUTINE                                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
IDEN   LDI  @N,R1
       ADDI 3,R1
       STI  R1,@NN
       STI  R1,@P
       STI  R1,@N
       LDI  @AD,AR1
       STI  AR1,@NA     ; NA INCLUDES THE BEGINING ADDRESS OF [h(t)]
       LDI  @SPAYDA,AR2
       LDI  24,RC
       RPTB AKTAR
       LDF  *AR2++(1),R3
AKTAR  STF  R3,*AR1++(1)
       CALL INVER
       LDI   @MISTA,R1
       BNZ   JOHN
       LDI  @AD,R1
       STI  R1,@NM1
       LDI  @SUMPAY,R1
       STI  R1,@NM2
       LDI  @M6,R1
       STI  R1,@NM3
       STI  R0,@MINUS
       CALL MAT
       LDI  @M5,AR1
       LDI  @PMAT,AR2
```

(Continued)

```
        LDF   *AR2++(2),R1
        STF   R1,*AR1++(1)
        LDF   *AR2++(1),R1
        STF   R1,*AR1++(1)
        LDF   *AR2++(1),R1
        STF   R1,*AR1++(1)
        LDI   @UP,AR2
        LDF   *AR2++(1),R1
        STF   R1,*AR1++(1)
        LDF   *AR2++(1),R1
        STF   R1,*AR1++(1)
        LDI   @M6,R1
        STI   R1,@NM1
        LDI   @M7,R2
        STI   R1,@NM3
        LDI   @M5,R1
        STI   R1,@NM2
        STI   R0,@MINUS
        LDI   1,R1
        STI   R1,@N
        STI   R1,@R
        CALL  MAD
JOHN    LDI   @P,R1
        SUBI  3,R1
        STI   R1,@P
        STI   R1,@N
        RETS

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* INTEGRATION SUBROUTINE                                        *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
RUNGE   LDI    @PAR,R1
        STI    R1,@NM1
        LDI    @VAR,R2
        STI    R2,@NM2
        LDI    @CONT,R3
        LDI    @M5,R5
        STI    R5,@NM3
        PUSH   R1
        PUSH   R2
        PUSH   R3
        PUSH   R5
        CALL   MAT
        POP    R5
        POP    R3
        POP    R2
        STI    R3,@NM1
        STI    R5,@NM2
        STI    R5,@NM3
        LDI    @R,R4
        STI    R4,@N
```

(Continued)

```
        PUSH  R2
        PUSH  R3
        PUSH  R5
        STI   R0,@MINUS
        CALL  MAD
        POP   R5
        POP   R3
        POP   R2
        LDI   @TIME,AR3
        LDI   R5,AR1
        LDI   @MK1,AR2
        LDI   1,RC
        RPTB  HI
        LDF   *AR1++(1),R4
        MPYF  *AR3,R4
HI      STF   R4,*AR2++(1)
        LDI   @MK1,AR1
        LDI   @M6,AR2
        LDI   1,RC
        RPTB  H1
        LDF   *AR1++(1),R4
        MPYF  0.5,R4
H1      STF   R4,*AR2++(1)
        STI   R2,@NM1
        LDI   @M6,R4
        STI   R4,@NM2
        LDI   @M7,R4
        STI   R4,@NM3
        PUSH  R2
        PUSH  R3
        PUSH  R5
        CALL  MAD
        POP   R5
        POP   R3
        POP   R2
        POP   R
        STI   R ,@NM1
        STI   R5,@NM3
        LDI   @M7,R4
        STI   R4,@NM2
        LDI   @[illegible],R4
        STI   R4,@N
        PUSH  R
        PUSH  R2
        PUSH  R3
        PUSH  R5
        CALL  MAT
        POP   R5
        POP   R3
        POP   R2
        STI   R3,@NM1
        STI   R5,@NM2
        STI   R5,@NM3
```

(Continued)

