379-00
Dezember 2000

Physical Model-Building Leading to Automatic Code Generation

Moormann, D.

Kap. 5.3.3 in

Flight Control Design - Best Practices

Research and Technology Organization, RTO-TR-29

DLR Oberpfaffenhofen
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Robotik & Mechatronik
Entwurfsorientierte Regelungstechnik
D–82234 Wessling

RTO-TR-29

NORTH ATLANTIC TREATY ORGANIZATION

RTO-TR-29

RESEARCH AND TECHNOLOGY ORGANIZATION
BP 25, 7 RUE ANCELLE, F-92200 NEUILLY-SUR-SEINE CEDEX, FRANCE

RTO TECHNICAL REPORT 29

Flight Control Design - Best Practices

(Commande des avions - le meilleur savoir-faire)

This report prepared by Task Group SCI-026 on Flight Control Law Design and has been sponsored by the former Flight Vehicle Integration Panel of AGARD, and the Systems, Concepts and Integration Panel of RTO.

NORTH ATLANTIC TREATY ORGANIZATION

Published April 2000
Distribution and Availability on Back Cover

Since the pilot accesses both position and force information, the impact on handling qualities of delays resulting from the feel system dynamics can be less significant with respect to effects of delays produced by the flight control system itself [Anon., 1991; Smith et Sarrafian, 1986]. From this point of view, the modelling and the inclusion of the feel system model in the aircraft simulation model for control laws design and/or flying-qualities evaluations [Mitchell et Aponso, 1995] is an open issue (see chapter 3.5 and 3.6).

Although the modelling of reversible flight controls usually can become very complex [Weiss, et al. 1998], irreversible force-feel systems can often be adequately described by a second-order system [Gibson et Hess, 1997].

In general, the basic items to be considered for the feel system modelling are:

- Natural frequency and damping
- Static force displacement characteristics
- Nonlinearities (in force gradients, break-out, hysteresis, friction, etc.)

Analyses of phenomena such as roll ratcheting require the additional introduction of a pilot model. It is evident that for this interaction between the physical coupling of aircraft response and motion of the pilots body the modelling of the stick dynamics is essential [Johnston et McRuer, 1987; Smith et Montgomery, 1996; Höhne, 1999; Koehler, 1999].

5.3.2.11 General Remarks

The modelling of an aircraft requires a background not only related to the "flying machine" (flight mechanics, aerodynamics, control system architecture, etc.) but also to the knowledge of other subjects relevant to simulation such as:

- Computer science
- Numerical analysis
- Programming

The required accuracy and complexity of the complete aircraft model defines the degree of detail of the sub-models. It is of no use to implement highly sophisticated sub-models if the basic inputs such as the ADS are not reliable (see BP2.1).

A configuration control of the aircraft simulation model is a must, especially for shared models. If this configuration management is not automated a formally revision process should be established.

The modelling environment has been discussed and some of the advantages of modern VMOS have been highlighted. They provide a common and versatile framework to different groups (such as Stress or Aerodynamics Dept.). Also the model / sub-models benefit from the demands, the knowledge, and the contributions of all the different users. This interaction can significantly improve the quality of modelling. As pointed out in [Robins et al., 1998], some disadvantages of using VMOS still exist. For example, VMOS software is not faultless. Where a bug is present, the user is at the mercy of the vendor for prompt support and fixes. Furthermore, an automatic code generator add-on is often expensive.

The availability of a Flexible Simulation Model helps the FCL engineer throughout design and analysis. FSM is a VMOS-based detailed and compact aircraft simulation model that is extremely flexible with regard to modifications (addition / elimination / modification of its components). Simplifications, approximations and changes required for analysis and design of FBW aircraft can be chosen by the user according to the respective application.

An important issue, in relation to modelling, is the availability of specific documentation. While the modelling of elements such as rigid body equations has been widely discussed in literature, it is hard to find a detailed model of a sensor. Moreover, those who have spent their time and energy in making such models tend to be protective towards their work. Nevertheless, some documentation is available for the basic [Ralfe et Staples,1986] and more detailed model [Messina et al., 1996; Anon., 1997].

