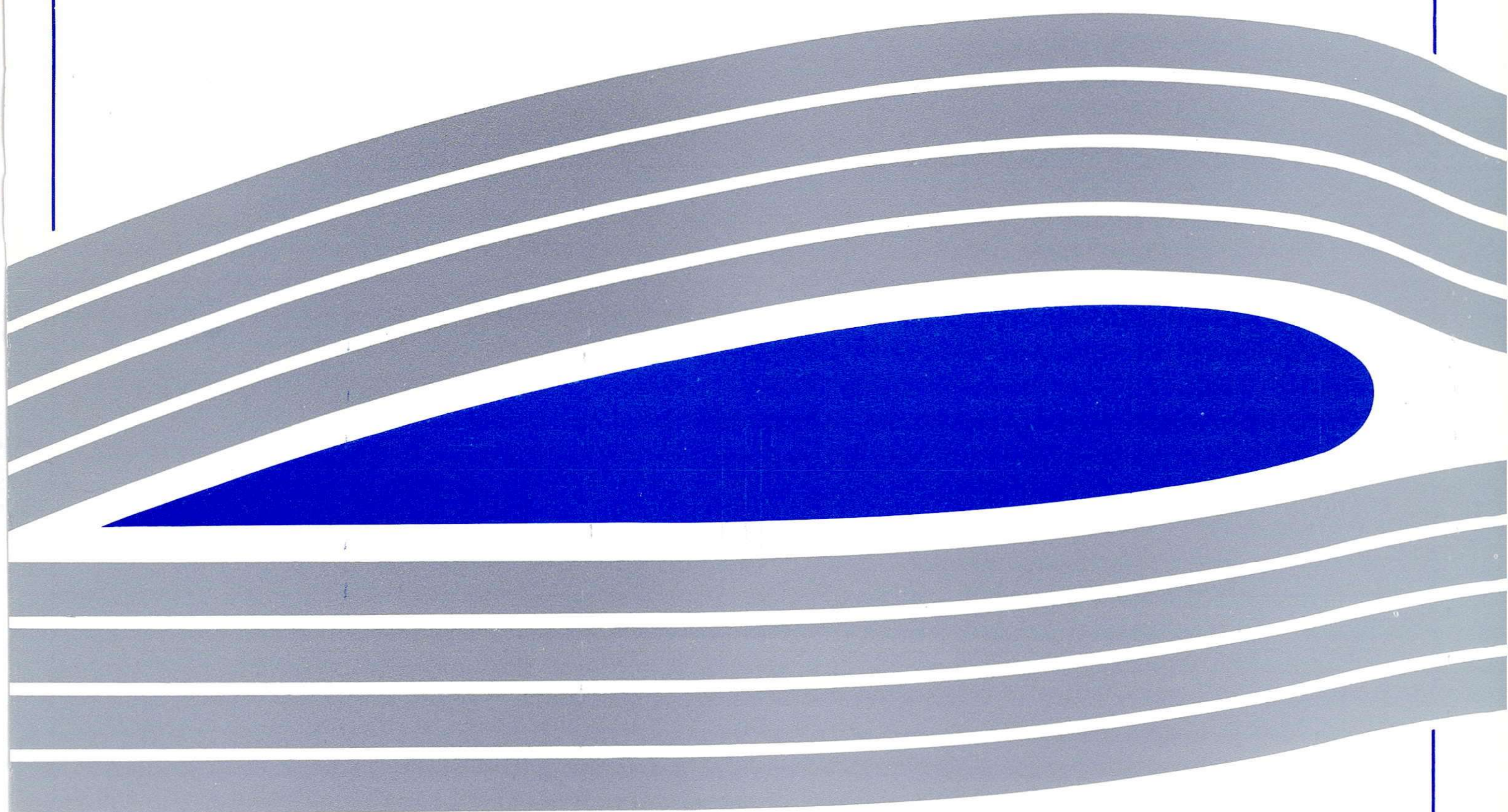
University of Glasgow
DEPARTMENT OF
AEROSPACE
ENGINEERING

# Development of a Generic Inverse Simulation Algorithm

Stephen Rutherford, Dr. Douglas G. Thomson

Internal Report No. 9410
July 1994



# Development of a Generic Inverse Simulation Algorithm

Stephen Rutherford, Dr. Douglas G. Thomson

Internal Report No. 9410
July 1994

# Summary

This report details the development of an inverse simulation algorithm based on numerical time integration. By contrast with algorithms using differentiation, *Gisa* suffers none of the problems associated with numerical instability. Also, being independent of the equations of motion, the algorithm is applicable to any vehicle. The results presented here are for a Westland Lynx helicopter.

# Contents

# Nomenclature

| | | |
|---|---|---|
| $g$ | acceleration due to gravity | $(m/s^2)$ |
| $h$ | height of obstacle in hurdle-hop manoeuvre | (m) |
| $I_R$ | inertia of main rotor | $(kg\ m^2)$ |
| $I_{tr}$ | effective inertia of transmission and gearing | $(kg\ m^2)$ |
| $I_{xx}, I_{yy}, I_{zz}$ | helicopter moments of inertia about centre of gravity | $(kg\ m^2)$ |
| $I_{xz}$ | helicopter product of inertia about y-axis | $(kg\ m^2)$ |
| $K_3$ | overall gain of engine/rotorspeed governor | (Nm/rad/s) |
| $l_1, ..., n_3$ | direction cosines for Euler transformation | |
| $L, M, N$ | components of external moments on vehicle | (Nm) |
| $m$ | helicopter mass | (kg) |
| $p, q, r$ | components of helicopter angular velocity at centre of gravity | (rad/s) |
| $Q_E$ | engine torque output | (Nm) |
| $s$ | track in hurdle-hop manoeuvre | (m) |
| $t_m$ | time taken to complete manoeuvre | (s) |
| $u, v, w$ | translational velocity components of helicopter centre of gravity | (m/s) |
| $V$ | helicopter flight velocity | (m/s) |
| $x_E, y_E, z_E$ | displacements relative to the earth fixed inertial frame | (m) |
| $X, Y, Z$ | components of external force on vehicle | (N) |

## Greek Symbols

| | | |
|---|---|---|
| $\theta_0$ | main rotor collective pitch angle | (rad) |
| $\theta_{1_s}, \theta_{1_c}$ | main rotor longitudinal and lateral cyclic pitch angles | (rad) |
| $\tau_{e_1}, \tau_{e_2}, \tau_{e_3}$ | engine and rotorspeed governor time constants | (s) |
| $\phi, \theta, \psi$ | body roll, pitch and sideslip attitude angles | (rad) |
| $\chi, \gamma$ | track and climb angles | (rad/s) |
| $\Omega$ | angular velocity of main rotor | (rad/s) |
| $\Omega_{idle}$ | angular velocity of main rotor at idle | (rad/s) |
| $\Omega_{TR}$ | angular velocity of tail rotor | (rad/s) |

## Vectors and Matrices

| | |
|---|---|
| $\mathbf{x}$ | state vector |
| $\mathbf{y}$ | output vector |
| $\mathbf{u}$ | input vector |
| $\mathbf{F}$ | error function |
| $[\mathbf{J}]$ | Jacobian matrix |

## Subscripts and Prefices

| | |
|---|---|
| CALC, DES | calculated and desired values |
| e | equilibrium or trim component |
| i, j | matrix indices |
| k | solution point |
| m | iteration number |
| Δ | perturbation component |

# 1. Introduction

Within the field of flight mechanics there is a growing interest in the subject of *inverse simulation.* Inverse simulation can be used to predict a set of control inputs that will cause a predefined displacement of a subject vehicle. More specifically, in mathematically defining some desired vehicle manoeuvre or flight path the algorithm will solve the equations of motion for a unique time history of control inputs. This contrasts with *conventional simulation*, which calculates the vehicle state variables (and consequently flight path) in response to imposed control inputs. A more formal definition can be obtained by considering the initial value problem which expresses the relationship between state, control and output vectors of a dynamic system and forms the basis of most vehicle simulations.

$$\dot{\mathbf{x}} = \mathbf{f}\,(\mathbf{x}, \mathbf{u});\;\; \mathbf{x}(0) = \mathbf{x}_e \tag{1}$$

$$\mathbf{y} = \mathbf{g}\,(\mathbf{x}) \tag{2}$$

The equations of motion (1) permit prediction of the behaviour of the state vector $\mathbf{x}$ in response to an imposed control vector $\mathbf{u}$ over a specified time period, $\mathbf{x}_0$ containing the state variables at $t = 0$. The output equation (2) states how the output vector $\mathbf{y}$ can be obtained from the state vector. More specific representations of equations (1) and (2), relating to helicopter simulation, can be found in Appendix A. These equations also summarise inverse simulation, which predicts the control vector $\mathbf{u}$ that will produce a desired output vector $\mathbf{y}$, and in doing so, the corresponding state vector $\mathbf{x}$.

The applications of inverse simulation are manyfold, particularly in the study of helicopter operations where manoeuvres are often flight path orientated. Indeed, the influence of manoeuvres on helicopter performance has been recognised by the authors of the current U.S. Military Handling Qualities Requirements [1]. One of the most successful and widely used inverse simulations is *Helinv*, developed in the Department Of Aerospace Engineering at the University of Glasgow [2]. It was written initially for the study of helicopter agility [3], and has subsequently been applied to investigations of handling qualities [4], offshore safety [5], and model validation, where comparisons with flight data have shown that *Helinv* predicts actual piloting strategy with reasonable accuracy [6]. There are, however, several inherent problems and limitations associated with the current algorithm. This report details the development of a new algorithm, *Gisa* (General Inverse Simulation Algorithm), designed to overcome these weaknesses.

The following section discusses the existing algorithm and its limitations, and suggests a new algorithm to overcome them. Development of the new algorithm is discussed in Section 3 while Section 4 documents its testing and verification. Finally the proposed applications of this method are presented in the conclusions.

## 2. The *Helinv* Algorithm : An Overview

The most basic feature of the current *Helinv* algorithm is its use of numerical differentiation to calculate rates of change of time dependent variables in the equations of motion (1), effectively converting them from first order differential to non-linear algebraic form. (The standard form of the equations of motion is given in Appendix A; A1-1 to A1-7). A Newton-Raphson method [7] is then used to solve the seven functions (B1-1 to B1-7) in Appendix B for seven unknown variables: the attitude angles $\theta$ and $\phi$, the control angles $\theta_0$, $\theta_{1s}$, $\theta_{1c}$ and $\theta_{0tr}$ and the rotorspeed $\Omega$. A more detailed discussion of the algorithm is given in Appendix B. Although the equations of motion are applicable to any vehicle, the calculation of aerodynamic forces and moments requires detailed, specific modelling; *Helinv*, for example, uses the model HGS, representing a single rotor helicopter, and its characteristics are described in Appendix C and treated in a more comprehensive manner in Reference [8]. One of the most fundamental elements of an inverse simulation is modelling of the required vehicle response - in this case the flight path to be flown, $\mathbf{y}$. A great strength of *Helinv* is the large library of manoeuvres available including nap-of-the-earth, air-to-air-combat, offshore operations and mission task elements [9], [10], and an example of such a model (the hurdle-hop) is described in Appendix D. As indicated in the introduction, however, there are some inherent weaknesses with algorithms that use numerical time differentiation, such as *Helinv*, and these are now discussed.

### 2.1 Limitations of Algorithms which use Numerical Time Differentiation

There are two main disadvantages to inverse methods using numerical differentiation.

i) Modelling enhancements require restructuring of the algorithm.

Referring to equations A1-1 to A1-7 in Appendix A clearly the current model consists of six body plus the rotorspeed degree of freedom. Enhancements requiring the addition of extra degrees of freedom will inevitably lead to modifying the existing

equations and hence the procedure by which they are solved. Different genres of aircraft - *Helinv* for example is limited to single main and tail rotor vehicles - would similarly require changes to the solution algorithm as they may possess different sets of states and controls. Clearly the helicopter model forms an integral part of the solution procedure, and thus changing the model requires changing the algorithm.

ii) It is generally accepted within mathematical literature that numerical differencing causes instability and inaccuracy.

The process of numerical time differentiation causes problems due to the wide range of frequencies associated with the vehicle modes. Helicopters for example, can have many degrees of freedom (e.g. 6 body, 1 rotorspeed, 6 flap, 6 lag, 3 inflow). The rigid body modes may be of a low frequency (less than 0.5 Hz) whereas the flapping mode frequencies may be in the region of 5 Hz. As the algorithm employs numerical differentiation it is vital that the time step chosen to discretise the problem captures all of the dynamic characteristics of the aircraft. This implies very small steps to model adequately the high frequency modes which in turn can produce computational problems due to rounding errors when subtracting small differences in states influenced by the slowly changing, low frequency, rigid body modes; a phenomenon observed in the current algorithm when the blade flapping dynamics are included.

An alternative algorithm, which addresses these problems, is discussed in the next section.

## 3. General Algorithm for Inverse Simulation : An Overview

The basis of a general algorithm for inverse simulation is as follows. Once the problem has been discretised, the aim is to produce a set of applied control inputs which will move the vehicle from its current position and heading to a new desired position and heading over a specified small time step. If the correct series of step inputs were applied to the model at the start of the interval, and equations (1) and (2) were solved in the conventional manner (by use of numerical integration) then they would produce precisely the required position and heading at the end of the interval. Hence, given an estimate of these control step inputs, and by calculating the resultant values for position and heading change, it should be possible, by iteration, to obtain the required control steps. By comparison with the method discussed earlier which solves seven functions, consisting of the equations of motion, for **u** and additional unknowns $\phi$, $\theta$ and $\Omega$, the proposed algorithm solves, by a Newton-Raphson method,

four functions of the output vector $\mathbf{y}$ for the control vector $\mathbf{u}$. This is an algorithm first suggested by Hess, [11], [12] and [13].

