CCSE TECHNICAL REPORT - CCSE-007

"MICROPROFESSOR CONTROL OF HIGH PERFORMANCE TACTICAL MISSILES"

BY

Dr. M. Elshafei Ahmed   Dr. Talal M. Bakri
Assistant Professors
Dept. of Systems Engg.   Dept. of Electrical Engg

COLLEGE OF COMPUTER SCIENCES AND ENGINEERING
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia

TABLE OF CONTENTS

MICROPROCESSOR CONTROL OF HIGH PERFORMANCE TACTICAL MISSILES

Dr. M. Elshafei Ahmed Dr. Talal M. Bakri
Systems Engineering Department Electrical Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

ABSTRACT

The next generation of high performance tactical missiles requires rapid, precise, and large angle rotational maneuvers. The equations of motion representing large axis maneuvers are coupled and nonlinear. This report addresses feasibility issues of implementing a recent control technique of missiles autopilot based on a member of intel 8096 family of high performance single chip microcontrollers. A nonlinear adaptive technique is adopted in this study in order to accommodate the uncertainty and nonlinearity in the missile dynamics, as well as the unknown environmental disturbance torques. A block diagram of the hardware and a detailed algorithm flow chart are presented. The described system implements a full attitude reference system using a quaternion parametrization. The quaternion is updated by a recent norm and orthogonality preserving algorithm. All the computations were analysed to minimize the execution steps and the memory requirements. The majority of equations were scaled for fast and efficient implementation by fixed point arithmetic. It is shown that the microcontroller is not fully utilized, and further functions as guidance and communications could be incorporated in the hardware board. The hardware and software framework established in this study can be used for developing and implementing other control strategies as well.

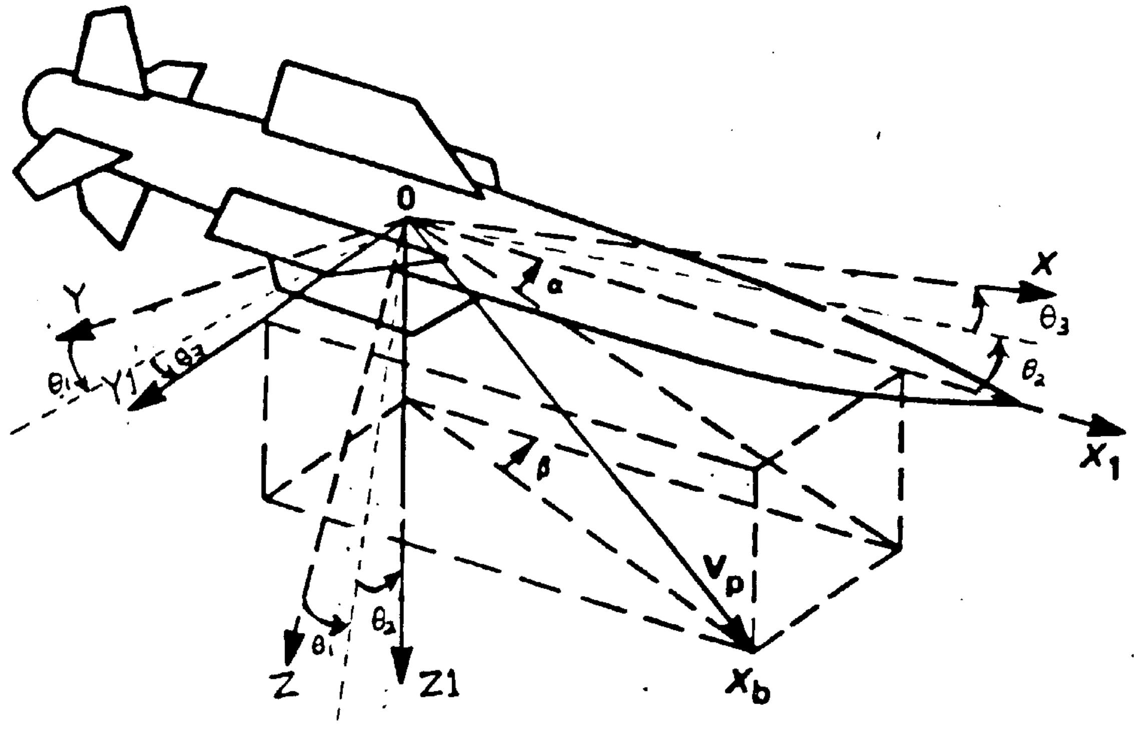


Figure 1. Rocket and reference coordinate system

I. INTRODUCTION

The last few years have witnessed a rapidly expanding horizon of microcontroller-based applications with ever increasing task complexity, mainly due to the availability of inexpensive high performance single chip microcontrollers as Intel 96 series, 8051 series, 80186, NEC V25, and Texas Instruments TMS320 series. The structure of such controllers integrates many of the hardware components for real time control applications, and makes them ideal for stand-alone dedicated tasks. In this study, we selected Intel 8797, a member of the 8096 Microcontrollers [1] (Trade mark named MS96 family). The microcontroller includes 232 byte register file, 8 k of EPROM/ROM, five 8-bit I/O ports, pulse width modulated output, eight 10-bit analog channels, a full duplex serial port, hardware and software timers, and 16-bit multiplications and divisions.

Missile flight control is a complex task which has traditionally been solved using technology based on analog and digital circuits, electromechanical elements, and pneumatic elements [2,3,4,5]. The new generation of microcontrollers can provide a digital control solution with substantial miniaturization of the hardware, and with software capability with a degree of sophistication that would have been impossible to consider for short-range tactical missiles.

Traditionally, the design of a missile control system [2-8] proceeds by first defining the typical speed or Mack number, altitude and other conditions at which the missile will operate. Then, the various aerodynamic coefficients are linearized near zero incidences. A linear model and a control system can then be derived on the assumption that the missile will exercise small perturbations about zero incidences. The calculations are repeated for as many combinations of incidences, missile parameters , and operating conditions as judgement deems advisable.

However, the next generation of high performance tactical missiles requires rapid, precise, and large angle rotational maneuvers. The equations of motion representing large axis maneuvers are coupled and nonlinear. Recently several interesting control techniques have been proposed [9,10,11] for handling such large angle reorientation/slew maneuvers.

In particular, the nonlinear adaptive technique proposed in [9] is adopted in this study. The advantages of this control scheme are the ability to handle the highly nonlinear dynamics of the missile, and its ability to accommodate the uncertainty in the missile parameters and unknown environmental disturbance torques. The control technique enables the missile attitude $\underline{\theta}$ to assymtotically track a model reference output $\hat{\underline{\theta}}$, where

$$\underline{\theta} = (\theta_1, \theta_2, \theta_3)^T$$

are the missile Roll, Pitch and Yaw, respectively, as shown in Fig. 1. The missile autopilot investigated in this study may be described with the aid of Fig. 2. The missile attitude is determined from three gyros which provide signals proportional to the angular rate of rotation

$$\underline{\omega} = (\omega_1, \omega_2, \omega_3)^T$$

around the body axis x_1 , y_1, and z1 as shown in Fig. 1.

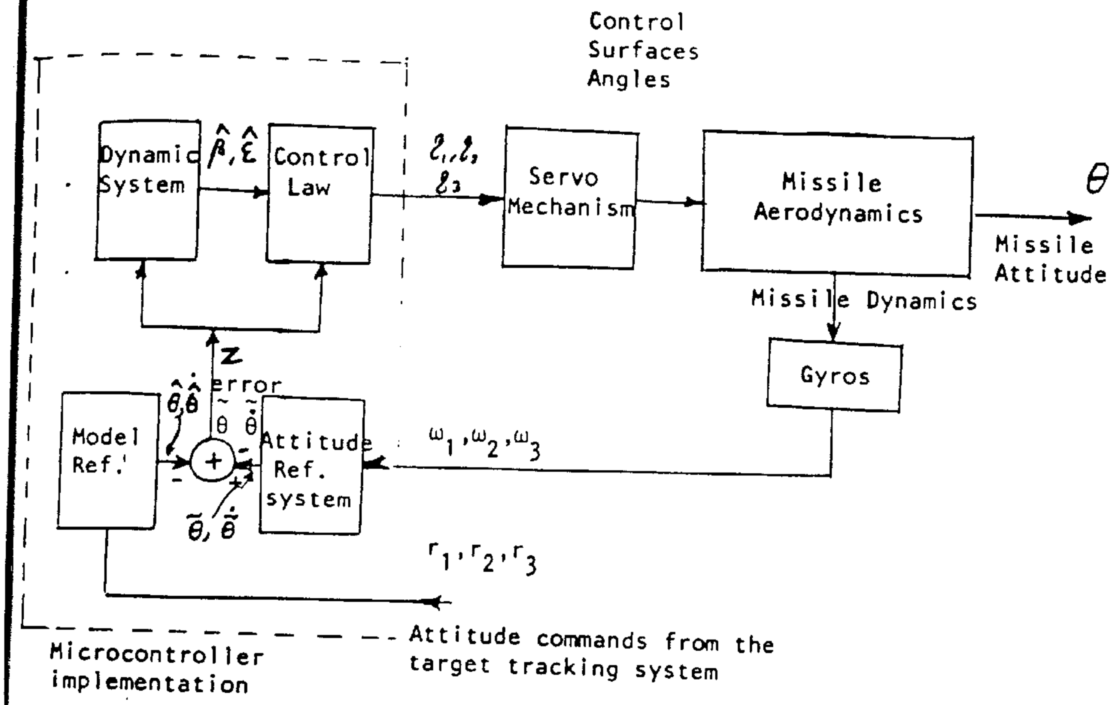
The attitude reference system then estimates the missile orientation $\underline{\theta}$ with respect to some initial position using the gyros output. In the mean time, the desired orientaion $\hat{\underline{\theta}}$, according to target tracking set points, are obtained from a model reference.

The error vector $\underline{Z}$ between the desired orientation and the estimated orientation, $\underline{\hat{\theta}}$ is used to generate the control signals which deflect the Elevator, Rudder, and Aileron control surfaces. However, throughout the paper the estimated orientation will be taken simply as the true orientation. Accordingly, the error vector will be defined by

$$Z = (\underline{\theta} - \underline{\hat{\theta}}, \underline{\dot{\theta}} - \underline{\dot{\hat{\theta}}})^T$$

The control law used in this study is nonlinear, and its parameters are continuously estimated and updated from a dynamic system block, as depicted in Fig. 2.