```
        LDI    @R,R4
        STI    R4,@N
        PUSH   R2
        CALL   MAD
        POP    R2
        LDI    @TIME,AR3
        LDI    R5,AR1
        LDI    @MK2,AR2
        LDI    1,RC
        RPTB   HI1
        LDF    *AR1++(1),R4
        MPYF   *AR3,R4
HI1     STF    R4,*AR2++(1)
        LDI    @MK2,AR1
        LDI    @M6,AR2
        LDI    1,RC
        RPTB   H11
        LDF    *AR1++(1),R4
        MPYF   0.5,R4
H11     STF    R4,*AR2++(1)
        STI    R2,@NM1
        LDI    @M6,R4
        STI    R4,@NM2
        LDI    @M7,R4
        STI    R4,@NM3
        PUSH   R2
        CALL   MAD
        POP    R2
        POP    R1
        STI    R1,@NM1
        STI    R5,@NM3
        LDI    @M7,R4
        STI    R4,@NM2
        LDI    @P,R4
        STI    R4,@N
        PUSH   R1
        PUSH   R2
        PUSH   R3
        PUSH   R5
        CALL   MAT
        POP    R5
        POP    R3
        POP    R2
        STI    R3,@NM1
        STI    R5,@NM2
        STI    R5,@NM3
        LDI    @R,R4
        STI    R4,@N
        PUSH   R2
        CALL   MAD
        POP    R2
        LDI    @TIME,AR3
        LDI    R5,AR1
```

(Continued)

```
        LDI   @MK3,AR2
        LDI   1,RC
        RPTB  HI2
        LDF   *AR1++(1),R4
        MPYF  *AR3,R4
HI2     STF   R4,*AR2++(1)
        LDI   @MK3,AR1
        LDI   @M6,AR2
        LDI   1,RC
        RPTB  H12
        LDF   *AR1++(1),R4
H12     STF   R4,*AR2++(1)
        STI   R2,@NM1
        LDI   @M6,R4
        STI   R4,@NM2
        LDI   @M7,R4
        STI   R4,@NM3
        PUSH  R2
        CALL  MAD
        POP   R2
        POP   R1
        STI   R1,@NM1
        STI   R5,@NM3
        LDI   @M7,R4
        STI   R4,@NM2
        LDI   @P,R4
        STI   R4,@N
        PUSH  R1
        PUSH  R2
        PUSH  R3
        PUSH  R5
        CALL  MAT
        POP   R5
        POP   R3
        POP   R2
        POP   R1
        STI   R3,@NM1
        STI   R5,@NM2
        STI   R5,@NM3
        LDI   @R,R4
        STI   R4,@N
        PUSH  R2
        CALL  MAD
        POP   R2
        LDI   @TIME,AR3
        LDI   R5,AR1
        LDI   @MK4,AR2
        LDI   1,RC
        RPTB  HI3
        LDF   *AR1++(1),R4
        MPYF  *AR3,R4
HI3     STF   R4,*AR2++(1)
        LDI   R5,AR1
```

(Continued)

```
        LDI    @MK1,AR2
        LDI    @MK2,AR3
        LDI    @MK3,AR4
        LDI    @MK4,AR5
        LDI    1,RC
        RPTB   HI4
        LDF    *AR2++(1),R1
        LDF    *AR3++(1),R6
        MPYF   2,R6
        LDF    *AR4++(1),R3
        MPYF   2,R3
        LDF    *AR5++(1),R4
        ADDF    R6,R1
        ADDF    R4,R3
        ADDF    R3,R1
        MPYF    0.1666666,R1
HI4     STF     R1,*AR1++(1)
        STI     R2,@NM1
        LDI     @INPUT,R6
        BNZ     W1
        LDI     @HO,R6
        BZ      WRN
        MPYI    14H,R6
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD
        LDI     @P,R1
        STI     R1,@N
        RETSU
W1      LDI     1,R6
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD
        STI     R0,@INPUT
        RETSU
WRN     ADDI    R7,R2
        ADDI    R6,R2
        STI     R2,@NM3
        STI     R5,@NM2
        CALL    MAD
        LDI     @P,R1
        STI     R1,@N
        RETSU

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* MATRIX INVERSE SUBROUTINE                                       *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
INVER      LDI    @AD,R1
           STI    R1,@NA
```

(Continued)