Although this algorithm is still prone to numerical errors, they should be much smaller than in the case of numerical differentiation. Also enhancements or fundamental modelling changes may be accommodated by replacing the functions $\mathbf{f}(\mathbf{x}, \mathbf{u})$ (1) without altering the structure of the algorithm. The one major disadvantage of this algorithm is the greatly increased computational time required. This and other implications of using the proposed technique will be discussed in Section 4, whilst in the following section a full description of a numerical integration based algorithm, *Gisa*, is presented.

## 3.1 *Gisa* - A Generic Inverse Simulation Algorithm

What follows is an explanation of the programme *Gisa* which uses an algorithm based on integration for inverse simulation. The example given uses the same model as *Helinv* - a single main and tail rotor helicopter. Firstly an overview, more formal than in the previous section, will be presented, and then a detailed discussion with reference to a flowchart.

Consider that the problem is discretised into a series of time points $t_k$ at each of which there is a predefined desired output vector $\mathbf{y}_{DES}(t_k)$ describing the position and heading of the helicopter, Figure (1). At the current time point $t = t_k$, the value of $\mathbf{x}(t_k)$ is known from solution of the previous time point $t = t_{k-1}$; $\dot{\mathbf{x}}(t_{k-1})$ having been integrated using, for example, a Runge-Kutta [14] method. The influence of the control vector on $\dot{\mathbf{x}}(t_k)_m$, $\mathbf{x}(t_{k+1})_m$ (by integration) and $\mathbf{y}(t_{k+1})_m$ can be found, using equations (1) and (2), by varying it about the current value $\mathbf{u}(t_k)_m$. The problem is effectively a Newton-Raphson solution for the control vector $\mathbf{u}(t_k)_m$ which will produce a value of $\mathbf{y}(t_{k+1})_m$ equal to $\mathbf{y}_{DES}(t_{k+1})$. The algorithm is now independent of the equations of motion, and depends exclusively on the output vector $\mathbf{y}$ and input vector $\mathbf{u}$ i.e. it is generic. Additionally it uses time integration rather than differentiation so there are no problems caused by different modal frequencies. It should be noted that though Jacobian evaluation (§3.2) also involves numerical differentiation,the derivatives are with respect to the control angles, change slowly only as the solution converges, and are thus more stable.

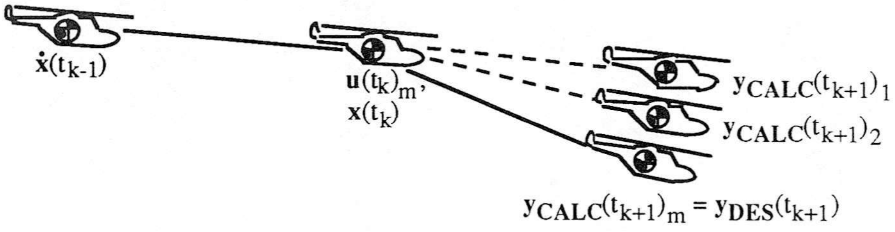


**Figure 1.** Helicopter at Previous, Current and Next Time Points Showing State, Control and Output Vectors

Referring to the flowchart of the solution procedure given in Figure 2, a step by step explanation of the algorithm is now given.

### 3.2 *Gisa* : a Newton-Raphson Solution Algorithm for the Control Vector $\mathbf{u}$

The inverse simulation begins, either by calculating a specified trim state or reading appropriate trim values from a dedicated trimmer, and then reading a predefined desired manoeuvre (Appendix D) from data files. The heart of the algorithm lies within the loop for the k$^{th}$ time point, a conventional simulation for k = 0 to n, around which there are m Newton-Raphson iterations per loop of k. The following will discuss the 'heart' of the algorithm, with later, more detailed explanation of the Jacobian calculation.

The initial solution occurs at t = 0, the value of $\mathbf{x}$ being known from the trim value $\mathbf{x_e}$ and the first estimate of $\mathbf{u}$ taken as the trim value $\mathbf{u_e}$. In the general case (the mth estimate at the kth time point) $\dot{\mathbf{x}}(t_k)_m$ can be evaluated using $\mathbf{x}(t_k)$ and the current estimate for $\mathbf{u}(t_k)_m$.

$$\dot{\mathbf{x}}(t_k)_m = \mathbf{f}\,[\mathbf{x}(t_k),\, \mathbf{u}(t_k)_m] \tag{3}$$

This in turn can be integrated, using a Runge-Kutta method for example, to produce estimates of $\mathbf{x}(t_{k+1})_m$ and $\mathbf{y}(t_{k+1})_m$ at the next time point.

$$\mathbf{x}(t_{k+1})_m = \int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}[(t_k)_m]\,dt + \mathbf{x}(t_k)_m \tag{4}$$

$$\mathbf{y}(t_{k+1})_m = \mathbf{g}\,[\mathbf{x}(t_{k+1})_m] \tag{5}$$

As the basis for a Newton-Raphson solution, an error function is defined as the difference between the latest estimate of the output vector, $\mathbf{y}(t_{k+1})_m$ and the desired value $\mathbf{y}_{\mathbf{DES}}(t_{k+1})$.

$$\mathbf{F_m} = \mathbf{y}(t_{k+1})_m - \mathbf{y}_{\mathbf{DES}}(t_{k+1}) \tag{6}$$

The function is tested against a predefined tolerance. If less then the tolerance then the programme moves on the next solution point k+1 and continues from equation (3). However if $\mathbf{F_m}$ is greater than the tolerance then a Jacobian is calculated and using its inverse (7), a new estimate of the control vector $\mathbf{u}(t_k)_{m+1}$ can be found. The Jacobian is a matrix evaluated by differentiating the output vector with respect to the control vector (8), the practicalities of which are detailed in the next section.

$$\mathbf{u}(t_k)_{m+1} = \mathbf{u}(t_k)_m - [\mathbf{J}]^{-1}\,\mathbf{F_m} \tag{7}$$

$$[\mathbf{J}] = \frac{d\,\mathbf{y}(t_{k+1})_m}{d\,\mathbf{u}(t_k)_m} \tag{8}$$

This new estimate is then used to calculate $\dot{\mathbf{x}}(t_k)_{m+1}$ and consequently the error function $\mathbf{F_{m+1}}$ within the m loop, m = m + 1. The programme is finished at the end of the manoeuvre time period (k=n). Evaluation of the Jacobian and its inverse is now discussed.

### 3.3 Evaluation of the Jacobian [J] and its Inverse $[\mathbf{J}]^{-1}$

Again consider the $m^{th}$ estimate at the $k^{th}$ time point. The Jacobian is a 4 x 4 matrix, the entries of which $j_{ij}(t_k)_m$ are evaluated by differentiating each of the elements of the output vector $y_i(t_{k+1})_m$ with respect to each of the elements of the control vector $u_j(t_k)_m$. This can be represented in general form.

$$j_{ij}(t_k)_m = \frac{\partial\, y_i(t_{k+1})_m}{\partial\, u_j(t_k)_m} \tag{9}$$

A more specific representation of the matrix can be made in terms of the actual output and control elements used within *Gisa*,

$$[J] = \begin{bmatrix} \frac{\partial x_E}{\partial \theta_0} & \frac{\partial x_E}{\partial \theta_{1s}} & \frac{\partial x_E}{\partial \theta_{1c}} & \frac{\partial x_E}{\partial \theta_{0tr}} \\ \frac{\partial y_E}{\partial \theta_0} & \frac{\partial y_E}{\partial \theta_{1s}} & \frac{\partial y_E}{\partial \theta_{1c}} & \frac{\partial y_E}{\partial \theta_{0tr}} \\ \frac{\partial z_E}{\partial \theta_0} & \frac{\partial z_E}{\partial \theta_{1s}} & \frac{\partial z_E}{\partial \theta_{1c}} & \frac{\partial z_E}{\partial \theta_{0tr}} \\ \frac{\partial \psi}{\partial \theta_0} & \frac{\partial \psi}{\partial \theta_{1s}} & \frac{\partial \psi}{\partial \theta_{1c}} & \frac{\partial \psi}{\partial \theta_{0tr}} \end{bmatrix} \qquad (10)$$

| | |
|---|---|
| where $x_E$, $y_E$, $z_E$ | are displacements relative to an earth fixed inertial frame, |
| $\psi$ | is the azimuth or heading angle and |
| $\theta_0$, $\theta_{1s}$, $\theta_{1c}$, $\theta_{0tr}$ | are the blade pitch angles of the main and tail rotors. |

Within the programme, however, there are no analytical expressions for the output vector **y** and so the Jacobian's elements must be calculated numerically, the general representation of which is given below.

$$\frac{\partial y_i(t_{k+1})_m}{\partial u_j(t_k)_m} = \frac{y_i(t_{k+1}, u_j(t_k)+\delta u_j(t_k))_m - y_i(t_{k+1}, u_j(t_k)-\delta u_j(t_k))_m}{2\delta u_j(t_k)_m} \qquad (11)$$

It is clear that all four output elements must be calculated at positive and negative perturbations from their current estimates and hence equations (3), (4) and (5) must be used a further eight times. The Jacobian inverse $[J]^{-1}$ is approximated by Crout's method [15], new estimates $\mathbf{u}(t_k)_{m+1}$ are calculated and the iteration continues. With the algorithm now described, some of the results produced by *Gisa* are discussed in the next section.

## 4. Presentation and Validation of Results Obtained Using *Gisa*

### 4.1 The Mathematical Model

The mathematical model used in *Gisa* is a partial non-linear model i.e. within the equations of motion the gravitational and inertial terms are non-linear but linearised versions of the external forces and moments are used. The form of this partial non-linear model will be discussed later, but first an explanation of why it was chosen. Consider, as was stated earlier, that the main modelling effort required is that of the external forces and moments, an exercise which is very time consuming. Although detailed modelling is necessary, and indeed will be the next period of

research, the main aim at this stage was the development of the generic inverse simulation algorithm, and bearing this in mind a linearised model, using derivatives evaluated in the HGS package, was deemed adequate. Linearised models are universally accepted as being accurate for small perturbations from the trim state but are nonetheless, by their vary nature, only approximations. Thus in order to increase the model fidelity without incurring large modelling changes, the evaluation of the inertial and gravitational terms are in non-linear form, and fully linearised terms used to calculate the external forces and moments. This in fact was also the form of model used by Hess [11].

Considering equation A1-1 in Appendix A, the equation for translational acceleration along the x body axis,

$$m \dot{u} = -(w q - v r) - g \sin\theta + X \tag{12}$$

and that the force X can be expressed in terms of a trim and perturbation component, i.e.,

$$X = X_e + \Delta X \tag{13}$$

then equation (12) can be converted to the following form.

$$m \dot{u} = -(w q - v r) - g \sin\theta + X_e + \Delta X \tag{14}$$

$X_e$ can be calculated from the expression below,

$$X_e = m\,(q_e\, w_e - v_e\, r_e + g \sin\theta_e) \tag{15}$$

and the term $\Delta X$ can be calculated using derivatives from the linearised module of HGS mentioned in Appendix C. Thus the equations of motion can be evaluated using non-linear inertial and gravitational terms and fully linearised force and moment components, obtained by adding the trim values (as in equation (15)) to the perturbed values of the form below.

$$\begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ \Delta L \\ \Delta M \\ \Delta N \end{bmatrix} = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ L_u & L_v & L_w & L_p & L_q & L_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{bmatrix} \begin{bmatrix} u' \\ v' \\ w' \\ p' \\ q' \\ r' \end{bmatrix} + \begin{bmatrix} X_{\theta 0} & X_{\theta 1s} & X_{\theta 1c} & X_{\theta 0tr} \\ Y_{\theta 0} & Y_{\theta 1s} & Y_{\theta 1c} & Y_{\theta 0tr} \\ Z_{\theta 0} & Z_{\theta q1s} & Z_{\theta 1c} & Z_{\theta 0tr} \\ L_{\theta 0} & L_{\theta 1s} & L_{\theta 1c} & L_{\theta 0tr} \\ M_{\theta 0} & M_{\theta 1s} & M_{\theta 1c} & M_{\theta 0tr} \\ N_{\theta 0} & N_{\theta 1s} & N_{\theta 1c} & N_{\theta 0tr} \end{bmatrix} \begin{bmatrix} \theta_0' \\ \theta_{1s}' \\ \theta_{1c}' \\ \theta_{0tr}' \end{bmatrix} \qquad (16)$$

$$\text{where} \quad X_u = \frac{\partial X}{\partial u} \quad \text{etc}$$

A number of results calculated using this model and the fully linearised version are presented in the next section. Additionally, comparisons are made with similar results obtained using *Helinv*.

## 4.2 Verification and Validation of Algorithm

For the purpose of validating the results produced by *Gisa*, the test manoeuvre chosen was a hurdlehop, for two main reasons. Firstly it is a manoeuvre used to drive *Helinv* whose results have been tested extensively [6], thus allowing verification of *Gisa's* results. Secondly the linearised force and moment terms in *Gisa* mean manoeuvres involving large deviations from the trim state are unsuitable as they render the model and consequently inverse solution inaccurate. Considering the hurdlehop manoeuvre described in Appendix D, Figure 3 (h=15m, s=500m, V=80kts) shows that the accelerations are both small, less than 0.25g, and confined exclusively to the longitudinal (x-z plane), and so a hurdlehop does not demand excessive deviations from the initial and final trim states. Additionally the severity of a hurdlehop manoeuvre can be varied simply by changing the obstacle height, and so the influence of severity on solution accuracy is easily investigated.