In section II, we present a brief introduction to the missile dynamics, its aerodynamic derivatives, and the attitude reference parameterization. In section III we review some of the techniques for autopilot design and present the nonlinear adaptive control method to be implemented in this study. section IV describes in detail the implemented control algorithm. Finally, in section V, we discuss the hardware configuration, the overall algorithm, and estimate the computation time and the microcontroller utilization.



Control Surfaces Angles
Dynamic System
$\hat{\beta}, \hat{\varepsilon}$
Control Law
$\delta_1, \delta_2, \delta_3$
Servo Mechanism
Missile Aerodynamics
θ
Missile Attitude
Missile Dynamics
Gyros
z
error
$\hat{\theta}, \hat{\dot{\theta}}$
$\tilde{\theta}$ $\tilde{\dot{\theta}}$
Model Ref.
+
Attitude Ref. system
$\omega_1, \omega_2, \omega_3$
$\tilde{\theta}, \dot{\tilde{\theta}}$
r_1, r_2, r_3
Attitude commands from the target tracking system
Microcontroller implementation

Figure 2. Missile attitude control system.

II. PRELIMINARIES

II.1 Missile Dynamic

In order to study the response or impose control actions on tactical missiles the appropriate Euler's equations of motion are needed. The general motion of a rigid missile has six degrees of freedom; three force equations and three moment equations. Let x, y, z be the reference axis in space with its origin at the center of gravity of the missile, and x_1, y_1, z_1 be the body fixed axis of the missile, as shown in Fig. 1. Let f_1, f_2, f_3 be the forces acting on the missile, and l_1, l_2, l_3 be the sum of aerodynamic and disturbance torques. Let v_1, v_2, v_3 be the velocity components along the x_1, y_1, z_1 fixed body axis, and let $(\omega_1, \omega_2, \omega_3)$ be the angular velocities of the missile around the fixed body axis. f_1 is the sum of the propulsive and drag, while f_2 and f_3 include the aerodynamic normal and lift forces as well as other disturbances of unpredictable nature and depend on the incidence angle, air density, altitude, and Mack number. v_1, the missile velocity along the x_1-axis, is a large positive quantity U, changing at most only a few percent per second. The components of velocity along the pitch and yaw axis, however, tend to be much smaller quantities which can be positive or negative and can have much larger rates of change. The direction of the velocity vector with respect to the missile axis is described with the aid of two angles; the angle of attack

$$\alpha = arctan\left(\frac{v_3}{U}\right) \approx \frac{v_3}{U}$$

and the side slip angle

$$\beta = arctan\left(\frac{v_2}{U}\right) \approx \frac{v_2}{U}$$

The matrix form of the dynamical equations of the missile are given by [13]:

$$m\dot{\underline{v}} + m\tilde{\omega}\underline{v} = \underline{f} \qquad (2.1.1)$$

$$I\dot{\underline{\omega}} + \tilde{\omega}I\underline{\omega} = \underline{L} \qquad (2.1.2)$$

where m is the missile mass, with $\underline{f} = (f_1, f_2, f_3)^T$, $\underline{L} = (l_1, l_2, l_3)^T$, $\underline{\omega} = (\omega_1, \omega_2, \omega_3)^T$, and $\underline{v} = (v_1, v_2, v_3)^T$. The square matrix $\tilde{\omega}$ is formed from the vector $\underline{\omega}$ as follows:

$$\tilde{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \qquad (2.1.3)$$

and $I \in R^{3\times3}$ is the inertia matrix given by

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \qquad (2.1.4)$$

Normally, the X-Z plane is a plane of symmetry, hence Ixy = Iyx = Iyz = Izy = ∅ . However, Ixz = Izx may not be zero.

When the X-Z is a plane of symmetry, the force and moment equations break down to the following equations [4,5]

$$m(\dot{v}_1 + \omega_2 v_3 - \omega_3 v_2) = X - mg\sin(\theta_2) + f_x \qquad (2.1.5)$$

$$m(\dot{v}_2 + \omega_3 v_1 - \omega_1 v_3) = Y + mg\cos(\theta_2)\sin(\theta_1) + f_y \qquad (2.1.6)$$

$$m(\dot{v}_3 - \omega_2 v_1 + \omega_1 v_2) = Z + mg\cos(\theta_2)\cos(\theta_1) + f_z \qquad (2.1.7)$$

$$I_{xx}\dot{\omega}_1 - (I_{yy} - I_{zz})\omega_2\omega_3 - I_{xz}(\omega_1\omega_2 + \dot{\omega}_3) = L + L_x \qquad (2.1.8)$$

$$I_{yy}\dot{\omega}_2 - (I_{zz} - I_{xx})\omega_3\omega_1 + I_{xz}(\omega_1^2 - \omega_3^2) = M + L_y \qquad (2.1.9)$$

$$I_{zz}\dot{\omega}_3 - (I_{xx} - I_{yy})\omega_1\omega_2 + I_{xz}(\omega_2\omega_3 - \dot{\omega}_1) = N + L_z \qquad (2.1.10)$$

Where X, Y, and Z are the aerodynamic forces; L, M, and N are the are the aerodynamic moments, and

f_x, f_y, f_z, L_x, L_y, and L_x

are disturbances .

II.2 Control Surfaces:

Let the deflection of the control surfaces of a typical tail-controlled missile be denoted by

$$\xi_1, \xi_2, \xi_3, \text{and}\, \xi_4$$

The deflections are positive if clockwise looking outwards along the individual hinge axis. For the crucified missile shown in Fig. 3, the control surfaces deflection angles are defined below.

Aileron deflection

$$\alpha_r = \frac{1}{2}(\xi_1 + \xi_3)$$

Elevator deflection

$$\alpha_p = \frac{1}{2}(\xi_1 - \xi_3)$$

Rudder deflection

$$\alpha_y = \frac{1}{2}(\xi_2 - \xi_4)$$

Here we shall assume that ξ_2 and ξ_4 act differentially, hence

$$\alpha_y = \xi_2 = -\xi_4$$

Aileron deflection produces moments about the x_1 axis, and hence, changes the missile roll. Positive elevator deflection produces negative force in the z_1 direction and anticlock wise moment about the y_1 axis (pitch). Positive Rudder deflection produces a positive force in the y_1 direction and negative moment about the z_1 axis, causing changes in the missile yaw.

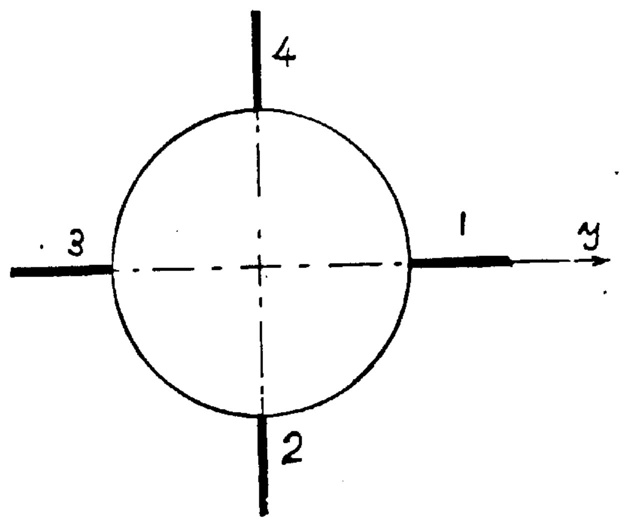


Fig.3, The control surfaces as seen from the rear of the missile.

II.3 Missile Aerodynamic Derivatives :

The resulting aerodynamic moments and forces are complex function of the control surface deflections, the incidence, Mack number, altitude, and air pressure [2,3,4,5,8]. The aerodynamic forces and moments are usually represented using dimensionless coefficients. The coefficient system provides a convenient method for deriving the over all aerodynmamic characteristics from the characteristics of the individual components of the missile (fuselage, wings tails, control surfaces, canards etc). The aerodynamic coeffients are independent of the size of the missile for a given configuration and hence are the standard means for converting data obtained from wind tunnels models to full scale characteristics.

The aerodynamic coefficients are defined by the following general relations:

$$\text{Force} \qquad F = C_F \frac{\gamma}{2} P M^2 S = C_F q S \qquad (2.3.1)$$

$$\text{Moment} \qquad m = C_m \frac{\gamma}{2} P M^2 S d d = C_m q S \mathrm{d} \qquad (2.3.2)$$

Where M: is the Mack number, P is the ambient pressure, S is a reference area
(usually the missile cross section, $\gamma = 1.4$ is the ratio of specific heats , and d is a reference length (usually the missile diameter). The ambient pressure is function of altitude and temperature [3]. A table of standard pressure vs altitude can be found in [8].

The normal (pitch) force , side force, axial force, pitching moment, yawing moments, and rolling moment dimensionless coefficients are then defined as follows

Normal (pitch plane)

$$C_Z = \frac{Z}{qS}$$

Side Force

$$C_Y = \frac{Y}{qS}$$

Axial Force

$$C_X = \frac{X}{qS}$$

Roll moment

$$C_L = \frac{L}{qSd}$$

Pitch moment

$$C_M = \frac{M}{qSd}$$

Yaw Moment

$$C_N = \frac{N}{qSd}$$

It is often convenient to exporess the pitching and Yawing moments in terms of the normal and side forces as

$$M = \frac{\gamma}{2} P M^2 S d C_Z$$

$$N = \frac{\gamma}{2} P M^2 S d C_Y$$

The aerodynamic coefficients are functions of alpha, beta , their derivatives, the body angular velocities, and the control surface deflections and their derivatives. The aerodynamic coeffients are usually approaximated by the multivariable Taylor polynomial series expansion assuming small incidence angles and small control surface angles.