```
        LDI    @INDEX,R1
        STI    R1,@NINDEX
        LDF    0.0,R0
        LDI    @NIPIVO,AR1
        RPTS   @N
        STI    R0,*AR1++(1)
        LDI    1,R1
        STI    R1,@I
YIRRMI  LDI    @NA,AR1
        LDI    @N,R1
        STI    R1,@NP
        STI    R1,@DEGSPC
        LDI    @N,R2
        MPYI3  R2,R2,R4
        SUBI   1,R4
        STI    R4,@NN
YUZZ    LDI    @DEGSPC,AR0
        LDI    1,R1
        STI    R1,@J
        LDI    @N,R1
        STI    R1,@NR
        LDI    @NAMAX,AR2
        STF    R0,*AR2
DONN    LDI    @J,R1
        SUBI   1,R1
        ADDI   @NIPIVO,R1
        STI    R1,@DEG
        LDI    @DEG,AR5
        LDI    *AR5,R1
        CMPI   1,R1
        BN     ADON
        BNZ    ICONT
        BR     JCONT
ADCON   LDI    @NAMAX,AR2
        LDF    *AR2,R6
        LDF    *AR1,R7
        ABSF   R6
        ABSF   R7
        CMPF   R6,R7
        BN     JCONT
CHANGE  LDI    @I,R1
        STI    R1,@ICOL
        LDI    @J,R1
        STI    R1,@IROW
        LDF    *AR1,R1
        STF    R1,*AR2
JCCONT  ADDI   AR0,AR1
        LDI    @J,R1
        ADDI   1,R1
        STI    R1,@J
        LDI    @NR,R1
        SUBI   1,R1
        STI    R1,@NR
```

(Continued)

```
          BP     DON
CAIPIV    LDI    @ICOL,R1
          SUBI   1,R1
          ADDI   @NIPIVO,R1
          STI    R1,@DEG
          LDI    @DEG,AR4
          LDI    *AR4,R1
          ADDI   1,R1
          STI    R1,*AR4
          LDI    @ICOL,R1
          SUBI   @IROW,R1
          BZ     CAINDE
SATIR     LDI    @IROW,R1
          SUBI   1,R1
          STI    R1,@DEG
          LDI    @N,R2
          MPYI   R2,R1
          ADDI   @NA,R1
          STI    R1,@DEG
          LDI    @DEG,AR3
          LDI    @NDEGRO,AR4
          SUBI   1,R2
          STI    R2,@DEG1
          LDI    @DEG1,RC
          RPTB   KRT
          LDF    *AR3++(1),R1
KRT       STF    R1,*AR4++(1)
          LDI    @ICOL,R1
          SUBI   1,R1
          STI    R1,@DEG2
          LDI    @DEG2,R2
          MPYI   @N,R2
          ADDI   @NA,R2
          STI    R2,@DEG2
          LDI    @DEG2,AR3
          LDI    @DEG,AR4
          LDI    @DEG1,RC
          RPTB   KRT1
          LDF    *AR3++(1),R1
KRT1      STF    R1,*AR4++(1)
          LDI    @NDEGRO,AR3
          LDI    @DEG2,AR4
          LDI    @DEG1,RC
          RPTB   KRT2
          LDF    *AR3++(1),R1
KRT2      STF    R1,*AR4++(1)
CAINDE    LDI    @NINDEX,AR4
          LDI    @IROW,R1
          STI    R1,*AR4++(1)
          LDI    @ICOL,R2
          STI    R2,*AR4++(1)
          STI    AR4,@NINDEX
          SUBI   1,R2
```

(Continued)

```
        STI  R2,@DEG
        LDI  @N,R3
        MPYI3 R2,R3,R1
        ADDI R2,R1
        ADDI @NA,R1
        STI  R1,@DEG
        LDI  @DEG,AR3
        LDI  @NPIV,AR4
        LDF  *AR3,R1
        STF  R1,*AR4
        LDF  1.0,R1
        STF  R1,*AR3
        LDF  *AR4,R3
        BZ   FT
        CALL  INVF
        LDI  1,R1
        STI  R1,@L
        LDI  @ICOL,R1
        SUBI 1,R1
        MPYI @N,R1
        ADDI @NA,R1
        STI  R1,@DEG
        LDI  @DEG,AR3
NEWCOL  LDF  *AR3,R2
        MPYF3 R2,R3,R4
        STF   R4,*AR3++(1)
        LDI   @L,R1
        ADDI  1,R1
        STI   R1,@L
        CMPI  @N,R1
        BP    ANY
        BR    NEWCOL
ANY     LDI  1,R1
        STI  R1,@L1
ANADON  LDI  @L1,R1
        CMPI @ICOL,R1
        BZ   FIVENN
        SUBI  1,R1
        LDI  @N,R2
        MPYI R2,R1
        ADDI @ICOL,R1
        SUBI 1,R1
        ADDI @NA,R1
        STI  R1,@DEG1
        LDI  @DEG1,AR3
        LDF  *AR3,R1
        STF  R1,@NTT
        STF  RO,*AR3
        LDI  1,R1
        STI  R1,@L
        LDI  @ICOL,R1
        SUBI 1,R1
        STI  R1,@DEG1
```

(Continued)