Figures 4 and 5 also show results for the above hurdlehop. In Figure 4 the desired output parameters $x_E$, $y_E$, $z_E$ and $\psi$ are compared to those calculated by *Gisa*. As the differences between these results are the functions which *Gisa* solves then the algorithm has clearly converged. The solutions of these four functions, the control inputs, or rotor blade pitch angles, $\theta_0$, $\theta_{1s}$, $\theta_{1c}$ and $\theta_{0tr}$, are shown in Figure 5, which can be verified as follows; the low amplitude, high frequency oscillations which are apparent will be discussed later. With reference to the hurdlehop diagram (Figure D4) in Appendix D, a positive application of collective, which controls vertical acceleration, is required in the climbing phase. This becomes negative just before the half way point in order to attain zero vertical velocity at the peak of the hurdlehop

profile. Input is negative throughout the descent until the helicopter approaches the manoeuvre exit, when a positive input is applied to level off at zero vertical velocity. Considering the first period corresponding to positive collective, the other three results can be justified as follows. The form of the lateral cyclic and tail rotor collective plots are similar to the main rotor collective while longitudinal cyclic's form is approximately a mirror image about the y-axis. An application of positive collective will tilt the rotor disc and consequently thrust vector backwards, so the longitudinal cyclic must be such as to maintain forward velocity i.e. negative tilts the rotor disc forwards. The applied collective will also increase the pitch and thus drag of the blades, and so to conserve rotorspeed the shaft exerts extra anticlockwise torque which produces a clockwise reaction on the fuselage. In order to stop the helicopter rotating, a restoring moment must be supplied by the tail rotor; this is achieved by increasing tail rotor collective in response to main rotor collective. Lateral cyclic is used to balance the right side force caused by the tail rotor; a positive input tilting the thrust vector to the left. Apparent on the plots for tail rotor collective and longitudinal and lateral cyclic are low amplitude, high frequency oscillations. These are mentioned by Hess [10], [11] and Thomson suggests that they are a consequence of an effective, infinite gain, feedback loop on position which modifies the dynamic charecteristics of the vehicle [16]. The rest of the manoeuvre is similarly justified, so the results would appear to be realistic.

Comparisons between the results calculated by *Gisa* and *Helinv* are illustrated in the first thirteen plots of Figure 6; four control angles, six state velocities and three attitude angles. The last three plots compare desired track, altitude and flight path velocity with those calculated by *Gisa*. Results for the four control angles, with the exception of *Gisa's* low amplitude, high frequency oscillations, are clearly very similar (these oscillations appear in *Helinv* if a smaller differentiation time step is used, Thomson [17]). Both collective and longitudinal cyclic vary only slightly in their peak amplitudes, while lateral cyclic and tail rotor collective, though not quite as accurate, also give good correlation. There is also a good match between *Helinv* and *Gisa's* state variables, particularly the longitudinal terms, u, w, q and $\theta$, which are closely linked to the hurdlehop definition, itself a longitudinal manoeuvre; note the similarity between u and w, and the earth fixed velocities $\dot{x}_E$ and $\dot{z}_E$ in Figure 4 as defined by the hurdlehop. This ties in with the very close collective and longitudinal cyclic results; controls which predominantly influence longitudinal motion. Similarly, as with the lateral cyclic and tail rotor collective results, the lateral state variables, v, p, r, $\phi$ and $\psi$ are not quite as accurate as the longitudinal, though still very good. The flight path parameters, track, altitude and velocity, of which accurate calculation are the algorithm's goal, agree very well with the desired values. Overall

the results are very similar, any discrepancies in the results accountable for by the different models, particularly considering that the aerodynamic derivatives used in *Gisa's* linearised terms are for a longitudinal trim state. Results such as these suggest that the algorithm is performing correctly.

## 4.3 Testing the Algorithm

It has been demonstated in the previous section that the algorithm works for a specific case. To further validate the algorithm's 'performance', the following section discusses the effect of factors such as manoeuvre severity, size of integration time step and convergence conditions. As well as investigating how the result accuracy is affected, comparisons are made of run times both for varied parameters and for comparison with *Helinv*.

### Hurdlehop Severity

Results for more severe hurdlehops are shown in Figures 7 and 8. The manoeuvres involve increased severity as hurdles of 25m and 35m respectively have to be cleared over the same track of 500m, so both the accelerations and consequently deviations from initial and final trim states are greater. As expected *Gisa's* results, though similar, do not match *Helinv's* as well as in Figure 6. The fact that the results have converged accurately (plots 14 - 16) supports the idea of modelling differences rather than weaknesses in the algorithm.

A shorter, lower hurdlehop was also investigated. Although the manoeuvre (0.15m height, 50m track) involves similar maximum accelerations as that in Figure 6, the much shorter period means greater jerk (demonstrated by consideration of the rate of change rather than amplitude of the control inputs). Considering how this affects trim deviations, it is no surprise that, although the accuracy of the hurdlehop has not been compromised, the match between *Gisa* and *Helinv* has deteriorated considerably.

The idea that a non-linear model would improve the accuracy is further supported by results (not shown) for a longitudinal acceleration (Figure 9) which involves much greater trim deviations. The desired parameters are matched but control inputs and states do not compare as favourably with *Helinv* as in the hurdlehop. To extrapolate, the pattern of accuracy loss since the good correlation in Figure 6 indicates that although the linear model is limited, the algorithm is valid.

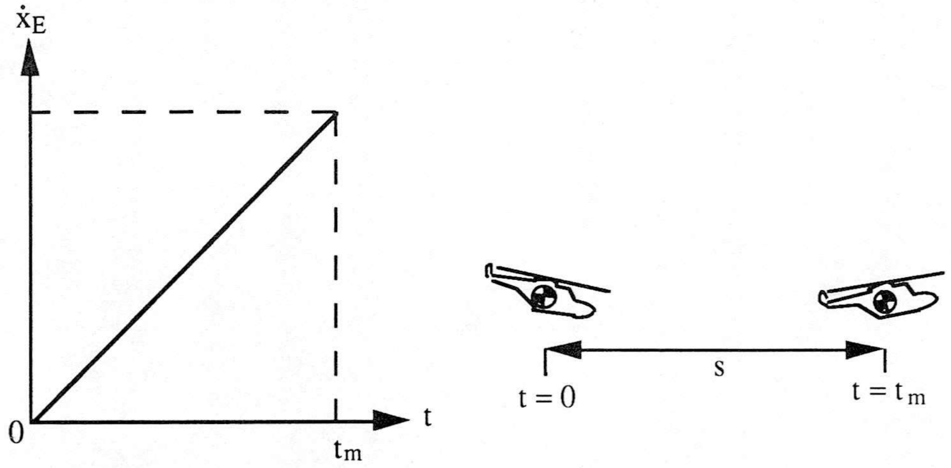


**Figure 9 Longitudinal Acceleration ($\dot{y}_E$ and $\dot{z}_E$ equal zero)**

### Integration Time Step

The authors of [17] suggest that numerical integration within algorithms such as *Gisa* should be very accurate for a wide range of time steps, but that if "there is an uncontrolled state variable, the integration inverse method may be unstable for small step time". In *Gisa* many states are uncontrolled but there is no evidence of the high frequency oscillations predicted in the reference. Tests have been made for time steps covering the range mentioned. Very little difference in accuracy has been found due to varying the time step. The greatest consideration has been finding a time step to smoothly capture all of the modes but not incur too long a cpu time - too great a time step may require many iterations to solve, too small a step involves too many solution points. It is hoped to find time in the future to test the model in [17] with *Gisa's* algorithm and compare results.

### Effect of Perturbation Size

The perturbation size used in calculating the Jacobian (equation (11)) has negligible effect on the accuracy of the results, suggesting a robust algorithm, though does influence the run time as can be seen in the tables below. In order to make any firm conclusions, however, some experimentation will be needed with a variety of manoeuvres.

### 'Nag' Tolerance, Convergence Parameter

Tolerance of the Nag (software library) integration is largely determined by the accuracy needed for results; it is pointless to use a very high tolerance, and high

run time, for graphical data if the detail will not be visible in the plots. Similary the convergence tolerance ($\mathbf{y}_{CALC}(t_{k+1})_m$ - $\mathbf{y}_{DES}(t_{k+1})$) should agree with the desired accuracy, but obviously cannot exceed that of the integration. Results were obtained for Nag / convergence tolerances of $10^{-3}$ / $10^{-2}$, $10^{-5}$ / $10^{-4}$ and $10^{-9}$ / $10^{-8}$ which were an exact match when plotted but as can be seen in Table 1, run times varied from 5 minutes to 2 hours. Again, the algorithm would appear to be accurate and so the choice of tolerance depends upon the intended application of the results.

The definition of convergence is of more concern, particularly considering convergence to desired values of zero such as $y_E(t_{k+1})$ and $\psi(t_{k+1})$ in a hurdlehop. It is for this reason that absolute differences, rather than percentage errors were chosen for $\mathbf{y}_{DES}(t_{k+1})$ and $\mathbf{y}_{CALC}(t_{k+1})_m$ as a percentage of values tending to zero requires consideration of the computing accuracy. An additional advantage of this is that absolute differences aid the choice of the Nag tolerance mentioned above.

### Effect of Matching Steps / Ramps with Displacements / Velocities

The algorithm used to generate results in this report uses earth fixed velocities and yaw rate - $[\dot{x}_E, \dot{y}_E, \dot{z}_E, \dot{\psi}]^T$ - as components of the output vector $\mathbf{y}(t_{k+1})_m$ and solves the problem for rate of change of control inputs, $[\dot{\theta}_0, \dot{\theta}_{1s}, \dot{\theta}_{1c}, \dot{\theta}_{0tr}]^T$ i.e. matches ramp inputs with velocities.

| | Control vector $\mathbf{u}(t_k)_m$ | Output vector $\mathbf{y}(t_{k+1})_m$ |
|---|---|---|
| Case (1) Gisa SD<br>(steps / displacements) | $[\theta_0, \theta_{1s}, \theta_{1c}, \theta_{0tr}]^T$ | $[x_E, y_E, z_E, \psi]^T$ |
| Case (2) Gisa SV<br>(steps / velocities) | $[\theta_0, \theta_{1s}, \theta_{1c}, \theta_{0tr}]^T$ | $[\dot{x}_E, \dot{y}_E, \dot{z}_E, \dot{\psi}_E]^T$ |
| Case (3) Gisa RD<br>(ramps / displacements) | $[\dot{\theta}_0, \dot{\theta}_{1s}, \dot{\theta}_{1c}, \dot{\theta}_{0tr}]^T$ | $[x_E, y_E, z_E, \psi]^T$ |
| Case (4) Gisa RV<br>(ramps / velocities) | $[\dot{\theta}_0, \dot{\theta}_{1s}, \dot{\theta}_{1c}, \dot{\theta}_{0tr}]^T$ | $[\dot{x}_E, \dot{y}_E, \dot{z}_E, \dot{\psi}_E]^T$ |

**Table 1** **Different Combinations of Steps / Ramps and Displacements / Velocities**

Results were also generated for the other combinations in Table 1. The choice of steps or ramps made no difference to the solution accuracy. However, as can be

seen in Figure 10 (Case 3 vs. Case 4), the choice of constrained velocities or displacements does change the results; constrained displacements produce more accurate values of $y_E$ (plot 14) and $\psi$ (plot 13), constrained velocities produce better values of V (plot 16). The reasons for this have yet to be investigated.

### Fully Linearised Model

Figure 11 shows a comparison of results between the fully -linear versions of *Helinv* (Helinvl) and *Gisa.* There is clearly good correlation between the results, whereas the linear and non-linear versions of *Helinv* are appreciably different. This observation is further evidence of the algorithm's validity. Constraining velocities or displacements has a similar effect as with the non-linear model.

### Comparison of cpu Times

In addition to calculating accurate results it is also desirable to minimise the amount of time taken for the algorithm to run. The tables below compare cpu times for the different cases discussed above.

*Gisa* vs. *Helinv* for hurdlehops of increasing severity (tolerance = $10^{-5}$, perturbation size = 0.05, s = 500m):

| h (m)<br>max load factor | 15<br>1.198 | 25<br>1.309 | 35<br>1.427 |
|---|---|---|---|
| | cpu time (min:sec) | | |
| Gisa RV | 10:02.47 | 10:40.63 | 11:10.57 |
| Helinv | 0:53.11 | 0:58.95 | 1:01.52 |

Effect of perturbation size and tolerance (hurdlehop - s=500m, h=15m) :

| perturbation<br>tolerance | 0.05<br>$10^{-3}$ | 0.05<br>$10^{-9}$ | 0.001<br>$10^{-5}$ | 1.0<br>$10^{-5}$ |
|---|---|---|---|---|
| cpu time (hr:min:sec) Gisa RV | 0:05:11.79 | 2:37:49.85 | 0:13:49.73 | 0:09:49.32 |

Effect of steps / ramps, displacements / velocities (hurdlehop - s=500m, h=15m, tolerance = $10^{-5}$, perturbation size = 0.05) :

| algorithm : | Gisa SD | Gisa SV | Gisa RD | Gisa RV |
|---|---|---|---|---|
| | | cpu time (min:sec) | | |
| non-linear | 5:27.68 | 7:26.15 | 6:55.03 | 8:52.32 |
| Helinv | 0:53.11 | | | |
| linear | 1:56.62 | 1:55.95 | 1:58.11 | 1:57.17 |
| Helinvl | 0:09.85 | | | |

From the above tables the following conclusions can be made.

i) The numerical differentiation algorithm (*Helinv*) runs a lot more quickly than the integration algorithm (*Gisa*).
ii) The fully-linearised model runs a lot more quickly than the non-linear model.
iii) Perturbation size has little effect on cpu time.
iv) Tolerance greatly influences cpu time and should be selected to suit the application of the results.
v) Choice of Cases (1) to (4) does not greatly change cpu times.