A crucial remark is to be made on the portability / compatibility of the simulation models. VMOS have both the possibility of incorporation of legacy code and the capability of automatic code generation. Therefore, it can be stated that they favour the exchange and reuse of simulation code. But a far-reaching real portability and/or compatibility does not exist yet. More efforts should be spent on standardisation, not only for the basics of the modelling data [Hildreth, 1998] but also for the simulation models themselves.

5.3.3 Physical Model-Building Leading to Automatic Code Generation

The task of the model-building and code generation process is to put all required aircraft data into a

computer readable format and make the (nonlinear) model available to synthesis and analysis tools for design and assessment of flight control systems.

The model-building and code generation process from a physically set-up flight dynamics library is described here for the High Incidence Research Model (HIRM)*, which was one of the benchmarks solved within the GARTEUR† Design Challenge on Robust Flight Control (1995-1997), where 18 teams from 7 European countries investigated the applicability of modern control concepts for developing robust flight control systems.

A unique simulation model for HIRM has been made available by this technique to all design teams either as Matlab/Simulink simulation code or as Fortran/C codes. The different codes describing the same aircraft dynamics model were built automatically from a 'generic' physical aircraft description, using the object oriented modelling and simulation code generation environment Dymola [Elmquist, 1993] This procedure guaranteed that groups working with different simulation environments still used the same aircraft model.

5.3.3.1 Object Modelling

Models of aircraft dynamics should be described in a notation close to the aircraft physics. The most natural way of modelling physical systems is as physical objects and phenomena, which are connected according to their physical energy flow interaction. This is different from modelling via signal flows or input-output block diagrams as traditionally used for controller modelling.
An aircraft consists of a variety of different systems, which represent the interacting disciplines involved in aircraft engineering (e.g. flight mechanics, aerodynamics, propulsion).

As displayed in Figure 5.3.5, an aircraft consists of a *body* (airframe), which is powered by one or more *engines* and which has *gravity* acting on it. The *aerodynamics* describes the effects of the airflow over the aircraft, which is influenced by the surrounding *atmosphere* and additional *winds*.

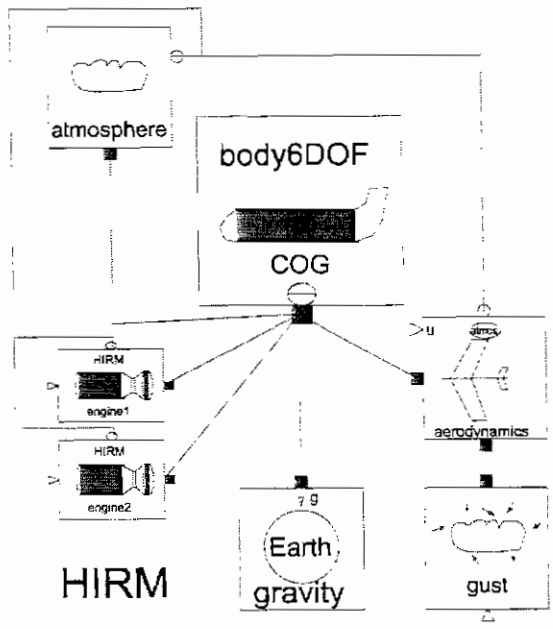

Figure 5.3.5: Object diagram of HIRM

Each of these phenomena is most conveniently described as one physical object. All objects are connected according to Figure 5.3.5 to represent the interactions within an aircraft. No connections between engine and aerodynamics have been considered for HIRM, but it should be noted that especially for fighters and for prop aircraft such effects might have strong influence and cannot be neglected.

In order to make the understanding of the objects easy, each component is described in its own coordinate system. Gravity, wind, and atmosphere are conveniently described in the aircraft-carried normal earth system, aerodynamics in the air-path axis system, and engines in a system which is related to the body axis system. Hence coordinate transformations and an object to describe the relationship between velocity, wind, and airspeed are needed in between all of these sub-systems when they are connected. Therefore, in addition to the basic aircraft components, coordinate transformations are also detailed and handled as objects in the aircraft library (see Figure 5.3.6).