The general expression for any C_λ , where lambda stands for X, Y, X, L, M, and N is

$$C_\lambda = C_{\lambda 0} + C_{\lambda\alpha}\alpha + C_{\lambda\alpha^2}\alpha^2 + C_{\lambda\beta}\beta + C_{\lambda\beta^2}\beta^2 + \frac{d}{2U}C_{\lambda\dot\alpha}\dot\alpha + \frac{d}{2U}C_{\lambda\dot\beta}\dot\beta +$$

$$\frac{d}{2U}C_{\lambda\omega_1}\omega_1 + \frac{d}{2U}C_{\lambda\omega_2}\omega_2 + \frac{d}{2U}C_{\lambda\omega_3}\omega_3 + C_{\lambda\xi_1}\xi_1 + C_{\lambda\xi_2}\xi_2 + C_{\lambda\xi_3}\xi_3 +$$

$$\frac{d}{2U}C_{\lambda\dot\xi_1}\dot\xi_1 + \frac{d}{2U}C_{\lambda\dot\xi_2}\dot\xi_2 + \frac{d}{2U}C_{\lambda\dot\xi_3}\dot\xi_3$$

The coefficients of the above expansions are called the aerodynamic derivatives and can be found from wind tunnel and flight tests [2,22], emperical formulas [24], or computer simulation and analysis techniques. Missile modeling and simulation will be treated in a subsequent report.

II.4 Missile Attitude System:

The relation between the rate of change of the attitude angles $(\theta_1,\theta_2,\theta_3)$ and the angular velocities $(\omega_1,\omega_2,\omega_3)$ can be stated by the following equation [7,13],

$$\underline{\omega} = (\omega_1, \omega_2, \omega_3)^T = R(\theta)\underline{\dot\theta} \qquad (2.4.1)$$

and

$$R(\theta) = \begin{bmatrix} \cos\theta_2\cos\theta_3 & \sin\theta_3 & 0 \\ -\cos\theta_2\sin\theta_3 & \cos\theta_3 & 0 \\ \sin\theta_2 & 0 & 1 \end{bmatrix}, \qquad (2.4.2)$$

Equation (2.4.1) is a key relation for determining the attitude of the missile since it relates the rate of change of the orientation of the missile $\underline{\theta}$ to the gyroscopes output

$$\underline{\omega} = (\omega_1, \omega_2, \omega_3)^T$$

The problem of computing Missile attitude comprises two closely related parts. These are the establishment of the orientation of the body coordinate system with respect to a reference coordinate system, and the transformation of vectors from one coordinate system to another. Different Techniques are available for solving the problem, for example: Euler angles, vector representation, Euler parameters (quaternions), Caley-Klein parameters, direction cosines, and Tensor representation. In this paper Euler parameters which are known as the Quaternions are adopted for the determination of the missile attitude [13,14]. Quaternions are usefull to simplify the notation required for equations involving general body relations. In this section, We reveiw briefly the quaternion parameterization method for attitude determination, and apply Pade' approximation method [12] to obtain a recursive set of equations for updating the quaternions and the missile attitude using the gyros measurements.

The Euler parameters q_i, are the components of a quaternion q defined by:

$$q = q_1 i + q_2 j + q_3 k + q_4 \tag{2.4.3}$$

Quaternions are an extension of complex numbers. The unit vectors i, j, k satisfy the relations

$$\begin{gathered} i^2 = j^2 = k^2 = -1 \\ ij = -ji = k \end{gathered} \tag{2.4.4}$$

$$jk = -kj = i$$

$$ki = -ik = j$$

The orientation of the missile with respect to its initial orientation may be found by integrating a quaternion differential equation [12].

$$\dot{q} = \frac{1}{2} q\,\omega = \frac{1}{2}(q_4 + iq_1 + jq_2 + kq_3)(i\omega_1 + j\omega_2 + k\omega_3) \qquad (2.4.5)$$

In terms of vector notation, equation (2.4.5) can be rewritten as

$$\underline{\dot{q}} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ -\omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \qquad (2.4.6)$$

A space vector may be expressed as a quaternion with $q_4 = 0$. If $\underline{q}$ represents the orientation of the fixed body axis with respect to reference axis, then the space vector is transformed from body x_B to reference axis x_R via the following transformation:

$$x_R = q\,x_B\,q^{-1} \qquad (2.4.7)$$

Now, and if the norm of q , q_o, is given by

$$q_o = \left(\sum_{i=1}^{4} q_i^2\right)^{1/2} = 1 \qquad (2.4.8)$$

then q is normalized and q^{-1} is given by its conjugate as

$$q^* = (q_4 + iq_1 + jq_2 + kq_3)^* = (q_4 - iq_1 - jq_2 - kq_3) \qquad (2.4.9)$$

which implies that x_R is given by

$$x_R = q x_B q^* \qquad (2.4.10)$$

Equation (2.4.10) is equivalent to the ordinary transformation by the direction cosine matrix C; Namely:

$$\underline{x}_R = C \underline{x}_B \qquad (2.4.11)$$

Using the above relations, namely (2.4.10) and (2.4.11), the cosine matrix may then be represented in terms of the Euler parameters as

$$C = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 - q_3 q_4) & 2(q_3 q_1 + q_2 q_4) \\ 2(q_1 q_2 + q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 - q_1 q_4) \\ 2(q_3 q_1 - q_2 q_4) & 2(q_2 q_3 + q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \qquad (2.4.12)$$

The cosine matrix is an orthogonal matrix that is traditionally used to compute the orientation of the missile by integrating the matrix differential equation [7] :

$$\dot{C} = C \ \tilde{\omega} \qquad (2.4.13)$$

It is possible to compute the pitch, yaw, and roll angles $(\theta_1, \theta_2, \theta_3)$ in terms of the quaternion parameters from the elements of the cosine matrix C as follows [13,14]: Define the parameter C_o as below

$$\begin{aligned} C_o = C_{11}^2 + C_{12}^2 &= (q_1^2 + q_2^2 + q_3^2 + q_4^2) - 8 q_1 q_2 q_3 q_4 \\ &= 1 - 8 q_1 q_2 q_3 q_4 \end{aligned} \qquad (2.4.14)$$

then

$$\theta_1 = \tan^{-1}\left(\frac{-C_{23}}{C_{33}}\right) \qquad (2.4.15)$$

$$\theta_2 = \tan^{-1}\left(\frac{C_{13}}{\sqrt{C_o}}\right) \qquad (2.4.16)$$

$$\theta_3 = \tan^{-1}\left(\frac{-C_{12}}{C_{11}}\right) \qquad (2.4.17)$$

where

$$-180^\circ \leq \theta_1 \leq 180^\circ$$

$$-90^\circ \leq \theta_2 \leq 90^\circ$$

$$-180^\circ \leq \theta_3 \leq 180^\circ$$

In high performance missiles and aircrafts, large angles maneuvers are obtained by coordination of the three angles. For example to obtain high yaw turn, the missile is allowed first to bank (roll) a certain angle, then the elevator and rudder deflections are coordinated to achieve the desired turn. Similarly, to perform vertical flight or vertical turns while avoiding the singularity at pitch angle = 90 degrees, the missile banks first by certain angle , then the vertical flight is achieved by coordinating the orientation between Yaw rotation and Pitch rotation (less than 90 degrees). One way to achieve such coordination is through the use of a coordination transform matrix CTM $\hat{T}$ such that

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix}_{coordinated} = \hat{T} \begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix}_{desired} \qquad (2.4.18)$$

It can easily be shown that the parameters q_1 to q_4 and C_{ij} are self normalized, i.e. having an absolute value bounded by one. $\underline{\theta}$ may then be evaluated using a truncated Taylor series expansion of arctan as follows:

$$\tan^{-1}x = \begin{cases} \dfrac{\pi}{2}-\dfrac{1}{x}+\dfrac{1}{3x^3}-\dfrac{1}{5x^5}+\dfrac{1}{7x^7}+\dots & (x>1) \\ x-\dfrac{x^3}{3}+\dfrac{x^5}{5}-\dfrac{x^7}{7}+\dots\dots & (-1\le x\le 1) \\ -\dfrac{\pi}{2}-\dfrac{1}{x}+\dfrac{1}{3x^3}-\dfrac{1}{5x^5}+\dfrac{1}{7x^7}+\dots & (-1>x) \end{cases} \qquad (2.4.19)$$

Moreover, for fixed-point implementation, it is always possible to use the middle expansion by proper normalization of the argument. This can be shown as follows :

To evaluate

$$\phi = arctan(Y,X) = arctan\left(\frac{Y}{X}\right)$$

1- find

$$\phi_0 = arctan(|Y|,|X|) \qquad \text{if}\,|Y|<|X|$$

$$\phi_0 = \frac{\pi}{4}+arctan\left(\left|\frac{Y}{X}\right|-1,\left|\frac{Y}{X}\right|+1\right) \qquad \text{if}\,|Y|\ge|X|$$

In the above step the middle expansion and fixed-point aritmetic can be used directly.

2- correct for the proper angle quarter

$$\text{if}\,X\ge 0, Y\ge 0 \qquad \phi=\phi_0$$

$$\text{if}\,X\ge 0, Y<0 \qquad \phi=-\phi_0$$

$$\text{if}\,X<0, Y\ge 0 \qquad \phi=-\phi_0+\pi$$

$$\text{if } X<0, Y<0 \qquad \phi=\phi_0-\pi$$

To solve the quaternion differential equation (2.4.5) numerically, G. Hyslop [12] proposed a numerical method that preserve the normality of the quaternions and solves the differential equation in a recursive fashion. Let T be the sampling period of the angular velocities measurement, and assume that the angular velocities are constant between samples, then the incremental rotations $\Delta\phi_i$ are given by

$$\Delta\phi_i = \int_{t-T}^{t} \omega_i dt \approx \omega_i T; \qquad i=1,2,3,\ldots$$

and define $\Delta\phi_0$ to be

$$\Delta\phi_0 = (\Delta\phi_1^2+\Delta\phi_2^2+\Delta\phi_3^2)^{1/2} \qquad (2.4.20)$$

Then an approximate solution to (2.4.5) may be given by

$$\underline{q}(t)=\underline{q}(t-T)e^{\frac{\Delta\phi}{2}} \qquad (2.4.21)$$

Using first order Pade' approximation [12], the vector form solution of q(t), which is also normal, is given by

$$\underline{q}(t)=\left\{\frac{16-\Delta\phi_0^2}{16+\Delta\phi_0^2}\, I+\frac{8}{16+\Delta\phi_0^2}\, \tilde{\Delta}\Phi\right\}\underline{q}(t-T) \qquad (2.4.22)$$

with

$$\tilde{\Delta}\Phi=\begin{bmatrix} 0 & \Delta\phi_3 & -\Delta\phi_2 & \Delta\phi_1 \\ -\Delta\phi_3 & 0 & \Delta\phi_1 & \Delta\phi_2 \\ \Delta\phi_2 & -\Delta\phi_1 & 0 & \Delta\phi_3 \\ -\Delta\phi_1 & -\Delta\phi_2 & -\Delta\phi_3 & 0 \end{bmatrix} \qquad (2.4.23)$$

and,

$$\Delta\phi_o = T(\omega_1^2+\omega_2^2+\omega_3^2)^{1/2}$$

let us define d_1, d_2, d_3 as follows:

$$d_1 = \frac{\Delta\phi_0^{\,2}}{16} \;, d_2 = \frac{1-d_1}{1+d_1} \;,\; d_3 = \frac{T}{2(1+d_1)} \qquad (2.4.24)$$

Since the sampling period satisfies $\pi > T\omega_i$, then it can be shown that d_1, d_2 and d_3 are also normalized for fixed point computation.