```
         LDI   @L1 R1
         SUBI  1,R1
         STI   R1,@DEG2
         LDI   @N,R2
         MPYI  @DEG1,R2
         ADDI  @L,R2
         ADDI  @NA,R2
         SUBI  1,R2
         STI   R2,@DEG1
         MPYI  @N,R1
         ADDI  @L,R1
         SUBI  1,R1
         ADDI  @NA,R1
         STI   R1,@DEG2
         LDF   @NTT,R2
         LDI   @DEG1,AR3
         LDI   @DEG2,AR4
DONGER   MPYF3 *AR3++(1),R2,R4
         LDF   *AR4,R1
         SUBF  R4,R1
         STF   R1,*AR4++(1)
         LDI   @L,R1
         ADDI  1,R1
         STI   R1,@L
         CMPI  @N,R1
         BLS   DONGER
FIVENN   LDI   @L1,R1
         ADDI  1,R1
         STI   R1,@L1
         CMPI  @N,R1
         BLS   ANADON
ICONT    LDI   @I,R1
         ADDI  1,R1
         STI   R1,@I
         LDI   @NP,R1
         SUBI  1,R1
         STI   R1,@NP
         BZ    SON
         SUBI  @NN,AR1
         BR    YUZ

SON      LDI   @N,R1
         SUBI  1,R1
         STI   R1,@DEG2
         STI   R1,@NTT
         LDI   1,R1
         STI   R1,@I
NEWL     LDI   @N,R1
         SUBI  @I,R1
         ADDI  1,R1
         STI   R1,@L
         SUBI  1,R1
         MPYI  2,R1
```

(Continued)

```
        ADDI   @INDEX,R1
        STI    R1,@DEG1
        LDI    @DEG1,AR1
        LDI    *AR1++(1),R1
        SUBI   *AR1--(1),R1
        BZ     NZERO
        LDI    *AR1++(1),R1
        STI    R1,@IROW
        LDI    *AR1++(1),R1
        STI    R1,@ICOL
        LDI    1,R1
        STI    R1,@J
CHCOL   LDI    @J,R1
        SUBI   1,R1
        STI    R1,@DEG1
        LDI    @N,R7
        MPYI3  R7,R1,R4
        ADDI   @IROW,R4
        SUBI   1,R4
        ADDI   @NA,R4
        STI    R4,@DEG1
        MPYI3  R7,R1,R4
        ADDI   @ICOL,R4
        SUBI   1,R4
        ADDI   @NA,R4
        STI    R4,@DEG2
        LDI    @NDEGRO,AR2
        LDI    @DEG1,AR3
        LDI    @DEG2,AR4
        LDF    *AR3,R1
        STF    R1,*AR2
        LDF    *AR4,R1
        STF    R1,*AR3
        LDF    *AR2,R1
        STF    R1,*AR4
        LDI    @J,R1
        ADDI   1,R1
        STI    R1,@J
        CMPI   @N,R1
        BLS    CHCOL
NZERO   LDI    @I,R1
        ADDI   1,R1
        STI    R1,@I
        CMPI   @N,R1
        BLS    NEWL
        RETS
FT      LDI    5,R1
        STI    R1,@MISTA
        RETS
```

(Continued)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                  *
* SUBROUTINE FOR THE MATRIX MULTIPLICATION                         *
*                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MAT    LDI  @N,R1
       STI  R1,@DEG1          ;    DEG1=N
       LDI  @R,R1
       STI  R1,@NR            ;    NR=R
       SUBI 1,R1
       STI  R1,@KR            ;    KR=R-1
       LDI  @P,R1
       STI  R1,@NPR           ;    NPR=P
       LDI  @N,R1
       MPYI @R,R1
       SUBI 1,R1
       STI  R1,@DEG           ;    DEG=N*R
       STI  R1,@MNR           ;    MNR=N*R
       LDI  @NR,IRO
       LDI  @NM1,AR1
       LDI  @NM2,AR2
       LDI  @NM3,AR3
       LDI  @N,R1
       STI  R1,@NN            ;    NN=N
NEW    LDF  0.0,R5        ; IRO A GORE GEREKENLERI DEGISTIR
BAS    MPYF3   *AR1++(1),*AR2++(IRO),R4
       ADDF    R4,R5
       LDI     @NN,R1
       SUBI    1,R1
       STI     R1,@NN
       BP      BAS
       STF     R5,*AR3++(1)
       SUBI    @MNR,AR2
       LDI     @N,R1
       STI     R1,@NN
       SUBI    @DEG1,AR1
       LDI     @NR,R1
       SUBI    1,R1
       STI     R1,@NR
       BP      NEW
       ADDI    @R,R1
       STI     R1,@NR
       LDI     @NM2,AR2
       ADDI    @DEG1,AR1
       LDI     @NPR,R1
       SUBI    1,R1
       STI     R1,@NPR
       BP      NEW
       RETSU
```