## 5. Conclusions

1. The integration algorithm for inverse simulation used within *Gisa* accurately predicts a time series of control inputs which will produce a predefined, discretised manoeuvre.

2. Although the results are currently valid for a limited set of manoeuvres only, it is anticipated that this is due to the linear aircraft modelling used. A detailed non-linear model should allow good results for a wider range of more severe manoeuvres.

3. The algorithm appears to be robust, working regardless of time step size, perturbation size and tolerance.

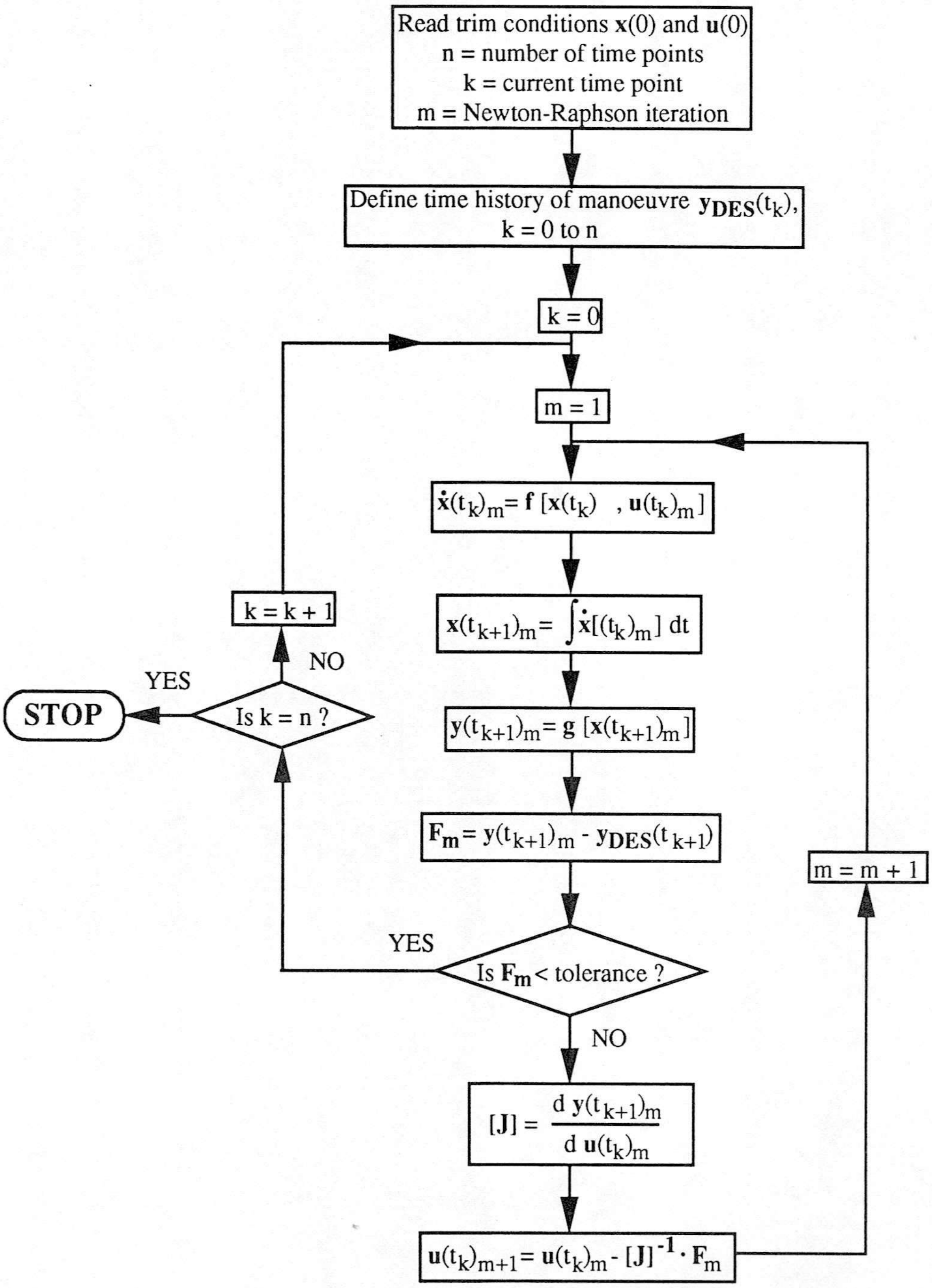
4. Computing time is a major disadvantage as the algorithm takes much longer to run than the differentiation method (*Helinv*). This will not be a problem, however, as more powerful computers become available.

5. There is a discrepency in the results calculated by using constrained displacements or velocities. This has to be investigated further.

6. At this stage the algorithm appears to be both accurate and stable. It is predicted that any modelling changes will pose no problems and hence *Gisa* should be truly generic.

7. The next planned stage in the algorithm's development is to incorporate a helicopter individual blade model. Subsequent applications will depend on the accuracy of results obtained.

Figure 2. Flowchart for Generalised Inverse Simulation Algorithm



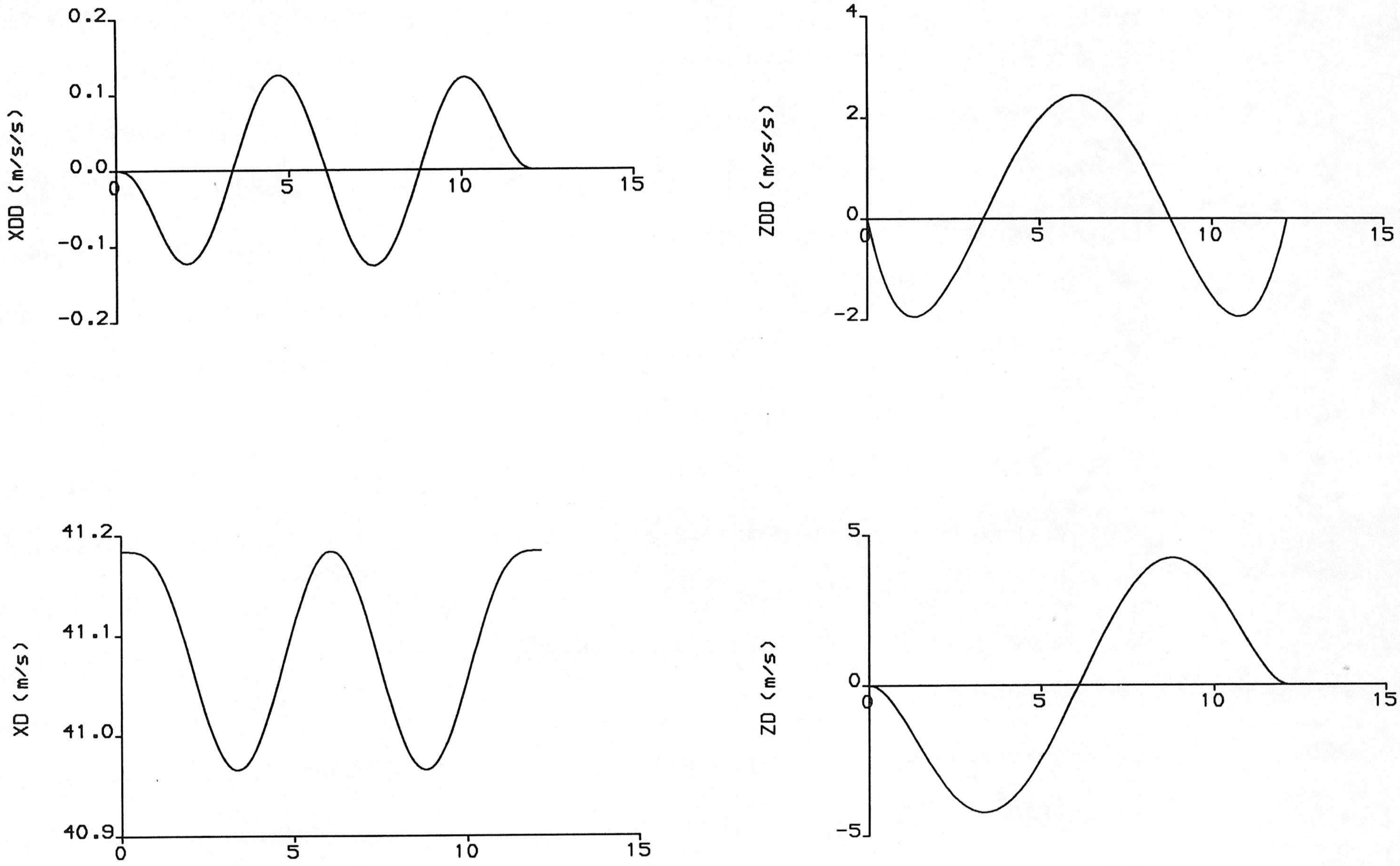


Figure 3 Earth Axis Accelerations and Velocities vs. Time (Hurdlehop - s =500m, h=15m)

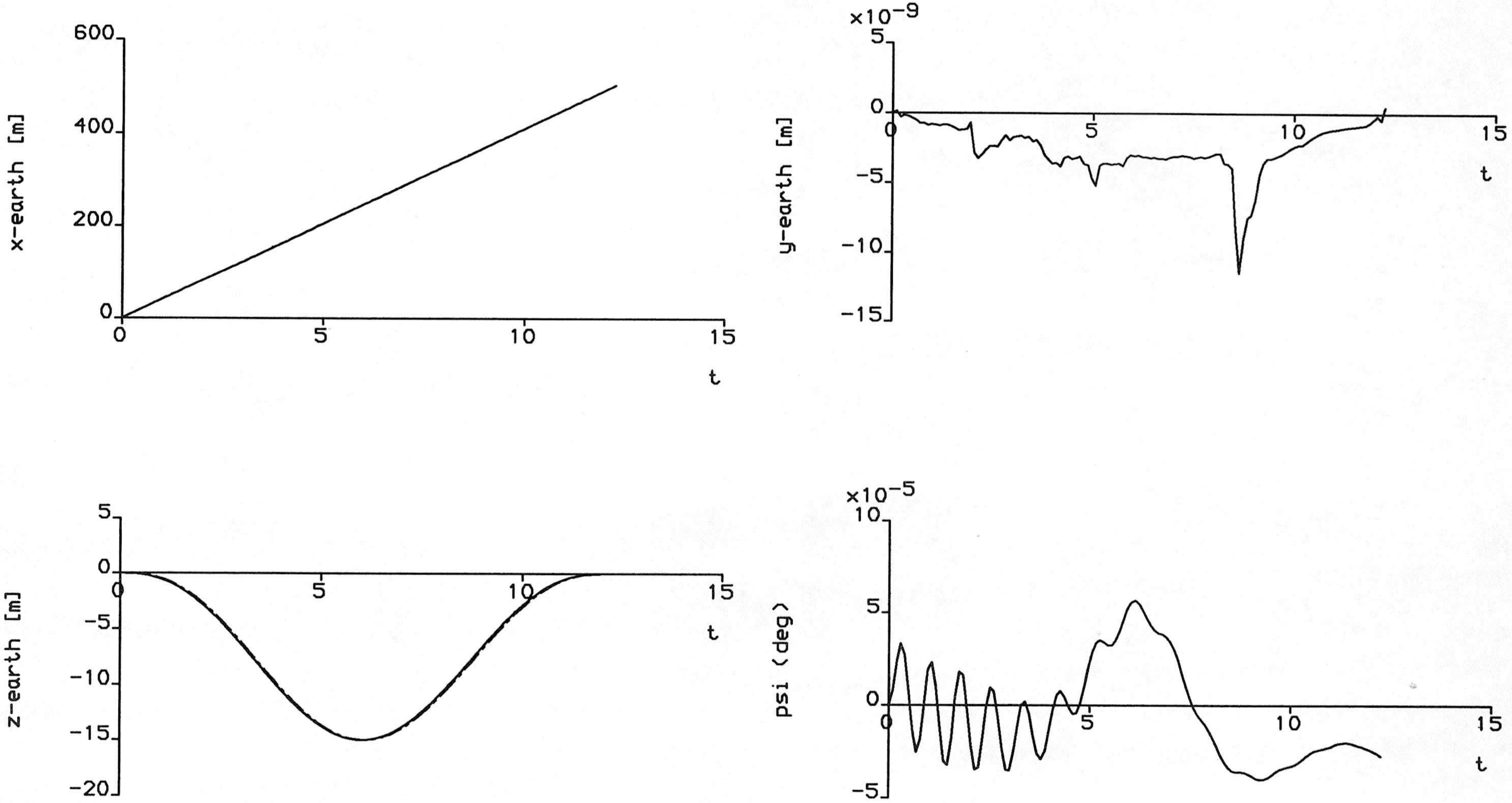
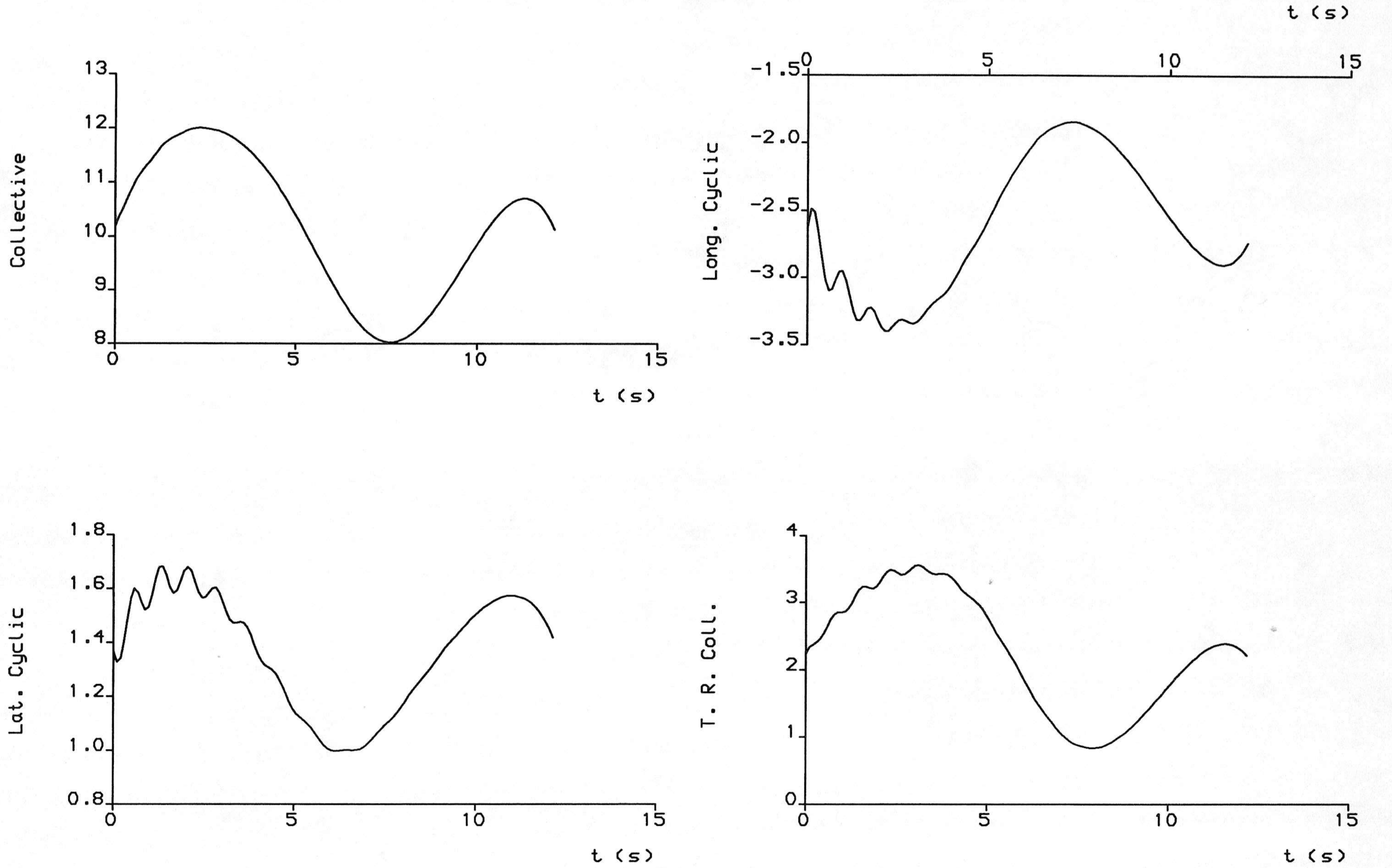