* HIRM data provided from DERA, Bedford, UK

† GARTEUR = Group of Aeronautical Research and Technology in EURope

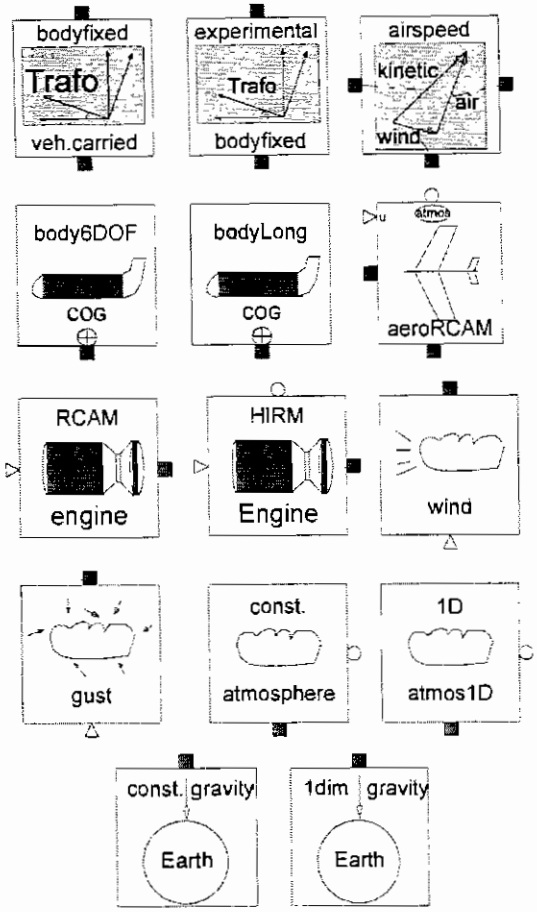

Figure 5.3.6: Aircraft model library

In the physical aircraft library different representations of one component can be found. There is a class Body with six degrees of freedom (Body6DOF) and a class with three degrees of freedom (BodyLong), which can be used to generate a nonlinear simulation model for the longitudinal axis only. There are also engine, atmospheric and gravity models of different complexity.

In a graphical view Figure 5.3.5, the interconnection structure of an aircraft can be most easily understood. If a more complex gravity model acts on the aircraft, this object can simply be taken from the aircraft library to replace the simple gravity object. In the same way one or more engines can be added or removed from the aircraft or can be modified. This is the most transparent user layer with no need to think about the structure of any specific simulation code.

The objects which form the physical model contain equations (and not assignments as common in programming or simulation languages). This makes the understanding and the reuse much easier than looking at low level code, whose purpose is to be understood by a computer. Once the objects are available in computer readable form the object equations can be sorted automatically by a symbolic equation handler. This is a main feature of Dymola.

Objects, formulated in that way do not necessarily have to represent causalities. This allows one object to fulfil different tasks. For example, the object which does the transformations between the body axis and the air-path axis system, is used for the transformation of the body-fixed velocities to the air-path axis system, as they are required within the aerodynamics. The same object is used for the transformation of the forces and moments from the aerodynamic to the body axis system. When connecting components as objects, only the relation between them is defined and not the order, in which those equations are finally solved.

In *Dymola*, graphical syntax components are coupled by drawing a line between the defined 'coupling' points of the objects, which are called 'cuts'. These couplings represent either energy or signal flow. For example, the cut *bsystem* (body axis system) has the following structure:

terminal
$^{v}T^{b}[3,3], r[3], v_b[3], w_b[3], a_b[3], z_b[3], F_b[3], M_b[3]$

cutbsystem $\left(^{v}T^{b}, r, v_b, w_b, a_b, z_b / F_b, M_b\right)$

The matrix $^{v}T^{b}$ defines the orientation of the body axis system with respect to the aircraft-carried normal earth system. The vector r is the aircraft's inertial position in the aircraft-carried normal earth system; the vectors v_b and a_b are the velocity and acceleration in the body axis system and the vectors F_b and M_b are the forces and moments, also formulated in the body axis system. In the same way there are cuts defined for the aircraft-carried normal earth system (*vsystem*) and for the air-path axis system (*asystem*).