(Continued)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* SUBROUTINE FOR THE MATRIX ADDITION                                *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MAD     LDI    @P,R2
        MPYI   @N,R2
        SUBI   1,R2
        STI    R2,@PN
        LDI    @NM1,AR1
        LDI    @NM2,AR2
        LDI    @NM3,AR3
        LDI    @PN,RC
        LDI    @MINUS,R1
        BNZ    NEGA
        RPTB   A1
        LDF    *AR1++(1),R1
        ADDF   *AR2++(1),R1
A1      STF    R1,*AR3++
        RETSU
NEGA    RPTB   A2
        LDF    *AR1++(1),R1
        SUBF   *AR2++(1),R1
A2      STF    R1,*AR3++
        RETSU


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* INVERSE OF THE FLOATING POINT NUMBER                               *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

INVF    LDF    R3,R6
        ABSF   R3
        PUSHF  R3
        POP    R1
        ASH    -24,R1
        NEGI   R1
        SUBI   1,R1
        ASH    24,R1
        PUSH   R1
        POPF   R1
        MPYF   R1,R3,R2
        SUBRF  2.0,R2
        MPYF   R2,R1
        MPYF   R1,R3,R2
        SUBRF  2.0,R2
        MPYF   R2,R1
        MPYF   R1,R3,R2
        SUBRF  2.0,R2
        MPYF   R2,R1
```

(Continued)

```
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        MPYF  R1,R3,R2
        SUBRF 2.0,R2
        MPYF  R2,R1
        RND   R1
        MPYF  R1,R3,R2
        SUBRF 1.0,R2
        MPYF  R1,R2
        ADDF  R2,R1
        RND   R1,R3
        NEGF  R3,R2
        LDF   R6,R6
        LDFN  R2,R3
        RETSU

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                              *
* TMS320C30 TO IEEE FLOATING POINT FORMAT CONVERSION           *
*                                                              *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
TMSC    LDI   *+AR1(1),R2
        LSH   8,R2
        SUBI  R5,R2
        BZ    Z0
        AND3  R5,*AR1,R1
        ASH   -24,R1
        LDI    R1,R2
        AND    1,R2
        ADDI   1,R1
        ASH    -1,R1
        SUBI    1,R1
        ADDI   40H,R1
        LSH    24,R1
        PUSH   R5
        POPF   R5
        BN     T0
        CMPI   0,R2
        BZ     T1
        LSH   -8,R5
```

(Continued)