A further manipulation of equation (2.4.22) yields the follwing recursive form

$$\underline{q}(t) = d_2\underline{q}(t-T) + d_3 \;\tilde{\Omega}(t)\; \underline{q}(T-t) \qquad (2.4.25)$$

where Ω is defined as $\Delta\Phi$ in (2.4.23). If t = kT, the last equation breaks down to the following normalized set of recursive equations;

$$q_1(kT) = d_2q_1[(k-1)T] + d_3[q_4[(k-1)T]\Delta\phi_1(kT) - q_3[(k-1)T]\Delta\phi_2(kT) + q_2[(k-1)T]\Delta\phi_3(kT)] \qquad (2.4.26)$$

$$q_2(kT) = d_2q_2[(k-1)T] + d_3[q_3[(k-1)T]\Delta\phi_1(kT) + q_4[(k-1)T]\Delta\phi_2(kT) - q_1[(k-1)T]\Delta\phi_3(kT)] \qquad (2.4.27)$$

$$q_3(kT) = d_2q_3[(k-1)T] + d_3[q_2[(k-1)T]\Delta\phi_1(kT) + q_1[(k-1)T]\Delta\phi_2(kT) - q_4[(k-1)T]\Delta\phi_3(kT)] \qquad (2.4.28)$$

$$q_4(kT) = d_2q_4[(k-1)T] + d_3[q_1[(k-1)T]\Delta\phi_1(kT) + q_2[(k-1)T]\Delta\phi_2(kT) - q_3[(k-1)T]\Delta\phi_3(kT)] \qquad (2.4.29)$$

Finally, the attitude of the missile $\underline{\theta}(kT)$ can be obtained from (2.4.15 - 2.4.17) using the updated Euler parameters.

III. THE ADAPTIVE CONTROL LAW AND IMPLEMENTATION :

The classical approach for missile attitude control [3,4,5,6,7,24] depends on reducing the coupling between roll, pitch and yaw, and hence each one can be controlled independently. The first step in this approach is to design the missile such that Ixz = 0 to eleminate the nonlinear coupling associted with Ixz in the equations (2.1.5 - 2.1.10). The second step is to stablize the roll, i.e. forcing $w_1 = 0$, hence the dynamics of the pitch and yaw will be almost decoupled. Finally the transfer functions are derived using the approximations

$$C_N = C_{N\alpha}\alpha + C_{N\alpha_p}\alpha_p$$

$$C_M = C_{M\beta}\beta + C_{M\alpha_y}\alpha_y$$

The classical feedback control theory is used to shape loop transmission, where the missile aerodynamic derivatives and performance specifications are transformed into scheduled control templates according the flight conditions (Mack number, and altitude).

However, the demand for high maneuverability, e.g. high performance bank to turn, induces inertia and kinematic cross coupling effects among the pitch, roll, and yaw channels [2,22]. The coupling becomes more severe with increasing missile roll rate, and even more severe in the case of asymmetrical airframes. Several multivariable control techniques were reported in the published literatures [9,10,11,18,19,21,23].

[Lobbia and Tso, 18] applied the Linear Quadratic Regulator LQR technique for control a cruise missile taking the coupling between the Yaw and Roll channels, neglecting the pitch channel coupling, and assuming steady flight conditions. In order to achieve high robustness in the presence of uncertainity, the design was based on the worst case condition which lead to poor response time.

[Lin & Lee , 19] Applied the generalized Singular Linear Quadratic GSLQ control for coordinating the pitch , yaw and roll commands. The method allows to synthesize state FB gain matrix based on minimizing a quadratic cost function that does not contain explicit penality on control. The exact control law requires solution of Riccati matrix differential equation and a vector differential equation. To make the solution tractable, a suboptimal solution was investigated based on the solution of stationary Riccati equation. The control law is simplified to the form

$$u = K_1 \Delta x + u_f$$

where

$$u_f = K_2(B_f - \dot{x}_r + A x_r)$$

$$\Delta x = x - x_r; \qquad x, x_r \in \mathbf{R}^{8 \times 1}$$

are the state vector and the desired state vector respectively. K_1, K_2 are gain matrices obtained by solving the Matrix Riccati Equation. B_f is a vector that accounts for estimated nonlinearity, uncertainity and disturbances. The gain matrices and the system matrices are computed and stored in templates corresponding to various flying conditions as Mack number and altitude. A gain scheduling strategy is required for template selections. The basic computations of the simplified control law break down to about 148 (MDO) and 132 (ASO), compared to about 114 (MDO) and 107 (ASO) for the computation of our control law. Although Iin & Lee approach demonstrated a dramatic improvement in performance over techniques based on uncoupled control channels. However, the method is still based on linearization of the aerodynamic forces and moment and requires tabulation of the missile dynamic models parameters for the various anticipated flight conditions. The control law is computed for each one of these conditions and stored in the on board computer along with the model templates.

[Dwyer, 11] developed an exact nonliear control law using a novel approach where a nonlinear transformation of variables was derived to reduce the problem of rigid body rotational maneovers to an equivalent linear control problem. The method has great potential in the control of spacecrafts and robotics where the body parameters are known and the nonliear aerodynamic transfer functions are not considered. Further investigation is still needed to apply this technique to missile autopilot.

[Ridgely et al, 21] consdidered the application of multivariable Linear Quadratic Gaussian LQG control with Loop Transfer Recovery. In this method , the formulation proceeds as regular LQG , however with free parameters in the cost function. These free parameters are then used to retune the frequency response characteristics based on the singular values of the filter loop transfer matrix. The method requires a great deal of iterations and simulations since not all the classical frequency domain design is extendable to MIMO.

Application of the conventional adaptive control methods has not been reported in the published literatures for the attitude control of missiles. The reason is probably due to the fact that the conventional adaptive control theory is based mainly on linear system models with fixed but unknown parameters or with slowly varying parameters. Convergence and stability of such algorithms in the presence of strong nonlinear dynamics has not been investigated yet. However, more recently a multivariable adaptive control scheme was investigated by JPL for the control of a large space structure [16].

In the next section we briefly present the adaptive control scheme proposed in [9], and introduce a few manipulations in order to break down the computations into efficient procedures for microcontroller implementation.

III.1 The Nonlinear Adaptive Control Law:

The control system comprises a model reference which accepts attitude set points, $r_1(t)$, $r_2(t)$, $r_3(t)$, from a target tracking system. The model reference produces reference angles $\underline{\hat{\theta}}=(\hat{\theta}_1,\hat{\theta}_2,\hat{\theta}_3)^T$ representing the desired roll, pitch and yaw respectively. The model reference produces as well $\dot{\underline{\hat{\theta}}}$, a desired rate of change of the missile attitude. The reference $(\underline{\hat{\theta}},\dot{\underline{\hat{\theta}}})$ are then compared with $(\underline{\theta},\dot{\underline{\theta}})$ the estimated value of the actual attitude of the missile. The error $\underline{Z}$ is then used to derive a nonlinear control law in which parameters are continuously estimated and updated using a dynamic system.

The nonlinear control law can be described by the following set of equations [9]: recalling the tracking error

$$\underline{Z}(t)=(\underline{e}(t),\dot{\underline{e}}(t))^T \qquad (3.1.1)$$

and assuming that the dynamics of the model reference is given by

$$\frac{d^2}{dt^2}\theta_i(t)=-k_i\theta_i(t)-\varepsilon_i\frac{d}{dt}\theta_i(t)+k_ir_i(t) \quad , \quad i=1,2,3 \qquad (3.1.2)$$

where $\qquad k_i>0 \quad , \quad \varepsilon_i>0$

let us define the following matrices :

$$K = diag(k_1, k_2, k_3) \quad , \quad \hat{C} = diag(\hat{c}_1, \hat{c}_2, \hat{c}_3) \tag{3.1.3}$$

$$A_m = \begin{bmatrix} 0 & I \\ -K & -\hat{C} \end{bmatrix} \tag{3.1.4}$$

The resulting A_m defined above is a stability matrix. Now let us define a matrix Q as

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \tag{3.1.5}$$

with Q_1, and Q_2 positive diagonals. Then, the solution, P, of the following Lyaponov equation

$$\mathbf{P}A_m + A_m^T\mathbf{P} + Q = 0 \tag{3.1.6}$$

is positive definite, and its partition is given by:

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

then

$$P_{21} = \frac{1}{2}K^{-1}Q_1 \tag{3.1.7}$$

and

$$P_{22} = \frac{1}{2}\hat{C}^{-1}(Q_2 + K^{-1}Q_1) \tag{3.1.8}$$

The elements of P_{21} and P_{22} are precalculated and stored permanently in the computer since they will be utilized in computing the control law.