**Figure 4** Comparison Between Desired (Hurdlehop) and Calculated (Gisa) Manoeuvre Parameters

*Plotted on 5-JUL-94 at 17:00:07*

Figure 5 Control Angles vs. Time for a Hurdlehop Manoeuvre (s=500m, h=15m)



Control Angles (deg) vs. Time.

Plotted on 5-JUL-94 at 16:37:27

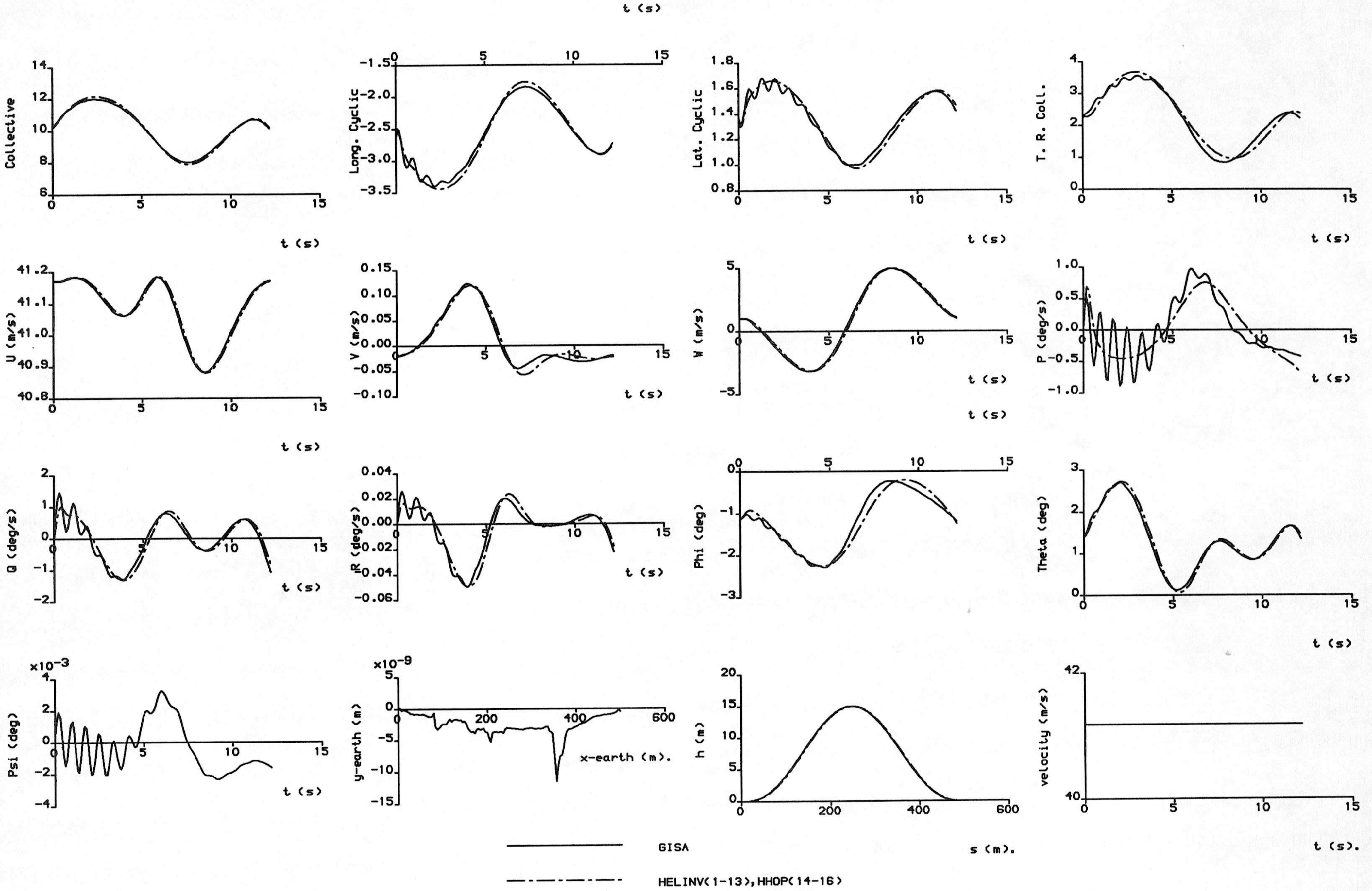


Figure 6 Comparison Between Helinv / Hurdlehop and Gisa - Control, State and Flight Path Variables (s=500m, h=15m)

Plotted on 5-JUL-94 at 16:47:08

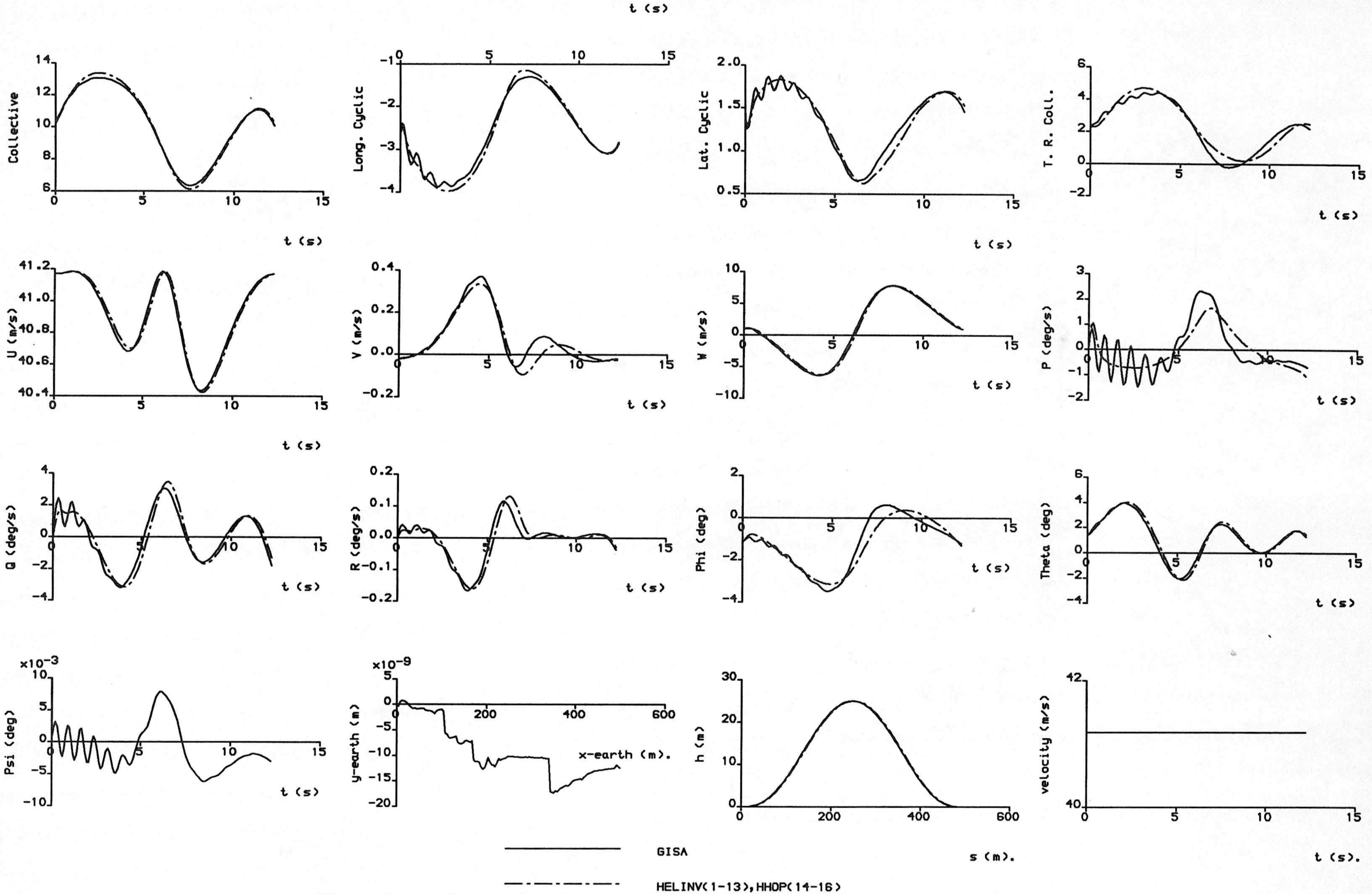


Figure 7 **Comparison Between Heliny / Hurdlehop and Gisa - Control, State and Flight Path Variables** (s=500m, h=25m)

*Plotted on 5-JUL-94 at 16:49:38*

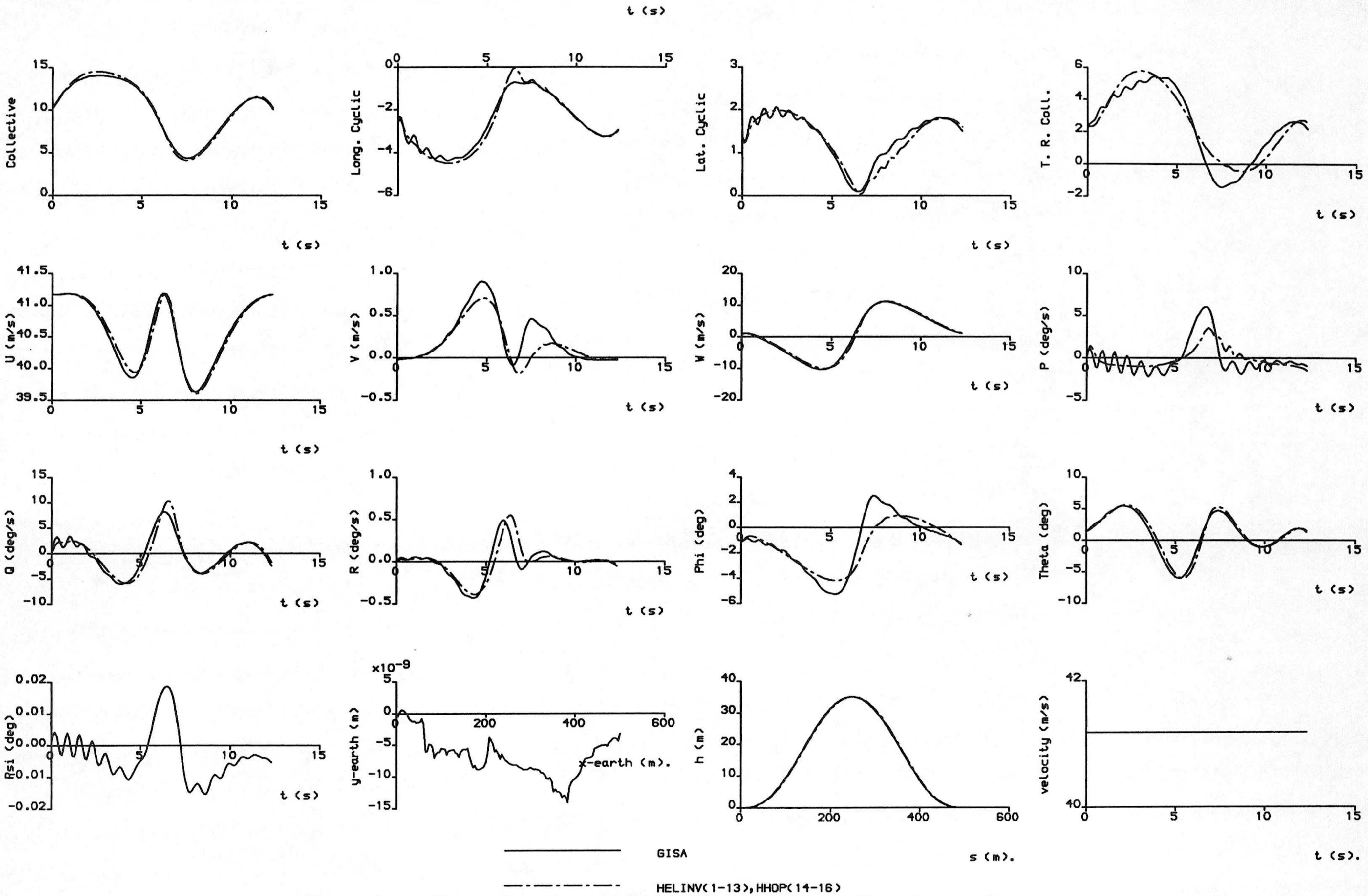


Figure 8 <u>Comparison Between Helinv / Hurdlehop and Gisa - Control, State and Flight Path Variables</u> (s=500m, h=35m)