```
        OR      R1,R5
        RETS
T1      LSH     -8,R5
        OR      *+AR1(1),R5
        OR      R1,R5
        RETS
T0      NEGI    R5
        LSH     -8,R5
        OR      R1,R5
        CMPI    0,R2
        BZ      T3
        LDI     80H,R2
        LSH     24,R2
        OR      R2,R5
        RETS
T3      OR      *+AR1(1),R5
        OR      R1,R5
        LDI     80H,R2
        LSH     24,R2
        OR      R2,R5
        RETS
Z0      LDI     0,R5
        RETS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                  *
* IEEE TO TMS320C30 FLOATING POINT FORMAT CONVERSION                *
*                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CMPQ    CMPI    0,R5
        BZ      Z3
        AND3    R5,*AR1,R1
        AND3    R5,*+AR1(1),R2
        LSH     -24,R1
        SUBI    40H,R1
        ADDI    1,R1
        LSH     1,R1
        CMPI    0,R2
        BNZ     GO
        SUBI    1,R1
GO      LSH     24,R1
        LDI     *+AR1(2),,R3
        AND3    R5,R3,R2
        CMPI    0,R5
        BN      GK
        OR3     R1,R2,R5
        RETS
GK      PUSH    R2
        POPF    R2
        NEGI    R2,R2
        BNZ     KRL
        SUBI    1,R1
        AND     *AR2,R1
        LDI     80H,R2
```

(Continued)

```
        LSH    24,R2
KRL     LSH    -8,R2
        OR3   R1,R2,R5
        RETS
Z3      LDI   *+AR1(1),R1
        LSH   8,R1
        LDI   R1,R5
        RETS

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* TO OBTAIN THE [h(t)]                                          *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
TANICI  LDI @XHM,AR1
        LDI @XHMAT,AR2
        LDI  R0,R5
ULUDAG  LDI  4,RC
        RPTB XHOMO1
        LDF  *AR2++(1),R1
        STF  R1,*+AR1(0)
        LDF  *AR2++(3),R1
        STF  R1,*+AR1(5)
XHOMO1  ADDI 1,AR1
        ADDI 5,AR1
        CMPI 1,R5
        BZ   BAZEN
        ADDI 1,R5
        BR   ULUDAG
BAZEN   LDI  4,RC
        RPTB XHOMO3
        LDF  *AR2++(4),R1
XHOMO3  STF  R1,*AR1++(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* TRANSPOSE OF HOMOGENEOUS MATRIX                                *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        LDI  @XHMT,AR0
        LDI  R0,R5
IYIAL   LDI  @XHM,AR1
        ADDI R5,AR1
        LDI  4,RC
        RPTB TRANSP
        LDF  *AR1++(5),R1
TRANSP  STF  R1,*AR0++(1)
        ADDI  1,R5
        CMPI  5,R5
        BN    IYIAL
```

(Continued)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* LAST THREE STEP OBSERVATIONS (BTI)                            *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LDI  @BTI,AR1
       LDI  @XSMAT,AR2
       LDI  @TT,R2
       SUBI 1,R2
       MPYI 2,R2
       ADDI R2,AR2
       SUBI  4,AR2
       LDI   4,RC
       RPTB  XBT
       LDF   *AR2++(1),R1
XBT    STF   R1,*AR1++(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
* LAST THREE STEP PARTICULAR SYSTEM OUTPUTS (PTI)               *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LDI  @PTI,AR1
       LDI  @XPMAT,AR2
       LDI  @TT,R2
       SUBI 1,R2
       MPYI 2,R2
       ADDI R2,AR2
       SUBI  4,AR2
       LDI   4,RC
       RPTB  XPT
       LDF   *AR2++(1),R1
XPT    STF   R1,*AR1++(1)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*  TO OBTAIN SUMPAY                                             *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LDI   @BTI,R1
       STI   R1,@NM1
       LDI   @PTI,R1
       STI   R1,@NM2
       LDI   @M6,R1
       STI   R1,@NM3
       LDI   @P,R6              ;R6=P
       ADDI  3,R6
       STI   R6,@P
       LDI   1,R7              ;R7=1
       STI   R7,@N
       LDI   @NEGONE,R1
       STI   R1,@MINUS
       CALL  MAD
       STI   R0,@MINUS
       STI   R6,@N              ; R6=P
```