The orientation control vector $\underline{u}(t)$ is obtained from the following relations [9];

$$\underline{v}(t) = R^T(\underline{\theta}) \ \ \underline{u}(t) \tag{3.1.9}$$

where

$$\underline{v}(t) = \prod(\underline{e},\underline{\dot{e}},\beta) S(\underline{Z},\beta) \qquad (3.1.10)$$

$$\prod(\underline{e},\underline{\dot{e}},\beta) = \beta_1 |\underline{e}|_2 + \beta_2 |\underline{\dot{e}}|_2 + \beta_3 |\underline{\dot{e}}|_2^2 + \beta_4 \qquad (3.1.11)$$

$$S(\underline{Z},\beta,t) = Sat\left[\frac{2\prod(\underline{e},\underline{\dot{e}},\beta)(P_{12}\underline{e} + P_{22}\underline{\dot{e}})}{\beta_5}\right] \qquad (3.1.12)$$

$$Sat(\underline{x}) = \begin{bmatrix} x & if & |x| \le 1 \\ \dfrac{x}{|x|} & if & |x| > 1 \end{bmatrix} \qquad (3.1.13)$$

The parameters β_i 's are some dynamic parametrs to be estimated through a dynamic system as shown below.

III.2 Parameters Estimation :

Following [9], the parameters β_i 's are estimated using a dynamic estimator of the form

$$\frac{d}{dt}\beta_1 = n_{11} |\underline{e}|_2 |\alpha(\underline{Z})|_2 \quad , \quad \frac{d}{dt}\beta_2 = n_{22} |\underline{\dot{e}}|_2 |\alpha(\underline{Z})|_2$$

$$\frac{d}{dt}\beta_3 = n_{33} |\underline{\dot{e}}|_2^2 |\alpha(\underline{Z})|_2 \quad , \quad \frac{d}{dt}\beta_4 = n_{44} |\alpha(\underline{Z})|_2$$

$$\frac{d}{dt}\beta_5 = -n_{55} |\underline{e}|_2 \qquad (3.2.1)$$

where

$$\alpha(\underline{Z}) = 2(P_{21}\underline{e} + P_{22}\underline{\dot{e}})$$

$$n_{ii} > 0 \quad , \beta_i(0) \in (0,\infty) \quad , \forall i = 1,2,\ldots,5.$$

III.3 The Control Surfaces and The Control Signals :

The orientation control command $\underline{u}^T = (\alpha_r, \alpha_p, \alpha_y)$ is related to the vector $\underline{v}(t)$ by equation (3.1.9), and related to the control surfaces commands by the relation

$$\underline{\xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \underline{u} \qquad (3.3.1)$$

The orientation matrix $R^{-1}(\theta)$ is obtained in terms of the quaternion q_1, q_2, q_3, q_4 and combined with relation (3.3.1) to get

$$v_2 = T_3 v_2 \qquad (3.3.2)$$

$$v_3 = C_{13} v_3 \qquad (3.3.3)$$

$$\xi_1 = T_1 v_1 + T_2 v_2 - T_1 v_3 \qquad (3.3.4)$$

$$\xi_2 = v_3 \qquad (3.3.5)$$

$$\xi_3 = T_2 v_1 - T_1 v_2 - T_2 v_3 \qquad (3.3.6)$$

where

$$T_1 = \frac{C_{11} + C_{12}}{C_o} \qquad (3.3.7)$$

$$T_2 = \frac{C_{11} - C_{12}}{C_o} \qquad (3.3.8)$$

$$T_3 = \sqrt{C_o} \qquad (3.3.9)$$

III.4 Implementation of The Nonlinear Adaptive Control:

The nonlinear adaptive control is actuated through the measurement of the error in the attitude of the missile. In order to generate the input command, the differential equation of the model reference and the various functions involved in the control have to be calculated using the microprocessor. The calculation procedure requires the equations to be put in the form of discrete recursive equations that can be computed every sampling interval.

The equations of the model reference are given by (3.1.2). Suppose that the system signals are sampled every T seconds for some small value of T. If the derivatives are approximated by a first order backward difference, Equation (3.1.2) can be written in the form

$$\dot{\hat{\theta}}_i(kT) = (1 - Tc_i)\dot{\hat{\theta}}_i[(k-1)T] - Tk_i\hat{\theta}_i[(k-1)T] + Tk_i r_i[(k-1)T] \quad (3.4.2)$$

$$\hat{\theta}_i[kT] = T\dot{\hat{\theta}}_i[kT] + \hat{\theta}_i[(k-1)T] \qquad i = 1,2,3 \qquad (3.4.3)$$

Using (3.4.3) and equations (2.3.12 - 2.3.14), the errors and its derivatives can be computed as

$$e_i[kT] = \theta_i[kT] - \hat{\theta}_i[kT] \qquad i = 1,2,3 \qquad (3.4.4)$$

$$\dot{e}_i[kT] = \dot{\theta}_i[kT] - \dot{\hat{\theta}}_i[kT] \qquad i = 1,2,3 \qquad (3.4.5)$$

Where $\dot{\theta}_i$'s are determine using (2.1.5) and the gyroscpes measurements ω_i 's as follws,

$$\dot{\theta}_1[kT] = (C_{11}[kT]\omega_1[kT] + C_{12}[kT]\omega_2[kT])\frac{1}{C_0[kT]} \qquad (3.4.6)$$

$$\dot{\theta}_2[kT] = (C_{12}[kT]\omega_1[kT] + C_{11}[kT]\omega_2[kT])\frac{1}{\sqrt{C_0[kT]}} \qquad (3.4.7)$$

$$\dot{\theta}_3[kT] = \omega_3[kT] - C_{13}[kT]\theta_1[kT] \qquad (3.4.8)$$

Using (3.4.4) and (3.4.5) the norm of the error and its rate can be computed as

$$|e[kT]| = (e_1^2[kT] + e_2^2[kT] + e_3^2[kT])^{\frac{1}{2}} \qquad (3.4.9)$$

$$|\dot{e}[kT]| = (\dot{e}_1^2[kT] + \dot{e}_2^2[kT] + \dot{e}_3^2[kT])^{\frac{1}{2}} \qquad (3.4.9)$$

The function $\underline{\alpha}(Z)$ of equation (3.2.1) can be evaluated as

$$\alpha_1[kT] = 2(P_{21}(1,1)e_1[kT] + P_{22}(1,1)\dot{e}_1[kT]) \qquad (3.4.12)$$

$$\alpha_2[kT] = 2(P_{21}(2,2)e_2[kT] + P_{22}(2,2)\dot{e}_2[kT]) \qquad (3.4.13)$$

$$\alpha_3[kT] = 2(P_{21}(3,3)e_3[kT] + P_{22}(3,3)\dot{e}_3[kT]) \qquad (3.4.14)$$

and the norm of $\underline{\alpha}(Z)$ can be computed as

$$|\alpha[kT]| = (\alpha_1^2[kT] + \alpha_2^2[kT] + \alpha_3^2[kT])^{\frac{1}{2}} \qquad (3.4.15)$$

The computation of the parameters β_i of (3.2.1 - 3.2.2) can be performed in the same fashion as follows:

$$\hat{\beta}_1[kT] = T * n_{11} * |e[kT]|_2 * \|\alpha[kT]\|_2 + \hat{\beta}_1[(k-1)T] \qquad (3.4.16)$$

$$\hat{\beta}_2[kT] = T * n_{22} * |\dot{e}[kT]|_2 * \|\alpha[kT]\|_2 + \hat{\beta}_2[(k-1)T] \qquad (3.4.17)$$

$$\hat{\beta}_3[kT] = T * n_{33} * |\dot{e}[kT]|_2^2 * |\alpha[kT]|_2 + \hat{\beta}_3[(k-1)T] \qquad (3.4.18)$$

$$\hat{\beta}_4[kT] = T * n_{44} * \alpha[kT] + \hat{\beta}_4[(k-1)T] \qquad (3.4.19)$$

$$\hat{\beta}_5[kT] = T * n_{55} * |e[kT]| + \hat{\beta}_5[(k-1)T] \qquad (3.4.20)$$

Once these parameters are available the control law can readily be computed in the following manner

$$\prod[kT] = \beta_1[kT]|e[kT]| + \beta_2[kT]|\dot{e}| + \beta_3|\dot{e}[kT]|^2 + \beta_4[kT] \quad (3.4.21)$$

$$\prod[kT] = \frac{\prod[kT]}{\beta_5[kT]} \quad (3.4.21)-$$

$$\Pi_i[kT] = \prod[kT]\alpha_i[kT] \qquad i = 1,2,3 \quad (3.4.23)$$

$$|\Pi[kT]| = (\Pi_1^2[kT] + \Pi_2^2[kT] + \Pi_3^2[kT])^{\frac{1}{2}} \quad 3.4.24)$$

$$S_i[kT] = \begin{cases} \Pi_i[kT] & \Pi \leq 1 \\ \dfrac{\Pi_i[kT]}{|\Pi|} & |\Pi| > 1 \end{cases} \quad (3.4.25)$$

Then

$$v_i[kT] = -\prod[kT]S_i[kT] \qquad i = 1,2,3 \quad (3.4.26)$$

Now it is the matter of programming all these equations on the microprocessor program memory.

IV. SOFTWARE DESCRIPTION :

The following algorithm is a step by step procedure suitable for direct implementation on microprocessors. The steps are as follows :

Step Ø : Set the following model parameters and initial conditions,

$$q_1(0), q_2(0), q_3(0); \beta_1(0), \beta_2(0), \beta_3(0), \beta_4(0), \beta_5(0);$$

$$\theta_1(0), \theta_2(0), \theta_3(0); \dot{\theta}_1(0), \dot{\theta}_2(0), \dot{\theta}_3(0); \ddot{\theta}_1(0), \ddot{\theta}_2(0), \ddot{\theta}_3(0);$$

$$\dddot{\theta}_1(0), \dddot{\theta}_2(0), \dddot{\theta}_3(0); \omega_1(0), \omega_2(0), \omega_3(0); r_1(0), r_2(0), r_3(0).$$

All initial conditions are selected offline and stored in the microprocessor memory. The initial value of the controller parameters can be found by simulation using the initially known missile model.

Now for every sampling interval do the following :

Step 1 : Solve the quaternion dynamics and compute the following :

$q_1(kT), q_2(kT), q_3(kT)$, and $q_4(kT)$

from equations (2.3.22 - 2.3.25) respectively.
This step requires 22 addition and subtractions operations (ASO), and 30 multiplication and division operations (MDO).