*Plotted on 5-JUL-94 at 16:51:50*

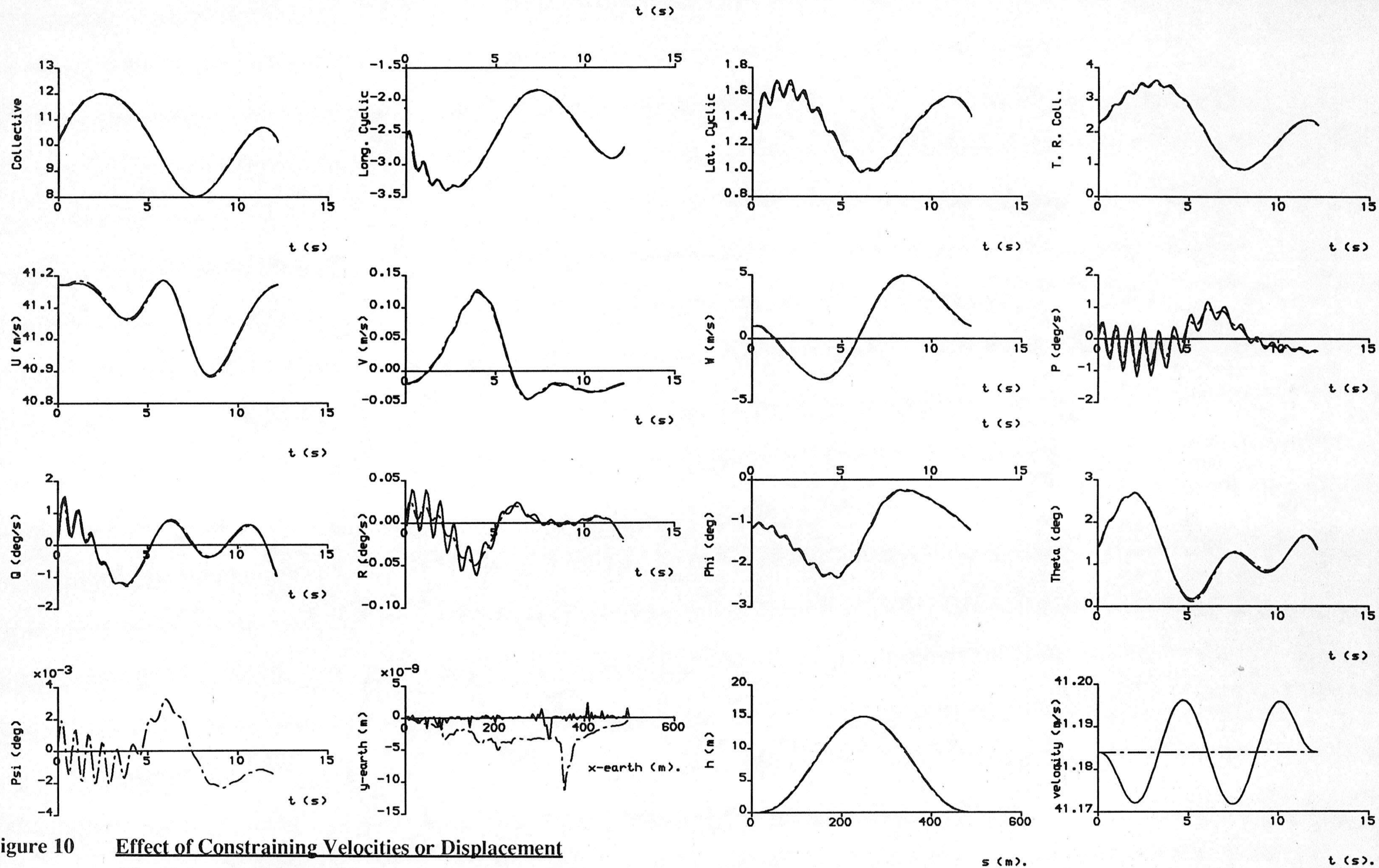


Figure 10 **Effect of Constraining Velocities or Displacement**

(Hurdlehop s= 500m, h=15m)

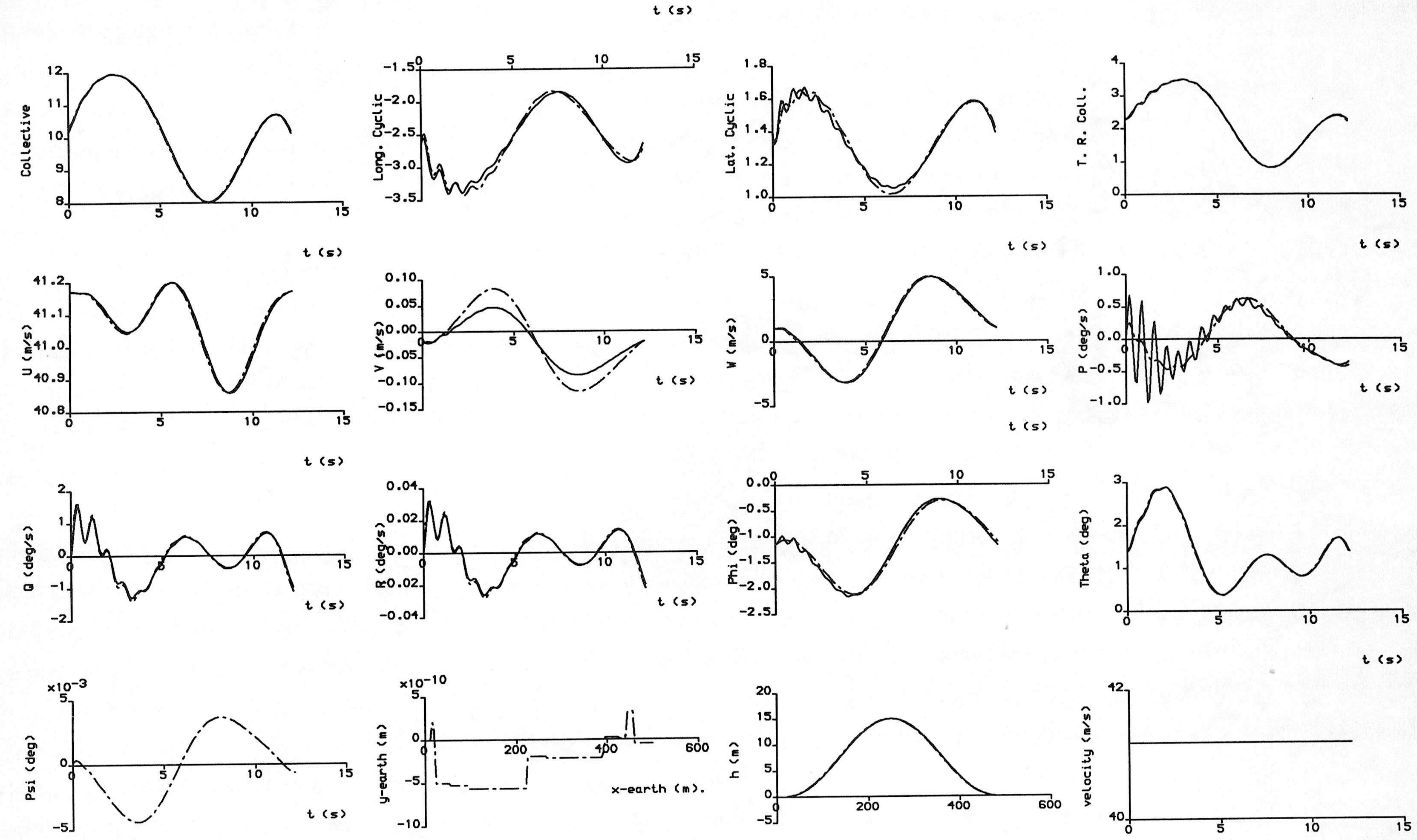


**Figure 11** Comparison Between Fully Linearised Versions of *Gisa* and *Helinv*
(Hurdlehop s=500m, h=15m)

## Appendix A Equations of Motion of a Single Main and Tail Rotor Helicopter

Considering equations (1) and (2) as applied to a single main and tail rotor helicopter; the state, control and output vectors are defined as follows :

$$\mathbf{x} = [\, u\; v\; w\; p\; q\; r\; \phi\; \theta\; \psi\; \Omega \,]^T$$

$$\mathbf{u} = [\, \theta_0\; \theta_{1s}\; \theta_{1c}\; \theta_{0tr} \,]^T$$

$$\mathbf{y} = [\, x_E\; y_E\; z_E\; \psi \,]^T$$

| | |
|---|---|
| where $u, v, w$ | are the components of translational velocity relative to a body fixed reference frame ($x_b$, $y_b$, $z_b$), |
| $p, q, r$ | are angular velocities about the body axes, |
| $\phi, \theta, \psi$ | are the Euler (or attitude) angles relating the body fixed axis set to the earth fixed inertial frame ($x_E$, $y_E$, $z_E$), |
| $\Omega$ | is the angular velocity of the main rotor and |
| $\theta_0, \theta_{1s}, \theta_{1c}, \theta_{0tr}$ | are the blade pitch angles of the main and tail rotors. |

The function **f** in equation (1) consists essentially of the following six Euler rigid body equations :

$$\dot{u} = -(wq - vr) + \frac{X}{m} - g\sin\theta \qquad \text{(A1–1)}$$

$$\dot{v} = -(ur - wp) + \frac{Y}{m} - g\cos\theta\sin\phi \qquad \text{(A1–2)}$$

$$\dot{w} = -(vp - uq) + \frac{Z}{m} - g\cos\theta\cos\phi \qquad \text{(A1–3)}$$

$$I_{xx}\dot{p} = (I_{yy} - I_{zz})\, q r + I_{xz}(\dot{r} + p q) + L \qquad \text{(A1-4)}$$

$$I_{yy}\dot{q} = (I_{zz} - I_{xx})\, r p + I_{xz}(r^2 - p^2) + M \qquad \text{(A1-5)}$$

$$I_{zz}\dot{r} = (I_{xx} - I_{yy})\, p q + I_{xz}(\dot{p} - q r) + N \qquad \text{(A1-6)}$$

plus the rotorspeed governor equation :

$$\dot{\Omega} = \frac{(Q_E - Q_R - Q_{TR} - Q_{tr})}{I_R} + \dot{r} \qquad \text{(A1-7)}$$

| | | |
|---|---|---|
| where | $m, I_{xx}, I_{yy}, I_{zz}, I_{xz}$ | are the aircraft's mass, moments of inertia and product of inertia about the y-axis, |
| | X, Y, Z, L, M, N | are the external forces and moments along and about the body axes, |
| | $Q_E, Q_R, Q_{TR}, Q_{tr}$ | are the torque output of the engine, and torques required to drive the main rotor, tail rotor and transmission, and |
| | $I_R$ | is the effective inertia of the rotor system. |

The rates of change of the attitude angles are related to the body axes angular velocities by the kinematic expressions.

$$\dot{\phi} = p + q \sin\phi \tan\theta + r \cos\phi \tan\theta \tag{A2–1}$$

$$\dot{\theta} = q \cos\phi - r \sin\phi \tag{A2–2}$$

$$\dot{\psi} = q \sin\phi \sec\theta + r \cos\phi \sec\theta \tag{A2–3}$$

The earth fixed velocities $\dot{x}_E$, $\dot{y}_E$ and $\dot{z}_E$ can be calculated from the translational body fixed velocities u, v and w and the attitude angles $\phi$, $\theta$ and $\psi$ by the Euler transformation equations where the transformation matrix $[l_1, \ldots, n_3]$ is effectively the function **g** in equation (2).

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A3}$$

$l_1 = \cos\theta \cos\psi$

$l_2 = \cos\theta \sin\psi$

$l_3 = -\sin\theta$

$m_1 = \sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi$

$m_2 = \sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi$

$m_3 = \sin\phi \cos\theta$

$n_1 = \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi$

$n_2 = \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi$

$n_3 = \cos\phi \cos\theta$

With the exception of (A1-7) the above representations of equations (1) and (2) are not unique to the helicopter, but are widely used in many rigid body

simulations. It is the calculation of the external forces X, Y and Z; and moments L, M and N which requires detailed, specific modelling. An example of this is the HGS model used within *Helinv* as summarised in Appendix C.

## Appendix B. The *Helinv* Numerical Algorithm

As with *Gisa, Helinv* incorporates several sets of pre-programmed manoeuvre descriptions which are required as system outputs from the simulation. In fact, the manoeuvres are essentially the inputs for the simulation. By contrast with *Gisa, Helinv's* output vector only contains the eath fixed coordinates. The ouput vector is then :

$$\mathbf{y} = [x_E, y_E, z_E]^T$$

and the azimuth or heading angle, $\psi$ is additionally constrained to fully define the manoeuvre. The earth fixed coordinates $x_E$, $y_E$ and $z_E$ are primarily influenced by longitudinal cyclic, lateral cyclic and collective respectively, and so $\psi$, the variable most influenced by tail rotor collective, is chosen as the fourth manoeuvre variable.

Solving the equations of motion for the current problem, it has been established that the`defining variables are the flight path coordinates ($x_E$, $y_E$ and $z_E$) and aircraft heading, $\psi$, and that the principle aim is to obtain the unknown control inputs which will produce them ($\theta_0$, $\theta_{1s}$, $\theta_{1c}$ and $\theta_{0tr}$). This is achieved by solution of the six body equations of motion and the engine equation (A1-1 to A1-7), with the remaining three unknowns being the fuselage pitch and roll attitudes ($\theta$ and $\phi$) and the angular velocity of the main rotor, $\Omega$. This choice may not at first be clear, however if one considers that when the attitude and earth referenced velocity vector of the vehicle are known it is possible to obtain

i) the body referenced velocity vector (u, v, w) and by differentiation the acceleration vector ($\dot{u}$, $\dot{v}$, $\dot{w}$);
ii) differentiation of the attitudes gives the angular velocities (p, q, r) and accelerations ($\dot{p}$, $\dot{q}$, $\dot{r}$);
iii) knowledge of the body velocities will allow the areodynamic forces and moments on the fuselage to be obtained, whilst the control angles and rotorspeed (and all the other state information) ensures that the rotor forces may be found, hence the external forces and moments (X, Y, Z, L, M and N) are available.

Examination of the seven equations of motion will show that there is then enough information to obtain values for all of the terms in them, and hence the equations are soluable. This is now discussed in more detail.

The solution is cast in 'time marching" form - that is the input information (the flight path) is expressed as a time series at equally spaced intervals, and the seven equations of motion are solved at each time point in the series usin the flight path information at that point. If we consider the case of a manoeuvre taking a time $t_m$, which is divided into a time series of n intervals then a general time point in the solution, $t_k$ may be defined as,

$$0 < t_k \leq t_m \quad \text{where} \quad 1 < k \leq (n + 1)$$

The input at this time point is

$$x_E(t_k),\ y_E(t_k),\ z_E(t_k),\ \psi(t_k)$$

which may be differentiated to give

$$\dot{x}_E(t_k),\ \dot{y}_E(t_k),\ \dot{z}_E(t_k),\ \dot{\psi}(t_k) \quad \text{and} \quad \ddot{x}_E(t_k),\ \ddot{y}_E(t_k),\ \ddot{z}_E(t_k),\ \ddot{\psi}(t_k)$$

At each time point values are obtained for all of the unsteady time variant terms in the equations of motion which converts them from differential form to non-linear, algebraic form. The equations are then solved by a Newton-Raphson iterative technique to find the seven unknowns

$$\theta(t_k),\ \phi(t_k),\ \Omega(t_k),\ \theta_o(t_k),\ \theta_{1s}(t_k),\ \theta_{1c}(t_k),\ \text{and}\ \theta_{otr}(t_k)$$

From equations (A1-1 to A1-7) the requirement is then to solve :

$$F_1(\theta(t_k),...,\theta_{otr}(t_k)) = \dot{u}(t_k) + (\ q(t_k) - v(t_k)\ r(t_k)) - \frac{X(t_k)}{m} + g \sin\theta(t_k) = 0 \qquad \text{(B1-1)}$$

$$\vdots$$

$$F_6((\theta(t_k),...,\theta_{otr}(t_k)) = I_{zz}\dot{r}(t_k) - (I_{xx} - I_{yy})\ p(t_k)\ q(t_k) - I_{xz}\ (\dot{p}(t_k) - q(t_k)\ r(t_k)) - N(t_k) = 0 \qquad \text{(B1-6)}$$

$$F_7(\theta(t_k),...,\theta_{otr}(t_k)) = \ddot{Q}_E(t_k)\ \tau_{e_1}\ \tau_{e_2} + (\tau_{e_1} + \tau_{e_3})\dot{Q}_E(t_k) + Q_E(t_k) - K_3\ (\Omega(t_k) - \Omega_{idle} + \tau_{e_2}\ \dot{\Omega}(t_k)) = 0 \qquad \text{(B1-7)}$$

Clearly from the preceding description of the mathematical model there are many intermediate calculations required. The sequence of calculations at the $m^{th}$ iteration is as follows.

i) Initial Guess of UnknownVariables

The estimate from the pevious iteration is used as the initial guess at the current iteration so that

$$\theta(t_k)_m = \theta(t_k)_{m-1}, \quad \phi(t_k)_m = \phi(t_k)_{m-1}, \quad \text{etc.} \quad m > 1$$

For the first iteration the converged values from the previous time point are used :

$$\theta(t_k)_m = \theta(t_{k-1}), \quad \phi(t_k)_m = \phi(t_{k-1}), \quad \text{etc.} \quad m = 1$$

ii) Calculation of the Body Referenced Translational Velocities

The body axes velocities are obtained from the transformation equation (A-2). The expression for $u(t_k)_m$ is

$$\begin{aligned} u(t_k)_m = {} & \dot{x}_E(t_k) \cos\theta(t_k)_m \cos\psi(t_k) + \dot{y}_E(t_k) \cos\theta(t_k)_m \sin\psi(t_k)_m \\ & - \dot{z}_E(t_k) \sin\theta(t_k)_m \end{aligned} \tag{B2}$$

and similar expressions are obtained for $v(t_k)_m$ and $w(t_k)_m$.

iii) Rates of Change of Euler Angles and Rotorspeed

Numerical differentiation is used to obtain the Euler angle rates and rate of change of rotorspeed. Backward differencing is used to give the following for $\dot{\theta}(t_k)_m$ and $\ddot{\theta}(t_k)_m$

$$\ddot{\theta}(t_k)_m = \frac{\theta(t_k)_m - \theta(t_{k-1})}{t_k - t_{k-1}} \quad \text{and} \quad \ddot{\theta}(t_k)_m = \frac{\theta(t_k)_m - 2\theta(t_{k-1}) + \theta(t_{k-2})}{(t_k - t_{k-1})^2} \tag{B-3}$$

and similar expressions may be obtained for $\ddot{\phi}(t_k)_m$, $\ddot{\phi}(t_k)_m$ and $\ddot{\Omega}(t_k)_m$.

iv) Calculation of the Body Referenced Translational Accelerations

The body accelerations are obtained by differentiation of the corresponding velocities, so that for $\dot{u}(t_k)_m$, from equation (B2)

$$\begin{aligned}\dot{u}(t_k)_m = {} & \ddot{x}_E(t_k) \cos \theta(t_k)_m \cos \psi(t_k) + \ddot{y}_E(t_k) \cos \theta(t_k)_m \sin \psi(t_k) - \ddot{z}_E(t_k) \sin \theta(t_k)_m \\ & - \dot{\theta}(t_k)_m \left[(\dot{x}_E(t_k) \cos \psi(t_k) + \dot{y}_E(t_k) \sin \psi(t_k)) \sin \theta(t_k)_m + \cos \theta(t_k)_m\right] \\ & - \dot{\psi}(t_k) \left[\dot{x}_E(t_k) \sin \psi(t_k) - \dot{y}_E(t_k) \cos \psi(t_k)\right] \cos \theta(t_k)_m \qquad \text{(B4)}\end{aligned}$$

and similar expressions may be obtained for $\dot{v}(t_k)_m$ and $\dot{w}(t_k)_m$.

### v) Calculation of Vehicle Angular Velocities and Accelerations

Equations (A3-1 to A3-3) may be recast to give body angular velocities in terms of the Euler angle rates so that, for example, the roll rate, p, may be found from

$$p(t_k)_m = \dot{\phi}(t_k)_m - \dot{\psi}(t_k) \sin \theta(t_k)_m \qquad \text{(B5)}$$

which may be differentiated to give

$$\dot{p}(t_k)_m = \ddot{\phi}(t_k)_m - \ddot{\psi}(t_k) \sin \theta(t_k)_m - \dot{\psi}(t_k) \dot{\phi}(t_k)_m \cos \theta(t_k)_m \qquad \text{(B6)}$$

Expressions for $q(t_k)_m$, $r(t_k)_m$, $\dot{q}(t_k)_m$ and $\dot{r}(t_k)_m$ are obtained in a similar way.

### vi) The External Forces and Moments

Having established estimates for all the vehicle states and controls it is possible to evaluate the corresponding external forces and moments : $X(t_k)_m$, $Y(t_k)_m$, $Z(t_k)_m$, $L(t_k)_m$, $M(t_k)_m$ and $N(t_k)_m$.

### vii) Engine Torque and its Rate of Change

The torque required to turn the main rotor, $Q_R(t_k)_m$, tailrotor, $Q_{TR}(t_k)_m$, and transmission, $Q_{tr}(t_k)_m$, are obtained from the calculation of external forces and moments. These values are then used in association with equation (A1-7) to obtain the required engine torque

$$Q_E(t_k)_m = (\dot{\Omega}(t_k)_m - \dot{r}(t_k)_m) I_R + Q_R(t_k)_m + Q_{TR}(t_k)_m + Q_{tr}(t_k)_m \qquad \text{(B-7)}$$

The rates of change of engine torque, $\dot{Q}_E(t_k)_m$ and $\ddot{Q}_E(t_k)_m$, are calculated by numerical differentiation using the values of engine torque from the previous two time points ($Q_E(t_{k-1})$, $Q_E(t_{k-2})$) in the same way as shown in equation (B3).

It is now possible to obtain values for the seven functions at the mth iteration. If the solution has not converged (i.e. the functions are not within a small tolerance of zero) then new estimates of the unknown variables are found. The new estimates are found from

$$\begin{bmatrix}\theta\\ \vdots\\ \vdots\\ \theta_{0tr}\end{bmatrix}_{(t_k)_{m+1}} = \begin{bmatrix}\theta\\ \vdots\\ \vdots\\ \theta_{0tr}\end{bmatrix}_{(t_k)_m} - \begin{bmatrix}\left(\dfrac{\partial F_1}{\partial \theta}\right) & \cdots\ \cdots & \left(\dfrac{\partial F_1}{\partial \theta_{0tr}}\right)\\ \vdots & & \vdots\\ \vdots & & \vdots\\ \left(\dfrac{\partial F_7}{\partial \theta}\right) & \cdots\ \cdots & \left(\dfrac{\partial F_1}{\partial \theta_{0tr}}\right)\end{bmatrix}^{-1}_{(t_k)_m} \begin{bmatrix}F_1\\ \vdots\\ \vdots\\ F_7\end{bmatrix}_{(t_k)_m} \tag{B8}$$

The Jacobian elements are calculated by numerical differentiation in the same way as desribed in §3.3, though, of course, for functions (B1-1 to B1-7). With new estimates the iteration continues in a conventional way.

## Appendix C HGS (Helicopter Generic Simulation) Mathematical Model

The mathematical model used in *Helinv* is known as HGS (**H**elicopter **G**eneric **S**imulation) and has the following characteristics :

- generic non-linear model of a single main and tail rotor helicopter with seven degrees of freedom (six body plus rotorspeed)
- option to include the extra degrees of freedom for flapping motion and dynamic inflow
- rotor disc model incorporating rigid flapping blades with root cut-out, linear twist, constant chord and 2-dimensional aerodynamic properties
- fuselage, tailplane and fin aerodynamics by look-up tables.

The HGS model also has a suite of programmes available which allow the calculation of general trim states and the response to control inputs. As well as the non-linear model, there is also a linearised version of HGS which allows stability analysis to be performed and is used in a linearised version of *Helinv* for the study of constrained trajectory flight. All of the HGS and *Helinv* modules have dedicated graphics facilities (where) applicable and advanced dynamics graphics tools are under development.

## Appendix D Mathematical Definition of Manoeuvres

When considering the mathematical definition of a manoeuvre two main requirements must be adhered to; that it is able to accurately compute the vehicle displacements and velocities throughout the manoeuvre, and that it displays a physically realisable degree of continuity. For the first, bearing in mind that an inverse simulation calculates a time history of control inputs which will accurately reproduce a desired, predefined manoeuvre, then the displacements, velocities and accelerations must be expressed as functions of time. Secondly if the resultant profile is insufficiently smooth, then rapid changes or discontinuities in the time derivatives of the profile may lead to numerical problems in the inverse algorithm. For simplicity, polynomial representations have been adopted. As both their order and constants are determined by boundary conditions, the combination of prescribed displacements, velocities and accelerations must be considered carefully. For the example of an aircraft undergoing rectilinear acceleration, a choice of initial and final displacements would be less representative than velocities. Additionally the aircraft may require to be in trim at the initial and final points and thus zero acceleration (or even jerk) must be stipulated, otherwise discontinuities would result at the boundaries.

For the purposes of inverse simulation, any manoeuvre is a time history of the output vector **y**, which contains the earth fixed coordinates $x_E$, $y_E$ and $z_E$; and the azimuth or heading angle $\psi$. For some manoeuvres, referring to figures (D1, 2, 3), it is easier and indeed sometimes desirable to express the earth fixed velocities in terms of functions of the flight velocity, V, track angle, $\chi$ and climb angle $\gamma$. The equations for velocities below can subsequently be manipulated to calculate displacements or accelerations.

$$\dot{x}_E(t) = V(t) \cos \gamma(t) \cos \chi(t) \qquad \text{(D1-1)}$$

$$\dot{y}_E(t) = V(t) \cos \gamma(t) \sin \chi(t) \qquad \text{(D1-2)}$$

$$\dot{z}_E(t) = - V(t) \sin \gamma(t) \qquad \text{(D1-3)}$$

When the manoeuvre can be expressed in terms of $x_E$, $y_E$ and $z_E$ explicitly, equations (D1-1 to 3) are obviously unnecessary. A hurdlehop, Figure D4, is a manoeuvre where it is necessary to clear an obstacle, height h at the midway point, over a distance s and then return to the original altitude and velocity. The hurdlehop is performed at constant heading in the longitudinal (x-z) plane. In order to ensure

continuity, initial and final accelerations and jerk are set to zero; the boundary conditions thus being :

i) $t = 0$, a) $V = V_0, \dot{V} = 0, \ddot{V} = 0$; b) $z_E = 0, \dot{z}_E = 0, \ddot{z}_E = 0$

ii) $t = \frac{t_m}{2}$, a) $V = V_{MAX}$; b) $z_E = -h$

iii) $t = t_m$, a) $V = V_0, \dot{V} = 0, \ddot{V} = 0$; b) $z_E = 0, \dot{z}_E = 0, \ddot{z}_E = 0$

where $t_m$ is the time taken to complete the manoeuvre. The seven velocity and $z_E$ boundary conditions allow definition of sixth order polynomials, which can be shown to be of the form :

$$V(t) = \left[\left(\frac{t}{t_m}\right)^6 - 3\left(\frac{t}{t_m}\right)^5 + 3\left(\frac{t}{t_m}\right)^4 - \left(\frac{t}{t_m}\right)^3\right] 64\,(V_0 - V_{MAX}) + V_0 \qquad \text{(D-2)}$$

$$z_E(t) = \left[\left(\frac{t}{t_m}\right)^6 - 3\left(\frac{t}{t_m}\right)^5 + 3\left(\frac{t}{t_m}\right)^4 - \left(\frac{t}{t_m}\right)^3\right] 64\,h \qquad \text{(D-3)}$$

$$y_E(t) = \psi(t) = 0 \qquad \text{(D-4)}$$

The longitudinal displacement $x_E(t)$ can be evaluated numerically using equation (D-5), and (D-6) can be rearranged in terms of the manoeuvre time $t_m$.

$$\dot{x}_E(t) = \sqrt{V^2 - \dot{z}_E(t)^2} \qquad \text{(D-5)}$$

$$s = \int_0^{t_m} \dot{x}_E(t)\, dt \qquad \text{(D-6)}$$

In essence then, the complete manoeuvre may be defined by the parameters s, h, $V_0$ and $V_{MAX}$.

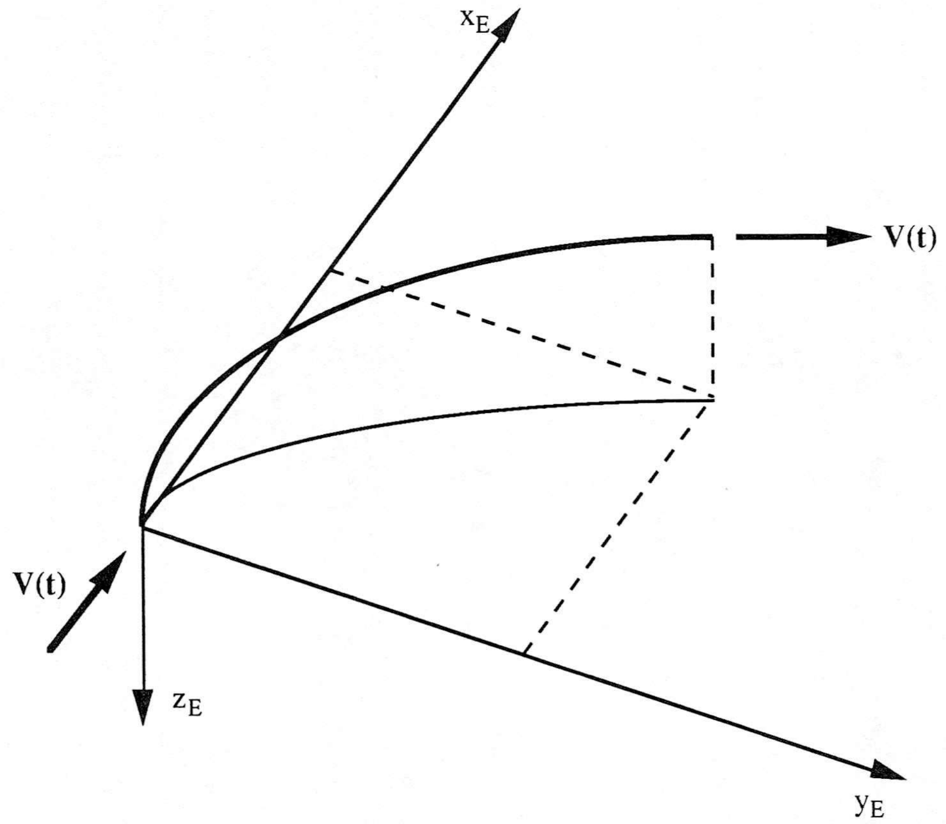


Figure D1 The 3-D Manoeuvre

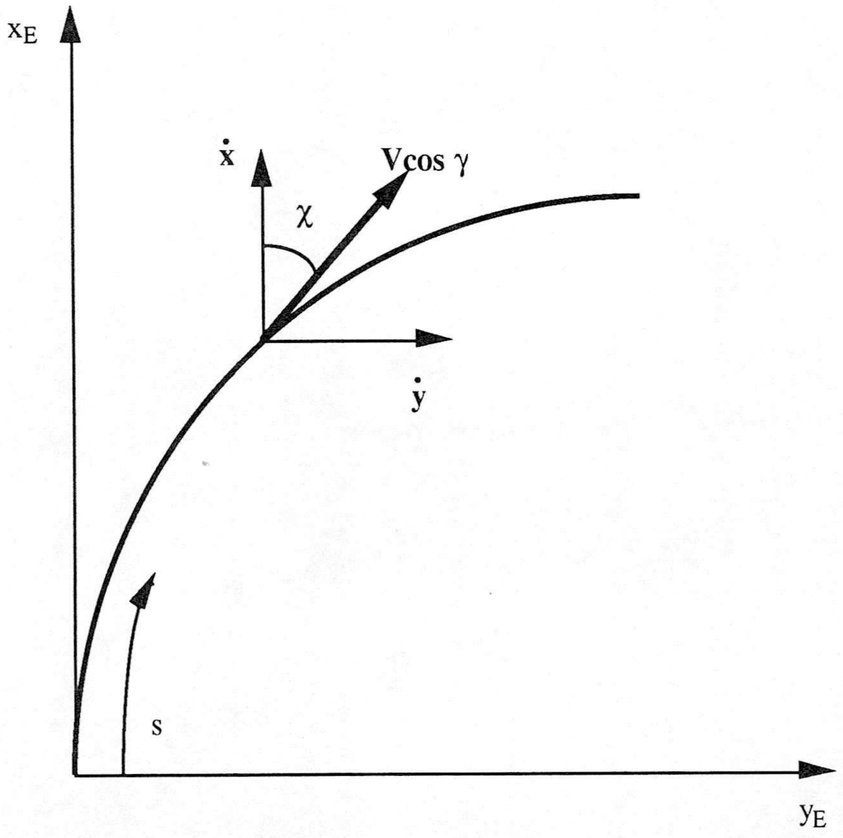


Figure D2 The Track

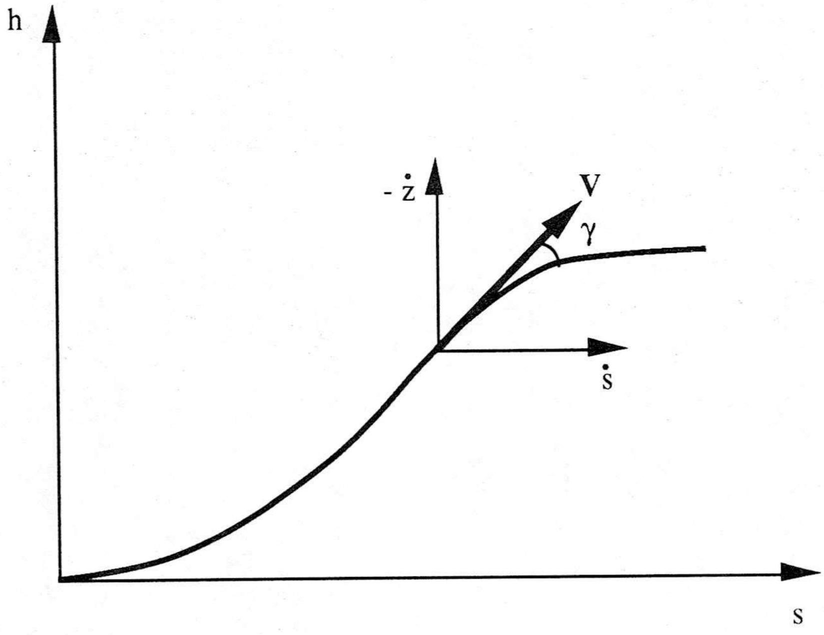


**Figure D3** <u>The Altitude</u>

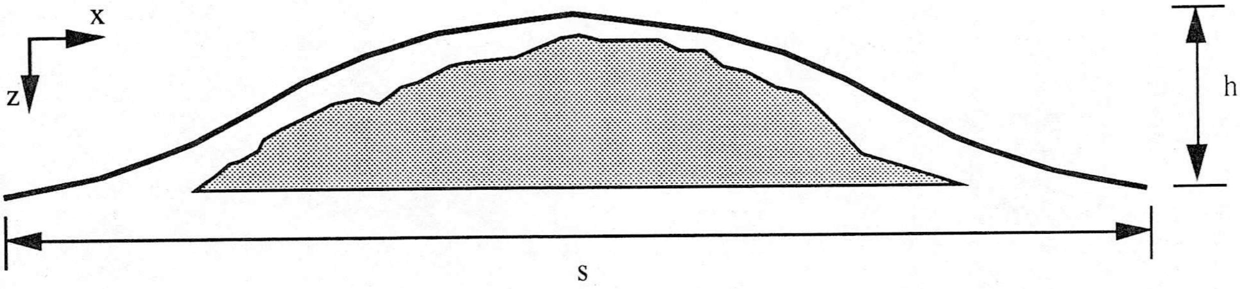


**Figure D4** <u>Hurdlehop Manoeuvre</u>

# References

1. Anon., "Aeronautical Design Standard, Handling Qualities Requirements for Military Rotorcraft.", ADS-33C, August 1989.

2. Thomson, D.G., "Evaluation of Helicopter Agility Through Inverse Solution of the Equations of Motion.", Ph.D, May 1987.

3. Thomson, D.G., "An Analytical Method of Quantifying Helicopter Agility.", Paper 45, Proceedings of the 12th European Rotorcraft Forum, Garmisch-Partenkirchen, Germany, September 1986.

4. Bradley, R., Thomson, D.G., "The Development and Potential of Inverse Simulation for the Quantative Assessment of Helicopter Handling Qualities.", Proceedings of the NASA/AHS Conference 'Piloting Vertical Flight Aircraft: Flying Qualities and Human Factors', San Fransisco, January 1993.

5. Thomson, D.G., Talbot, N., Taylor, C., Bradley, R., Ablett, R.", An Investigation of Piloting Strategies for Engine Failures During Take-Off from Offshore Platforms", Paper O5, Proceedings of 19th European Rotorcraft Forum, Cernobbio, Italy, September 1993.

6. Thomson, D.G., Bradley, R., "Development and Verification of an Algorithm for Helicopter Inverse Simulation.", Vertica, Vol. 14, No. 2, May 1990.

7. Cheney, W., Kincaid D., "Numerical Mathematics and Computing. Second Edition.", Brooks / Cole Publishing Company, 1985.

8. Thomson, D.G., "Development of a Generic Helicopter Model for Application to Inverse Simulation", Internal Report No. 9216, June 1992.

9. Thomson, D.G., Bradley, R., "Modelling and Classification of Helicopter Combat Manoeuvres.", Proceedings of ICAS Congress, Stockholm, Sweden, September 1990.

10. Thomson, D.G., Bradley, R., "Mathematical Definition of Helicopter Manoeuvres.", Internal Report No. 9225, June 1992.

11. Hess, R.A., Gao, C., Wang, S.H., "Generalized Technique for Inverse Simulation Applied to Aircraft Maneuvers.", Journal of Guidance, Vol. 14, No. 5, October 1991.

12. "Inverse Simulation of Large-Amplitude Aircraft Maeuvers.", Journal of Guidance, Control and Dynamics, Vol. 16, No. 4, July 1993.

13. "A Generalized Algorithm for Inverse Simulation Applied to Helicopter Maneuvering Flight.", Journal of the American Helicopter Society, October 1993.

14. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., "Numerical Recipes in Fortran. The Art of Scientific Computing. Second Edition.", Cambridge University Press, 1992.

15. Stroud, K.A., "Engineering Mathematics. Programmes and Problems. Second Edition.", Macmillan Press Ltd., 1982.

16. Thomson, D.G., Bradley, R., "Prediction of the Dynamic Characteristics of Helicopters in Constrained Flight.", Aeronautical Journal of the Royal Aeronautical Society, December 1990.

17. Thomson, D.G., "The Helinv Numerical Algorithm.", Internal Report No. 9408, June 1994.

18. Lin, K.C., Lu, P., Smith, M., "The Numerical Errors in Inverse Simulation.", American Institute of Aeronautics and Astronautics, 1993.