(Continued)

```
        LDI    @XHMT,R1
        STI    R1,@NM1
        LDI    @M6,R1
        STI    R1,@NM2
        LDI    @M7,R1
        STI    R1,@NM3
        CALL   MAT
        LDI    @SUMPAY,R1
        STI    R1,@NM1
        STI    R1,@NM3
        LDI    @M7,R1
        STI    R1,@NM2
        STI    R7,@N                    ;R7=1
        CALL   MAD
        STI    R6,@N                    ; R6=P
*********************************
*                               *
* TO OBTAIN SPAYDA              *
*                               *
*********************************
        LDI    @XHMT,R1
        STI    R1,@NM1
        LDI    @XHM,R1
        STI    R1,@NM2
        LDI    @M7,R1
        STI    R1,@NM3
        STI    R6,@R
        CALL   MAT
        LDI    @SPAYDA,R1
        STI    R1,@NM1
        LDI    @M7,R2
        STI    R2,@NM2
        STI    R1,@NM3
        CALL   MAD
        SUBI   3,R6
        STI    R6,@P
        STI    R6,@N
        STI    R7,@R
        LDI    @XHMAT,AR1
        LDI    14H,R2
        ADDI3  R2,AR1,AR2
        LDI    3BH,RC
        RPTB   SHN
        LDF    *AR2++(1)),R1
SHN     STF    R1,*AR1++(1)
        LDI    2,R1
        STI    R1,@TCDD
        RETS
```

(Continued)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* READ FROM COMPUTER VIA COMMUNICATION INTERFACE                   *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DMA      LDI  @PRLIO,AR1
         LDI  @PRLIOO,AR2
         LDI  20H,IOF
         LDI  5,RC
         RPTB DV
XFO      LDI  IOF,R1
         AND  8,R1
         BZ   XFO
         LDI  *AR1,R2
         STI  R2,*AR2++(1)
         STI  R0,*+AR1(2)
DV       STI  R0,*+AR1(1)
         RETS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                 *
* WRITE TO COMPUTER VIA COMMUNICATION INTERFACE                   *
*                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

DMAO     LDI  @PRLIO,AR1
         LDI  @OUTPUT,AR2
         LDI  60H,IOF
         LDI  *AR2++(1),R2
         STI   R2,*AR1
         STI  R1,*+AR1(1)
CV       LDI  IOF,R1
         AND  8,R1
         BZ   CV
         STI  R0,*+AR1(2)
         LDI  *AR2++(1),R2
         STI   R2,*AR1
         STI  R1,*+AR1(1)
CV1      LDI  IOF,R1
         AND  8,R1
         BZ   CV1
         STI  R0,*+AR1(2)
         LDI  R0,IOF
         RETS
```

(Concluded)

A.1.14. Longitudinal Transfer Functions of Raven 201

Assumed conditions (Sea-level, ISA +0°):

Mass = 75 kg

Speed = 25 m/s

Flap = 30°

Propeller effect and small descent gradient for $\theta \cong 0^\circ$ are assumed to be negligible.

The equations of the longitudinal motion have been given with Eqs (3.38) to (3.40)

$$m\dot{u} - \mathring{X}_u u - \mathring{X}_w w - \mathring{X}_{\dot{w}} \dot{w} + (mW_e - \mathring{X}_q)q + mg_1\theta = \mathring{X}(t) \qquad (3.38)$$

$$-\mathring{Z}_u u + (m - \mathring{Z}_{\dot{w}})\dot{w} - \mathring{Z}_w w - (mU_e + \mathring{Z}_q)q + mg_2\theta = \mathring{Z}(t) \qquad (3.39)$$

$$-\mathring{M}_u u - \mathring{M}_{\dot{w}} \dot{w} - \mathring{M}_w w + I_y \dot{q} - \mathring{M}_q q = \mathring{M}(t) \qquad (3.40)$$

Non-dimensional aerodynamic dervatives have been calculated and put in the Eqs(3.38) to (3.40)

$$270 \frac{\partial}{\partial t} \hat{u} + 0.21\hat{u} - 0.705\alpha + 1.06\theta = 0$$

$$2.12\hat{u} + 268.6 \frac{\partial}{\partial t} \alpha + 3.85\alpha - 265.7 \frac{\partial}{\partial t} \theta = - 0.312\eta'$$

$$2.8\alpha + 4.36 \frac{\partial}{\partial t} \alpha + 2.22 \frac{\partial^2}{\partial t^2} \theta + 13.2 \frac{\partial}{\partial t} \theta = -0.96\eta'$$