Step 2 : Solve the dynamics equations of the reference model and compute the following ;

$\dot{\theta}_i(KT) \quad i = 1,2,3$

using equation (3.4.2).

$\ddot{\theta}_i(KT) \quad i = 1,2,3$

using equation (3.4.3).
This step requires 12 ASO and 12 MDO operations.

Step 3 : Solve for $(\underline{\dot{\theta}},\underline{\theta})$ as follows :

$\dot{\theta}_1(kT)$

using equation (3.4.6)

$\dot{\theta}_2(kT)$

using equation (3.4.7)

$\dot{\theta}_3(kT)$

using equation (3.4.8)

$\theta_1(kT)$

using equations (2.3.12) and (2.3.15)

$\theta_2(kT)$

using equation (2.3.13 and 2.3.15)

$\theta_3(kT)$

using equation (2.3.14 and 2.3.15)

This step requires 19 (ASO) and 12 (MDO) plus three arctan operations.

Step 4 : Compute the error in the missile attitude and the parameter $\underline{a}(\underline{Z})$;

$\dot{e}_i(KT) \quad i=1,2,3$

using equation (3.4.5).

$e_i(kT) \quad i=1,2,3$

using equation (3.4.4).

$|\dot{\underline{e}}[kT]|$

using equation (3.4.10).

$|\underline{e}[kT]|$

using equation (3.4.9).

$\alpha_1[kT], \alpha_2[kT], \alpha_3[kT]$, and $\alpha[kT]$

using equations(3.4.12 - 3.4.15) respectively.

This step requires 28 (ADS) and 27 (MDO) plus 3 square root operations.

<u>Step 5</u> : Solve the dynamics of the controller parameters in the following way:

$\beta_1[kT]$

using equation (3.4.16)

$\beta_2[kT]$

using equation (3.4.17)

$\beta_3[kT]$

using equation (3.4.18)

$\beta_4[kT]$

using equation (3.4.19)

$\beta_5[kT]$

using equation (3.4.20)

This step requires 5 (ASO) and 13 (MDO).

<u>Step</u> <u>6</u> : Control law computation. Find first the following parameters :

$\prod[kT]$

using equation (2.4.22)

$\Pi_i[kT] \quad i = 1,2,3$

using equation (3.4.23)

$\Pi[kT]$

using equation (2.4.24)

<u>Step</u> <u>7</u> : IF

$\Pi[kT] > 1$

GO To Step 9

Step 8 : Compute S(kT) as

$$S_i(kT) = \Pi_i \qquad i = 1,2,3$$

Go To Step 10.

Step 9 : Compute S(kT) as

$$S_i(kT) = \Pi_i \frac{[kT]}{|\Pi|} \qquad i = 1,2,3$$

Steps 6 - 9 require 6 (ASO) and 16 (MDO) plus one square root operation and one logical decision.

Step 10 : Compute the control variables as follows:

$v_i[kT] \qquad i = 1,2,3$

using equation (3.4.26)

$T_i[kT] \qquad i = 1,2,3$

using equations (3.3.7-3.3.9)

Finally compute the control surfaces deflections as:

$\xi_i[kT] \qquad i = 1,2,3$

using equation (3.3.4-3.3.6)

This step requires 6 (ASO) and 11 (MDO) plus one square root.

<u>Step</u> <u>11</u> : Wait for the next sample and go to step 1.

A major difficulty in programming embedded controllers as the 8797 is that for efficient utilization of the scarce on chip data memory, and for high speed execution it is necessary to program the algorithm using assembly language. However, a conceptual C program is given in Appendix A which demonstrates how scaling of variables is used [15] to allow a great deal of computations to be performed using 16-bit fixed point arithmetic.

V. MICROPROCESSOR AND HARDWARE IMPLEMENTATION :

V.1 The Microprocessor :

The Intel 8797, a member of MCS96 family, is a 16-bit micro-controller with dedicated real-time I/O subsystem and a set of 16 bit arithmetic instructions. The MCS96 family has been designed for high speed, high performance real-time control applications.

The MS96 series has several unique features that lend this series the best for such applications; e.g. the availability of the A/D conversion channels on chip, the programmable High Speed Output HSO unit for handling multi teal-time events, and the integration of program and data memory on chip. Unfortunately, such series of microcontroller lacks internal support of floating point operations, and there is no compatible Math Coprocessor chips for this series. Hence, mathematical computations have to performed using fixed point arithmetic with extra care to scale variables, or to execute floating point operations unefficiently by software. Recently, 1988-1989, a fourth generation of 32 bit microcontrollers and Digital Signal Processors DSP became commercially available; e.g. Texas Instrument TMS 320C30, Intel 80960, AT & T DSP-32.

All were provided with internal Floating Point Math coprocessors for unpreceded number crunching capability. However, the computational capability was enhanced at the expense of the real-time hardware features. For example the HSO was removed from the new Intel 80960. On the other hand, the 80960 is at least 10 times faster than the MS86 series in floating point computations, it takes about one microsec to perform floating point multiplications and about 14.5 microsecond to obtain arctan. Similarly, TMS 320C30 is capable of executing 33 Million Floating Point Instructions Per second, that is almost 100 times faster than the intel MS96 microcontrollers. If size/power consumption is not critical, a popular low cost alternative for enhanced floating point number crunching is the use of 80186/8087 microprocessors. However, a board having the same functions as those on the MS96 chip would require 5 -10 times power/size the MS96 based board.

There are two main sections of the 8797, the CPU section and the I/O section. Each of these sections consists of a number of functional blocks as shown in Fig. 4. The 8797 has, in addition, an 8 K bytes EPROM on the chip.

The CPU uses a 16-bit ALU which operates on a register file (256 bytes) instead of a single accumulator. The low 24 bytes of the register file are special function registers SFRs. The remaining 232 bytes are general purpose RAM. The serial port has several programmable modes and may be used in conjunction with an appropriate radio transceiver for remote commands. The A/D converter has 8 multiplexed analog channels and 10-bit resolution. D/A conversion in the form of PWM signals can be generated in a programmable High Speed Output unit HSO. The HSO contains two timers, 4 software timers, and others functions not used in this application.

The following is a summary of the main relevant features:

EPROM size	8K bytes
Data bus	16-bit
A/D resolution	10-bit
A/D conversion	22 μ sec.
On chip user mem.	232 bytes
Clock	12 MHz

Analog output	PWM
Analog inputs	8 channels
Timers	2 timers, 16 bits each.

The HSO unit, shown in Fig. 5, includes an eight register Content Addressable Memory CAM for storing real-time pending actions.

Each of the CAM registers is 23 bits; 16 bits of them specify the time at which the action to be carried out, and the remaining 7 bits specify the nature of action and whether timer 1 or timer 2 is used as a reference.

The embodiment hardware is basically the single chip micro-controller shown in Fig. 6, where the program resides entirely in the internal EPROM. The attitude set point commands are received from a guidance system via the chip I/O ports number 3 and 4.

Pins P1.0 and P1.1 of port1 are used for selection of attitude set points r1 to r3, while pins P1.2 and P1.3 are used for handshaking with the guidance unit as shown in Figure 6. The servomechanisms of the control surfaces are driven by PWM signals on 3 pins of the high speed output unit, HS0.0 to HS0.2. The PWM signals are generated by 3 software timers. The PWM rate is fixed at 400 Hz, and the direction of rotation is determined by separate output lines, P1.5, P1.6, and P1.7 respectively. The PWM conversion range is about 11 bits. A fourth software timer of the HSO unit is used to generate 50 Hz. time-base interrupts. The analog input from the gyros corresponding to the angular rates w1, w2, and w3 are directed to the analog channels ACH0 to ACH3, and sampled at 50 sps.