This cut structure represents physical connections. When objects are connected, *Dymola* adds equations for the cut variables. All quantities of the cut before the slash operator (Across variables) are set equal when connected, as it is reasonable for positions, velocities and accelerations, quantities after the slash operator (Through variables) are summed up to zero, as it is reasonable for forces and moments. This principle is used for connecting engines to the aircraft body for example. The engines have the same position, airspeed, and accelerations than the aircraft's body, their forces and moments sum up with all the other forces and moments acting on the

aircraft. Because of that formulation it is easy to add more engines to the aircraft just by adding another engine object to Figure 5.3.5 and connecting it to the aircraft's body.

This object-oriented equation-based form of describing physical systems helps to understand the physical system and enables the user to modify the model most conveniently.

5.3.3.2 Hierarchical Object Structure

An important aspect in object oriented modelling of physical systems is the encapsulation of objects. The internal implementation of details, e.g. of the aerodynamics, are not visible, when viewing the HIRM object model as depicted in Figure 5.3.5. By encapsulation, the implementation of an object can be changed without affecting the functionality of the whole model.

Figure 5.3.7 demonstrates, how the HIRM model is structured. Here only the aerodynamics model is extracted. In the same way details of the engine, gravity, wind, and atmospheric models can be displayed.

Extracting the *aerodynamics* results in the aerodynamics sub-model, which consists of the aerodynamic equations *aero equations* and the object *airspeed*. The latter object describes the kinematics between the inertial motion (flight-path velocity), the wind velocity, and the aircraft's movement relative to the air (aircraft velocity).

Using the graphical interface, 'double clicking' on *aerodynamics* displays the parameter window of this object. This window allows the parameters to be modified. In the same way, all of the other objects (body, engines) can be instantiated with their parameters.

The objects of HIRM model will be detailed in the following sections. Boxes in the following sub-sections contain Dymola code in the form of real equations.

5.3.3.3 Code Generation

From the graphical and textual model description *Dymola* generates efficient code for different simulation environments (see Figure .5.3.8).

Its symbolic equation handler generates a state space model from the parameter instantiated equations of each object and from the equations derived from the interconnection structure. The equations are sorted and solved according to the specified inputs and outputs. Equations which are formulated in an object but not needed for the specified configuration are removed automatically. The result is a mathematical model with a minimum number of equations for the specified task.

As a next step, simulation code for different simulation environments (e.g. Simulink, ACSL, ANDECS_DSSIM) is generated automatically. The code for Simulink can be a m-file or a cmex-file. Fortran or C code can be exported in the DSblock neutral simulation-model format [Otter, 1992], to be used in any other simulation run-time environment capable of importing Fortran or C models. This is targeted in particular at the ANDECS design environment for control engineering [Grübel, et al, 1993].

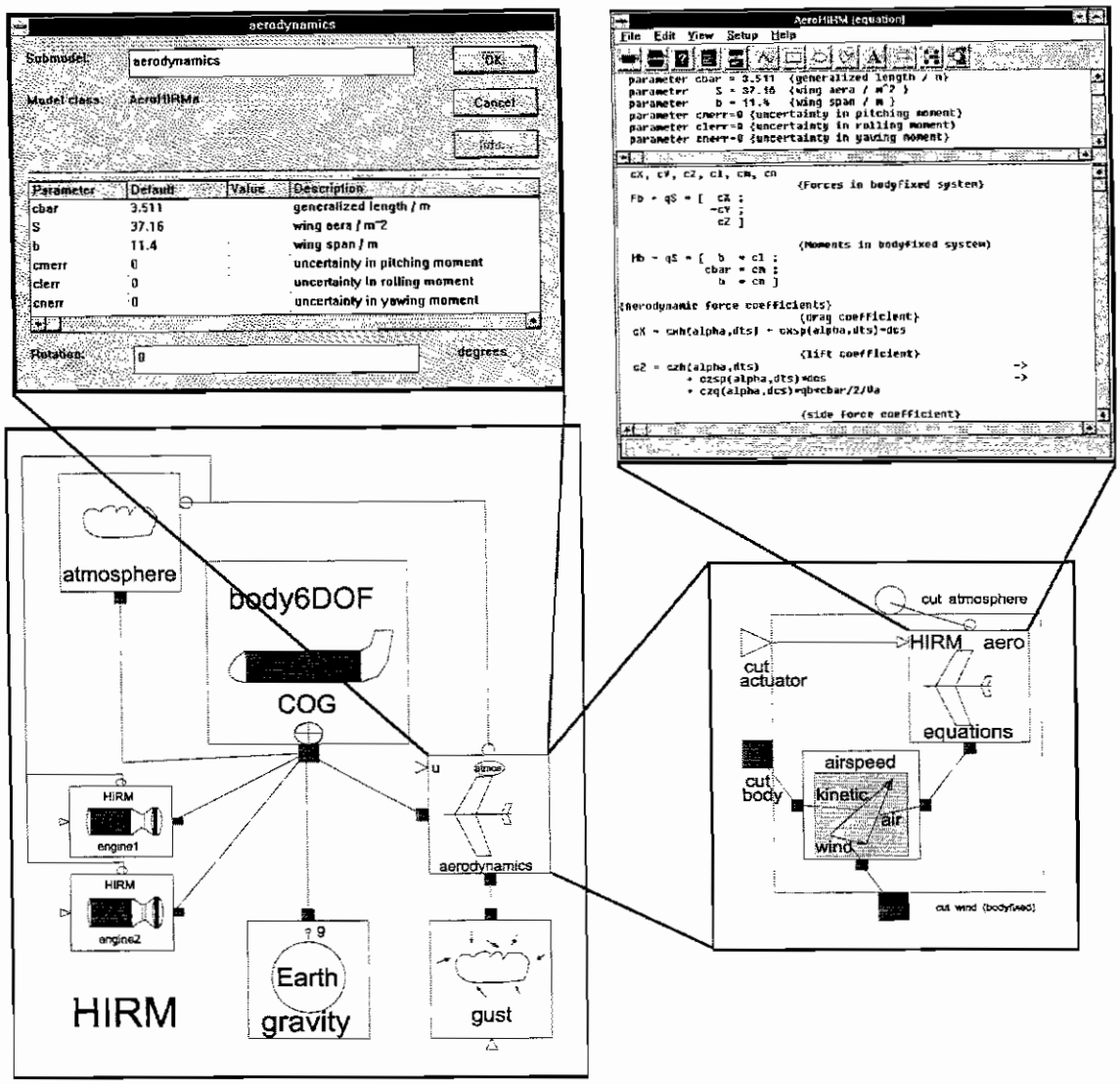

Figure 5.3.7: Structure of aircraft physical model

5.3.3.4 General Remarks

It is most natural to model physical systems on a physical level in the form of equations. For simulation purposes, simulation code can be generated automatically from physical equations provided that a suitable software tool like Dymola is available. It has been shown that aircraft dynamics code for Simulink is generated for common use in the Robust Control Design Challenge. This is achieved by using a generic *Dymola* flight dynamics object library.

This approach to automatic code generation has the further advantage that not only can efficient parameterised simulation code be obtained for different simulation and analysis environments but a parameterised symbolic code can be produced as well. This can be used as input for symbolic analysis tools such as PUM (Matlab Toolbox for Parametric Uncertainty Modelling) [Lambrechts, Terlouw, 1992] or PARADISE (PArametric Robustness Analysis and Design Interactive Software Environment) [Sienel, Ackermann, 1996].