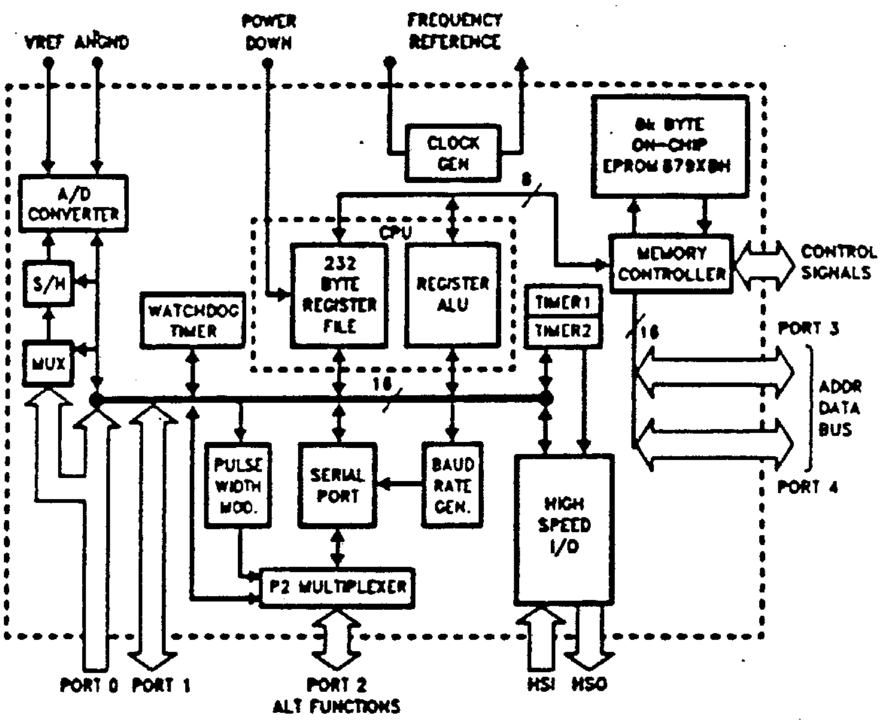


Figure 4. MCS®-96 Block Diagram

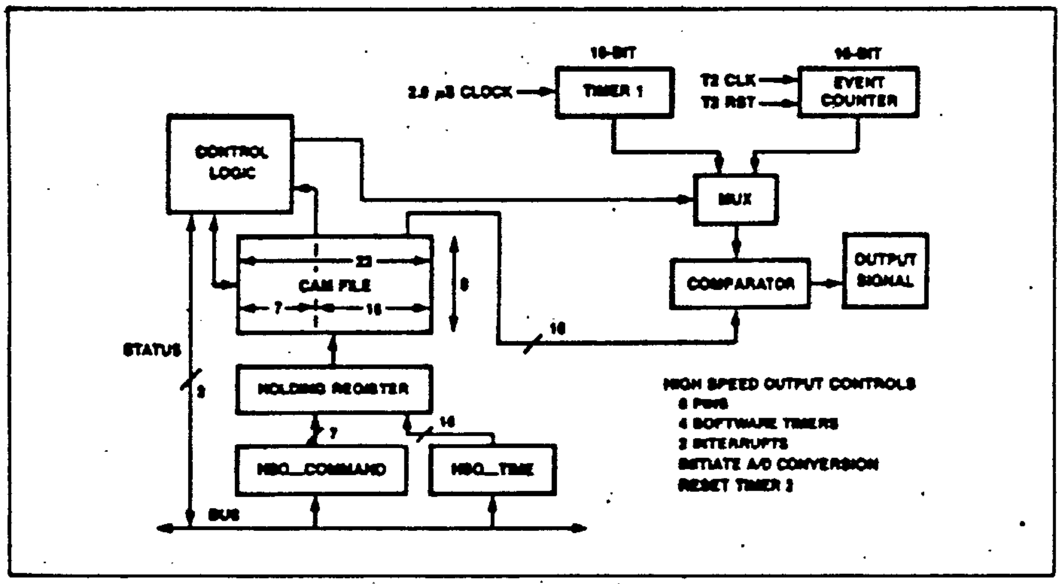


Fig. 5 HSO Block Diagram.

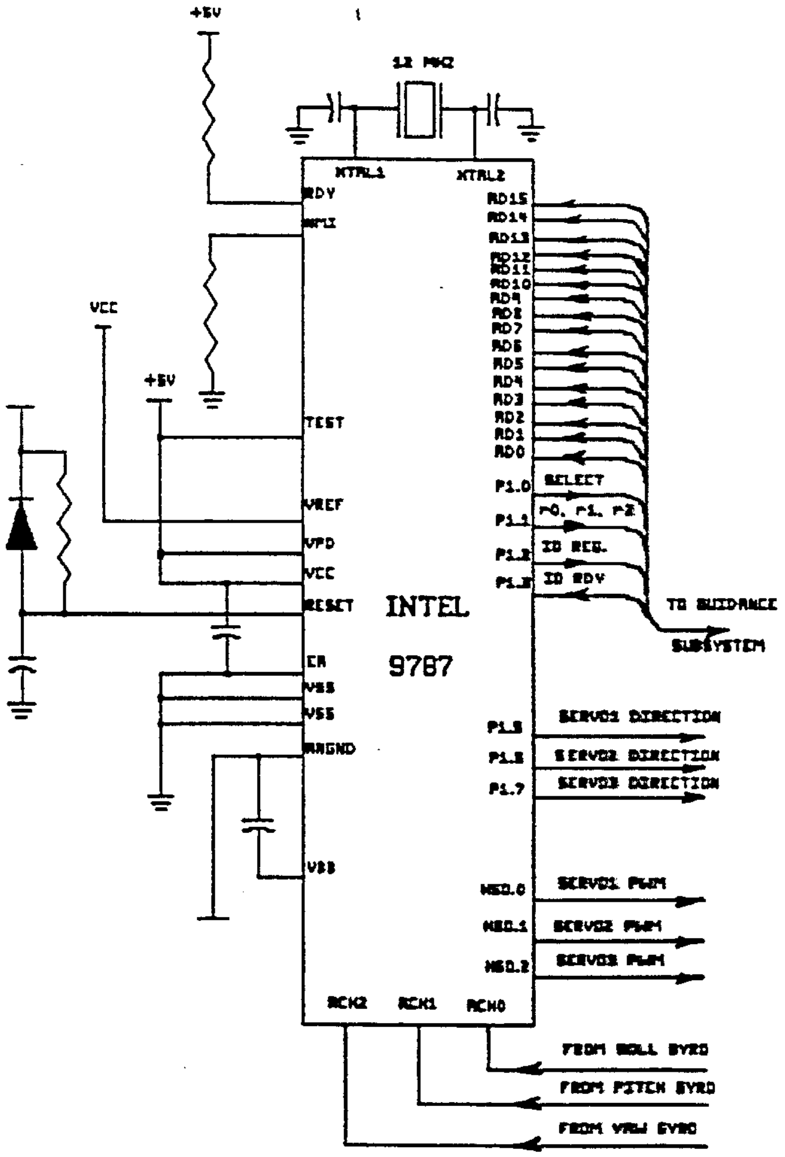


Fig. 6 Controller Hardware

VI.2 Arithmetic Operations Count :

A table of the arithmetic operations count is shown below:

	FXD/FLT	ADD/SUB	MUL/DIV	SQRT	ARCTAN	D/A PWM
Quaternion	Fixed	22	30	-----	-----	3
Model Refer.	Fixed	12	12	-----	-----	-----
attitude	Fixed	19	12	-----	3	-----
Con. Param.	Float	28	27	3	-----	-----
Param. Estim.	Float	5	13	-----	-----	-----
Control Law	Float	6	16	1	-----	-----
Delections	Float	6	11	1	-----	-----

The 8797 microcontroller performs 16x16 bit multiplication-/divisions in 6.25 micro sec. and 16 bit addition/subtractions in one micro sec. On the other hand, the floating point addition/subtraction and multiplication/division takes about 10 microsecond, and 22 microsecond respectively.
The floating point operations are performed by software. The floating point numbers consist of a 16-bit mantissa, a 6 bit exponent and two sign bits; a total of 24 bits.

The square root and arctan are performed using fixed point arithmetic as well. In this case their maximum execution time is estimated to be 68 micro sec. and 155 micro sec. respectively. Floating point square root is obtained by scaling the number by 2^{-2m} and using the fixed-point square root algorithm.

From the above analysis, the overall execution time per sample including data shuffling, A/D and HSO set up is estimated to be less than 4.5 micro sec. With a sample rate of 50 sps, the microcontroller will still have ample time to execute further functions as guidance and communications.

CONCLUSION

This report addresses some implementation issues of high performance missile attitude control using a 16-bit microcontroller. The implementation included a numerically efficient attitude reference system and a recently proposed nonlinear adaptive control system for rapid and precise large angle maneuver.

It is shown that the microcontroller is not fully utilized. Hence, other improved numerical techniques can be employed to improve the accuracy of the computations. For example, an improved integration method in the model reference and use of a second order Pade approximation for updating the quaternion may be considered. The control system may also be enhanced by including lateral acceleration measurements to achieve rapid correction of the missile incidence. Other on-board missile functions as guidance and communications can easily be accommodated in the microcontroller as well.

REFERENCES :

[1] INTEL, Embedded Controller Handbook, Vol. 2, 1988.

[2] M.J. Hemsch and J.N. Nielsen (edrs), Tactical Missile Aerodynamics, American Institute of Aeronautics and Astronautics, 1986.

[3] G. Merrill, Principles of Guided Missile Design, Van Nostrand Company, Inc, New York, 1956.

[4] P. Garnell and D. East, Guided Weapon Control System, Pergamon, Oxford, 1977.

[5] J.H. Blakelock, Automatic Control of Aircraft and Missiles, John Wiley &Sons, 1966.

[6] F. Kovacevic, "Microprocessor Control of Rocket Flight Stabilization," Microprocessors and Microsystems, Vol. II, No. 6, pp. 330-335, 1987.

[7] J. Wertz, Spacecraft Attitude Determination and Control, D. Reidel Publishing Company, 1978.

[8] A.L. Greensite, Analysis and Design of Space Vehicle Flight Control Systems, Spartan Books, New York, 1970.

[9] S.J. Singh, "Nonlinear Adaptive Attitude Control of Spacecraft," IEEE Transaction on Aerospace and Electronic Systems, Vol. AES-23, No. 3, pp. 371-379, 1987.

[10] C.K. Carrington and J.L. Junkins, "Nonlinear feedback control of spacecraft slew maneuvers", The Journal of the Astronautical Science, Vol. 32, No. 1, pp. 29-45, 1984.

[11] T.A.W. Dwyer, "Exact nonlinear control of large angle rotational maneuver," IEEE Transaction on Automatic Control, AC-29, No. 9, pp. 769-774, 1984.

[12] G. Hyslop, "A Norm Orthogonality Preserving Algorithm," IEEE Trans. on Aerospace and Electronic System, Vol. AES-23, No. 6, Nov. 1987, pp. 731-737.

[13] L. Meirovitch, Methods of Analytical Dynamics, McGraw-Hill, 1970.

[14] J. Wilcox, "A New Algorithm for Strapped-Down Inertial Navigation," IEEE Trans. on Aerospace and Electronic System, Vol. AES-3, No. 5, 1967, pp. 796-802.

[15] M.E. Ahmed and P.R. Belanger, "Scaling and Round Off in Fixed-Point Implementation of Control Algorithms," IEEE Trans. Industrial Electronics, Vol. IE-31, No. 3, August 1984.

[16] D.S.Bayard, C.C.Ih, and S.J.Wang,"Adaptive Control For Flexible Structures With Measurement Noise", Proceedings of the 1987 American Control Conference, pp. 368-379.

[17] M. Elshafei Ahmed and R. L. Steadman, Microprocessor-based Attitude Reference System, Research Report, 1984.

[18] R.N. Lobbia and G.Tso," An Application of LQR Theory In An Inegrated Guidance-Control Design For Advanced Cruise Missiles", Proceedings of the 1987 American Control Conference, pp. 770-778.

[19] C.F. Lin and S.P. Lee," Robust Missile Autopilot Design Using a Generalized Singular OPtimal Control Technique", J. Guidance, Vol 8, No. 4, pp. 498-507, July-August 1985.

[20] F.V. Nesline and P. Zarchan, " Missile Guidance Design Tradeoffs for High-Altitude Air Defense", J. Guidance, Vol. 6, No . 3, pp.207-212, May-June 1983.

[21] D. B. Ridgely, S.s. Banda, T.E. McQuade, and P.J. Lynch, " Linear Quadratic Gaussian with Loop-Transfer Recovery Methodolgy for an Unmanned Aircraft", J. Guidance, Vol. 10, No. 1, pp. 82-89, January-February 1987.

[22] V.E. Larimore, V.M. Lebow, and R.K. Mehra, " Identification of Parameters and Model Structure For Missile Aerodynamics", Proceedings of the 1985 American Control Conference, pp. 18-26.

[23] C.F. Lin, " Optimal Missile Midcourse And Terminal Guidance and Control Law Design", Proceedings of the 1985 American Control Conference, pp. 1-6.

[24] B. Etkin, <u>Dynamics of Flights: Stability and Control</u>, John Wiley & Son New York, 1982.

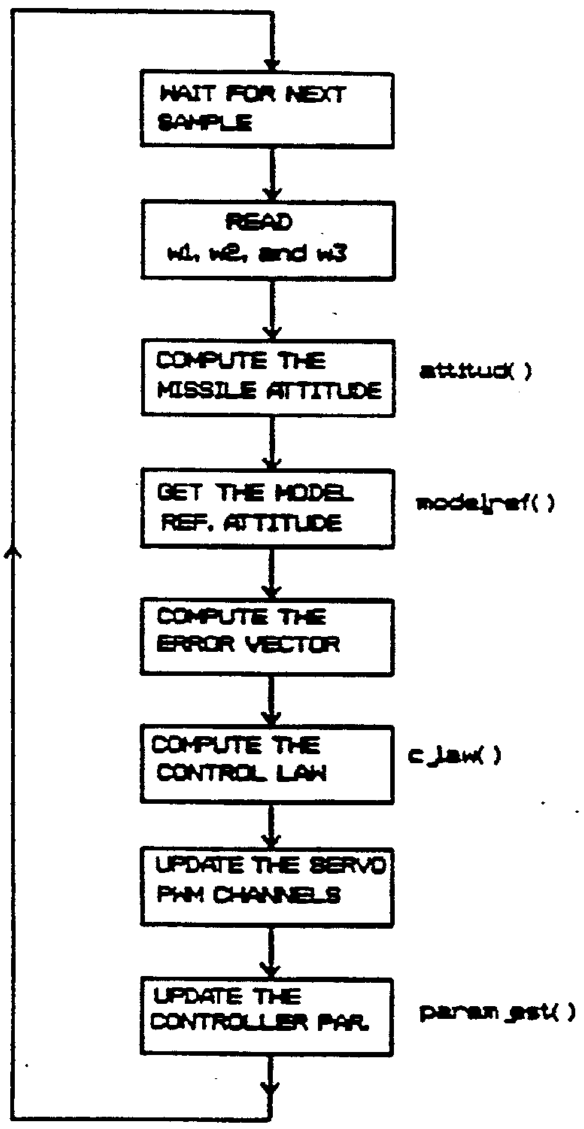


Fig. 7 Program Flow Chart

APPENDIX

```
/************************************************************/
/*                 MISSILE ATTITUDE CONTROL                 */
/************************************************************/
/* Demonstration program for misslie attitude control       */
/* For clarity, memory space minimization is not considered */
/* the program shows computationally efficient calculations */
/* using fixed point arithmetic and short floating point    */
/* arithmetic.                                              */
/* integer numbers here mean 16-bit signed binary number of */
/* absolute value less than one.                            */
/* floating point numbers are 24 bit ( 16-bit mantissa,     */
/* 6 bit exponent, and two sign bits                        */
/* ******************************************************** */
/* T                     ; Sample period = 0.05 sec.        */
/* P1, P2, P3, P4, P5, and P6; found from a Lyapunov equation */
/* N1, N2, N3, N4, and N5   ; determine the dynamics of     */
/*                          controller parameters evolution */
/* BETA1, BETA2, BETA3,     ; initial values of the         */
/* BETA4, BETA5             controller parameters.          */
/* Ai1, Ai2, Ai3, Bi1, Di0  ; for i = 1, 2, 3. parameters of */
/* Di1, SKF_i1, SKF_i2      ; the model reference.          */
/* SKF_Z1, SKF_Z2, SKF_Z3   ; scale factors for control     */
/*                          surfaces deflection.            */
/* W1_SKF, W2_SKF, W3_SKF   ; Scale factors for angular     */
/* SKFF_W1, SKFF_W2, SKFF_W3 ; velocity mesurements from gyros */
/* ******************************************************** */
#include "stdio.h"
#include "data.h"  /* file contains all constants           */
/* ******************************************************** */
int     q1, q2, q3, q4 ;
int     w1_val, w2_val, w3_val, r1, r2, r3;
int     y1t_1, y1t_2, y2t_1, y2t_2, y3t_1, y3t_2;
int     c11, c12, c13, c23, c00, c0;
float   beta1, beta2, beta3, beta4, beta5;
float   theta_1, theta_2, theta_3   ;
float   dtheta_1, dtheta_2, dtheta_3;
float   thetam_1, thetam_2, thetam_3;
float   dthetam_1, dthetam_2, dthetam_3;
float   e_1, e_2, e_3, de_1, de_2, de_3 ;
float   zeta_1, zeta_2, zeta_3;
float   alpha_new, alpha_old, e_new, e_old, de_new, de_old;
/* ******************************************************** */

main()
  {
        float x;
        q1          = 0; q2         = 0; q3         = 0; q4 = L ;
        thetam_1    = 0; thetam_2   = 0; thetam_3   = 0;
        dthetam_1 = 0; dthetam_2 = 0; dthetam_3 = 0;
        beta1       = BETA1; beta2 = BETA2; beta3 = BETA3;
        beta4       = BETA4; beta5 = BETA5;
loop: while(timer != 00) x = -x ;/* wait for the next sample */
/* read w1, w2, w3, from the gyros                                  */
       w1_val = ad_ch(0);

       w2_val = ad_ch(1);

       w3_val = ad_ch(2);

/* get the attitude set points (normalized)
/* from the guidance system                                         */
       r1 = r_port(1);                                              */
       r2 = r_port(2);
       r3 = r_port(3);

/* find the attitude of the missile
       attitud() ;                                                  */

/* find the desired reference attitude
       model_ref();                                                 */

/* compute the error in the attitude
       e_1 = theta_1 - thetam_1 ;                                   */
       e_2 = theta_2 - thetam_2 ;
       e_3 = theta_3 - thetam_3 ;

/* and the rate of change of error
       de_1 = dtheta_1 - dthetam_1 ;                                */
       de_2 = dtheta_2 - dthetam_2 ;
       de_3 = dtheta_3 - dthetam_3 ;

/* now compute the control law
       c_law();                                                     */

/* drive the servo mechanisms of the control surfaces               */
       zeta_1 = SKF_Z1*zeta_1;
       z      = int(zeta_1)   ;
       pwm_ch(0,z);

       zeta_2 = SKF_z2*zeta_2;
       z      = int(zeta_2)   ;
       pwm_ch(1,z);

       zeta_3 = SKF_Z3*zeta_3;
       z      = int(zeta_3)   ;
       pwm_ch(2,z);

/* finally update the parameters of the controller                  */
       param_est()

/* repeat for ever                                                  */
       goto loop
}
```

5802

```
*************************************************************** */
Computes the missile attitude by integrating                    */
the quaternion differential equation (2.3.4)                    */
Integration is performed using Pade approximation               */
CONSTANTS:                                                      */
     W1_SKF, W2_SKF, W3_SKF, SKFF_W1, SKFF_W2, SKFF_W3          */
INPUT     :                                                     */
     w1_val, w2_val, w3_val, q1, q2, q3                         */
OUTPUT    :                                                     */
     updated q1, q2, q3, theta_1, theta_2, theta_3,             */
     dtheta_1, detha_2, and dtheta_3                            */
     c00, c0, c11, c12, c13, c33, c23;                          */
*************************************************************** */
t attitud()
*************************************************************** */

  int  w1_t, w2_t, w3_t;
  int  d1, d2, d3, tmp1, tmp2, tmp3,tmp4;

  w1_t  = w1_val*W1_SKF;/* skf = wi_max*T/L1, L1 = 4 */
  w2_t  = w2_val*W2_SKF;/* wi_max*T < PI */
  w3_t  = w3_val*W3_SKF;

  d1    = (w1_t*w1_t) + (w2_t*w2_t) + (w3_t*w3_t) ;
  /* this is d1 of eq. (2.3.20) */
  d2    = 0.5+ d1/2 ;
  d3    = 0.25/d2   ; /* this is d3 of  eq. 2.3.20 */
  d2    = 0.5- d1/2 ;
  d2    = (d2*d3)*4 ; /* this is d2 of eq. 2.3.20  */
  w1_t  = w1_t*d3 ;
  w2_t  = w2_t*d3 ;
  w3_t  = w3_t*d3 ;
  tmp1  = q1*d2 + q2*w3_t - q3*w2_t + q4*w1_t      ;
  tmp2  = q2*d2 - q1*w3_t + q3*w1_t + q4*w2_t      ;
  tmp3  = q3*d2 + q1*w2_t + q2*w1_t - q4*w3_t      ;
  tmp4  = q4*d2 + q1*w1_t + q2*w2_t - q3*w3_t      ;
  q1    = tmp1   ;
  q2    = tmp2   ;
  q3    = tmp3   ;
  q4    = tmp4   ;

  c00   = 4( 0.25 - 2*q1*q2*q3*q4) ;
  c0    = int_sqrt(c00)     ;
  tmp1  = q4*q4 - q2*q2     ;
  tmp2  = q1*q1 - q3*q3     ;
  c11   = tmp1 + tmp2       ; /* cij defined in eq. (2.3.10)  */
  c12   = 2(q1*q2 - q3*q4) ;
  c13   = 2(q1*q3 + q2*q4) ;
  c33   = -tmp1 + tmp2      ;
  c23   = 2(q2*q3 + q1*q4) ;
  tmp1  = int_atan( -c23,c33); /* normalized theta1 */
  tmp2  = int_atan( c13,c0) ;  /* normalized theta2 */
  tmp3 =  int_atan( -c12,c11); /* normalized theta3 */
```