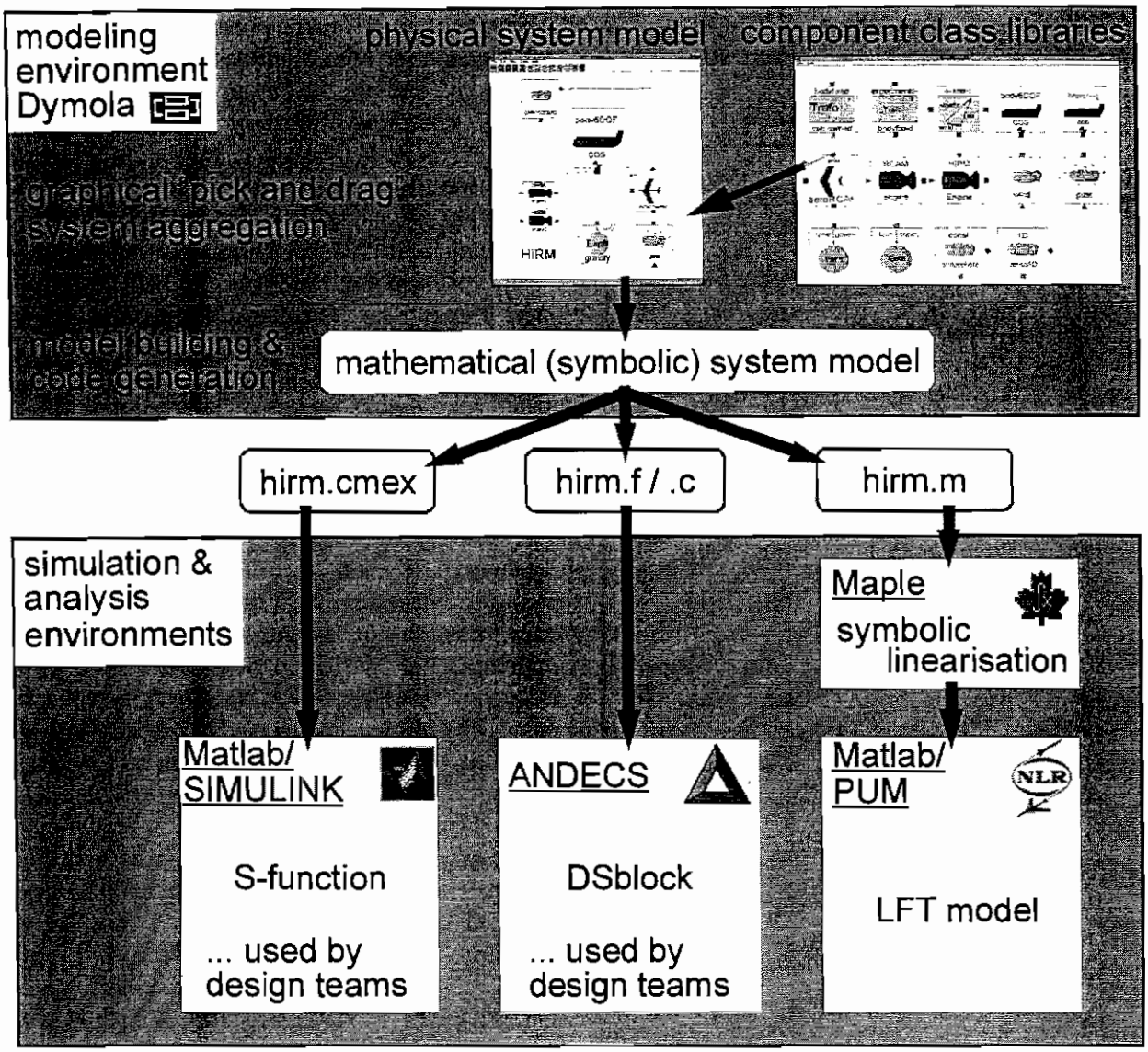

Figure 5.3.8: From system configuration to simulation/analysis model

5.3.4 System Identification and Model Validation

System identification is an inverse problem of obtaining model description in some suitable form for a system, given its behavior as a set of observations. The highly successful application of system identification to flight vehicle is possible partly due to the advances in measurement techniques and data processing capabilities provided by digital computers, partly due to the ingenuity of the engineers in advantageously using the developments in other fields like estimation and control theory, and partly due to the fairly well-understood basic physical principles underlying flight vehicles enabling adequate modeling and the possibility of carrying out proper flight tests.

5.3.4.1 Principles of System Identification

The general approach to aircraft system identification is shown in Figure 5.3.9. During the flight tests, specifically designed control inputs are applied to excite the characteristic aircraft motions. The applied control inputs and aircraft responses are measured and recorded. A suitable model is postulated for the phenomenon being investigated and the unknown parameters within the model are so determined as to match the model response with the flight measured aircraft response.

A coordinated approach to flight vehicle system identification can be divided into three major parts [Hamel; Jategaonkar, 1996 and 1998]: