C12

BUILDING A FLIGHT MECHANICS LIBRARY USING ECOSIMPRO

Borja García Gutiérrez and Ramón Pérez Vara, Empresarios Agrupados
Magallanes, 3 28015 Madrid - Spain
bgg@ecosimpro.com - rpv@ecosimpro.com

Abstract

A components library has been developed in EcosimPro to build modular flight mechanics models for aeroplanes and other aircraft. The library facilitates calculation of the actuations and dynamic response of aircraft taking into account possible automatic pilots. In order to verify the library components, an F-16 model and two automatic pilot models have been built which were used to calculate the actuations of the aircraft and its dynamic responses with and without an automatic pilot.

Key Words: Simulation, EcosimPro, object-oriented physical system modelling, flight mechanics, control and automatic pilot systems.

1 INTRODUCTION

The purpose of this work is to develop an EcosimPro components library which will facilitate the construction of modular models for calculating the actuations of aeroplanes and other aircraft and their dynamic response to manoeuvres.

But before going into details about creating a library with these characteristics in EcosimPro, we will first give a brief description of the model that is going to be built.

The first step is to establish the hypotheses in the physical-mathematical aircraft model to be simulated, which will give us the general equations of movement. A solid, rigid aircraft is considered, on which only gravitational, aerodynamic and propulsive forces will act.

With these hypotheses established, and assuming that the initial conditions and an expression of the intervening forces have been built into the model, we can introduce the equations which will determine system evolution.

We must now select some axes on which to work, so the second step is to describe the two systems used to represent the attitude of the aircraft. The first system, denominated Body Axes, has its origin (O_b) at the centre of the aircraft mass, axis O_bX on the symmetry plane in the direction of forward motion, axis O_bZ also on the symmetry plane but with the downward motion of normal flight attitude, and axis O_bY forming a right-handed frame. The second system, denominated Earth Axes, has its origin (O_e) at a point on the Earth's surface, axis O_eN on the horizontal plane directed north, axis O_eE also on the horizontal plane but directed towards the east, and axis O_eD perpendicular to the horizontal plane and directed downwards. The relative direction of these axes can be seen in Figure 1.

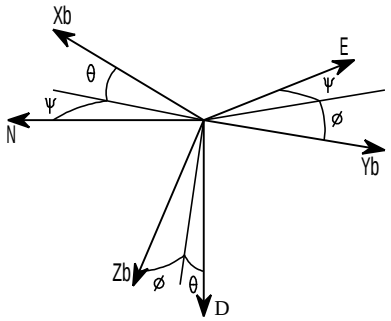

Figure 1: Relative direction between Body Axes and Earth Axes

As we will see later, this relative position will determine the attitude of the aircraft with respect to earth through the angles *?* (yaw), *?* (trim) and *j* (roll).

The third step is to introduce a hypothesis in the atmosphere model to calculate the air properties based on flight height. The international standard atmosphere (ISA) will be used.

The last step is to determine the scope of the control and automatic pilot models. They all respond to typical designs Control Augmentation System (CAS). These systems –which will send a signal to the control surfaces of the aircraft- have been modelled as first-rate delayed action actuators with limited deflection and deflection velocity.

2 PHYSICAL-MATHEMATICAL MODEL OF THE AIRCRAFT

As mentioned previously, our aircraft will be considered as a rigid solid with six degrees of freedom (although for several reasons not discussed in this paper they will be reduced to four and referred to as control parameters of the aircraft). The six equations representing the movement of a rigid solid can therefore be contemplated in the case of being subjected to the three types of forces described earlier. Since this will be developed on non-inertial Body Axes, the inertia terms will also have to be introduced into the calculations.

$$m \cdot \left(\dot{\vec{V}} + ?_{be} \wedge \vec{V} \right) = \vec{F}_G + \vec{F}_A + \vec{F}_T \qquad (1)$$

$$\bar{\bar{I}} \cdot \dot{?}_{be} + ?_{be} \wedge \left(\bar{\bar{I}} \cdot ?_{be} \right) = \vec{M}_G + \vec{M}_{?} + \vec{M}_{?} \qquad (2)$$

In addition, based on the linear and angular velocities, the equations necessary to obtain the position and attitude of the aircraft with respect to Earth will have to be included in the model. To do this it will be necessary to provide the kinematic ratios that determine the attitude as a function of the angular velocity and the reference system change matrix which will enable us to calculate the velocity of the aircraft on Earth Axes based on the attitude.

2.1 GRAVITATIONAL FORCES

For the gravitational forces in our model, it will be assumed that gravity is a constant which is independent of the aircraft. This means working n the hypothesis that the Earth is flat.

Taking into account the above consideration, gravitational force will only have to be projected on Body Axes, known to be a force applied in the direction of the O_eD axis and a product of the mass of the aircraft (m) multiplied by the acceleration of gravity (g) is an absolute value, such that

$$\begin{aligned} F_{xG} &= -m \cdot g \cdot \text{sen}\, ? \\ F_{yG} &= m \cdot g \cdot \text{sen}\, f \cdot \cos ? \\ F_{zG} &= m \cdot g \cdot \cos f \cdot \cos ? \end{aligned} \qquad (3)$$

If we also fix the centre of masses as the source for calculating the moments on the aircraft, because of the gravitational field there will be no moment at all on it and, therefore

$$\vec{M}_G = \vec{0} \qquad (4)$$

2.2 AERODYNAMIC FORCES

For calculating the components of the aerodynamic force and moment, we use a series of tables giving each one of the aerodynamic coefficients as a function of the conditions of flight and of the angles of control surface deflection. These are expressed as follows:

$$\begin{aligned} F_{xA} &= q \cdot S \cdot C_{xT} \\ F_{yA} &= q \cdot S \cdot C_{yT} \\ F_{zA} &= q \cdot S \cdot C_{zT} \end{aligned} \qquad (5)$$

$$\begin{aligned} M_{xA} &= q \cdot S \cdot b \cdot C_{lT} \\ M_{yA} &= q \cdot S \cdot c \cdot C_{mT} \\ M_{zA} &= q \cdot S \cdot b \cdot C_{nT} \end{aligned} \qquad (6)$$

where q is the dynamic pressure, S is the wing surface, b is the wing span, c the average wing chord and the force and moment coefficients are obtained with the following formulae:

$$\begin{aligned} C_{xT} &= C_{xT}(a, q, de) \\ C_{yT} &= C_{yT}(a, ß, p, r, da, dr) \\ C_{zT} &= C_{zT}(a, ß, q, de) \end{aligned} \qquad (7)$$

$$\begin{aligned} C_{lT} &= C_{lT}(a, ß, p, r, da, dr) \\ C_{mT} &= C_{mT}(a, p, de, Xr, Xcg) \\ C_{nT} &= C_{nT}(a, ß, p, r, da, dr, Xr, Xcg) \end{aligned} \qquad (8)$$

where a is the angle of attack, b is the sideslip angle, p, q and r are the components of the angular velocity of the aircraft on Body Axes, d_e, d_a and d_r are the control surface deflections (elevator, ailerons and rudder, respectively) and $X_R$andX_{CG} are the nominal and real positions of the centre of masses adimensionalised with the wing chord.

The model has some validity intervals for variables which are limited by the content of values in the tables, so it should therefore be noted that some moments will be subject to the following restrictions

$$\begin{aligned} a &\in [-15°, 45°] \\ ß &\in [-30°, 30°] \\ de &\in [-25°, 25°] \\ da &\in [-21'5°, 21'5°] \\ dr &\in [-21°, 21°] \end{aligned} \qquad (9)$$

2.3 PROPULSIVE FORCES

As in the case of aerodynamic force, there is a series of tables for the case of propulsive force from which the thrust can be determined on the basis of the flight conditions and the engine control parameter. As far as all the calculations are concerned, it is assumed that the direction of thrust coincides with the O_bX axis, and that therefore

$$\begin{aligned} F_{xT} &= F_{xT}\left(M_\infty, h, p\right) \\ F_{yT} &= 0 \\ F_{zT} &= 0 \end{aligned} \qquad (10)$$

where $M_¥$ is the flight Mach of the aircraft, h is the height of flight and p is the engine control parameter.

According to the hypotheses made, the thrust does not generate any moments with respect to the centre of masses of the aircraft, although a certain constant angular moment (H) due to engine rotation will be taken into consideration .

$$\begin{aligned} M_{xT} &= 0 \\ M_{yT} &= -r \cdot H \\ M_{zT} &= q \cdot H \end{aligned} \qquad (11)$$

3 PHYSICAL-MATHEMATICAL MODEL OF THE ATMOSPHERE

The atmosphere model to be used is that known as the ISA (International Standard Atmosphere) model for which the following hypotheses are made: air is an ideal gas, the sea level conditions are (T_0=288K and p_0=101.325Pa) and we know that the temperature profile is a continuous function of height and also, therefore, of pressure and density. From these considerations we get the values for the temperature (T), the pressure (p) and the density (ρ) of the air and the speed of sound (a) at any height

$$? = \frac{p}{R \cdot T} \qquad (12)$$

$$T = \begin{cases} T_0 + a_0 \cdot h \\ T_{11} \\ T_{11} + a_{11} \cdot h \end{cases} \qquad (13)$$

$$p = \begin{cases} p_0 \cdot \left[1 + \left(\frac{a_0}{T_0} \cdot h\right)^{\frac{-g}{R \cdot a_0}}\right] \\ p_{11} \cdot \exp\left[\frac{-g}{R \cdot T_0} \cdot \left(h - 11.000\right)\right] \\ p_{11} \cdot \left\{1 + \left[\frac{a_{11}}{T_{11}} \cdot \left(h - 20.000\right)\right]^{\frac{-g}{R \cdot a_{11}}}\right\} \end{cases} \qquad (14)$$

$$a = \sqrt{? \cdot R \cdot T} \qquad (15)$$

where the subscripts *0* and *11* represent the values of the variable at sea level and 11,000 m, a is the temperature gradient with height, R is the air gas constant, and the expressions for temperature and pressure are given for the following three intervals: from sea level up to 11,000 m, from 11,000 m up to 20,000 m and for values greater than 20,000 m.

4 PHYSICAL-MATHEMATICAL MODEL OF THE CONTROL AND AUTOPILOT SYSTEMS

With a view to simulating some manoeuvres which may be of interest, we are going to model two complementary autopilots; one longitudinal autopilot that acts on the movements on the plane of symmetry of the aircraft (control of the angular velocity of pitch) and one lateral-directional autopilot that acts on movements outside the plane of (control of the angular velocity of roll and yaw).

We will see a detailed sketch of these systems when the modelling for EcosimPro has been done. For the time being, it is limited to a representation that helps us to understand which ones are system input variables and which ones are output variables. See Figures 2 and 3.

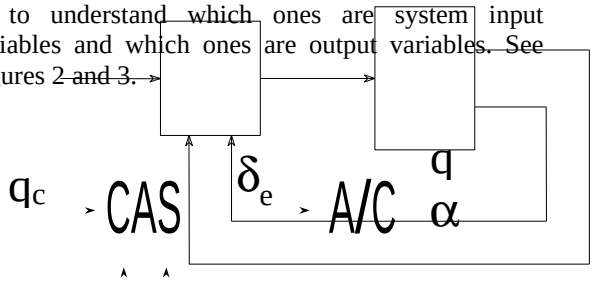

Figure 2: Longitudinal autopilot

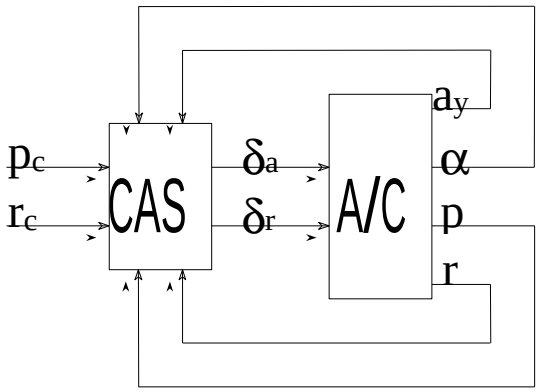

Figure 3: Lateral-directional autopilot

where *A/C* represents the aircraft, *CAS* represents the control or autopilot system and p_c, q_c and r_c represents the control or controlled variables.

5 PHYSICAL-MATHEMATICAL MODEL OF THE ACTUATORS

A simple response model (time lag of the first order) has been selected for the actuators and certain limits have also been taken into consideration for the angle of deflection and for the velocity of deflection, such that

$$\begin{aligned} d' &= \frac{1}{t}\cdot(dc - d) \\ d &\in [d\,min, d\,max] \\ d' &\in [d'\,min, d'\,max] \end{aligned} \qquad (16)$$

where *dc* and *d* are the pilot-controlled and real angles of deflection of the control surface, and *t* is the characteristic response time of the actuator.

6 GENERAL ASPECTS OF THE FLIGHT_SIM LIBRARY

Under this heading we will discuss the elements that have to be dealt with before we can begin to develop a library of this kind. In other words, the definition of the coordinates of the different reference axes, constants, ports and functions not directly linked to the model.

6.1 COMMON ELEMENTS

All these elements are included in the file FLIGHT_common.el, which is usual practise in the complete EcosimPro library.

To be able to handle vectorial variables, two sets of coordinates are defined, one set for Body Axes (x,y,z) and another set for Earth Axes (N,E,D).

```
ENUM MovilAxis = {x, y, z}
ENUM EarthAxis = {N, E, D}
```

Of the constants that have been defined, it is worth noting the acceleration of gravity (g), the air gas constant (R) and the ratio of air specific heat (γ).

```
CONST REAL g = 9.806        "gravity acceleration (m/s**2)"
CONST REAL R = 287.         "air gas constant (J/kg*K)"
CONST REAL gamma = 1.4      "ratio of air specific heat ()")
```

Lastly, to run the values of the forces on the model and to process the variables for the state of the system, two ports have been defined: *Forces* and *State*, respectively. The first runs two vectorial *SUM* variables corresponding to the force and to the moment on the Body Axes, and the second runs thirteen *EQUAL* variables which describe the complete evolution of the system: the position with respect to Earth, the velocity and the acceleration on Body Axes and on Earth Axes, and the attitude with respect to Earth and with respect to the current incident.

```
PORT Forces
  SUM REAL F[MovilAxis]          "external forces (N)"
  SUM REAL M[MovilAxis]          "exteranl moments (N*m)"
END PORT

PORT State
  EQUAL REAL alpha               "angle of attack (rad)"
  EQUAL REAL beta                "sideslip angle (rad)"
  EQUAL REAL VT                  "aircraft velocity (m/s)"
  EQUAL REAL V[MovilAxis]        "body axis velocity (m/s)"
  EQUAL REAL Vdot[MovilAxis]     "body axis accel. (m/s**2)"
  EQUAL REAL W[MovilAxis]        "body axis angular velocity
                                   (rad/s)"
  EQUAL REAL Wdot[MovilAxis]     "body axis angular
                                   accel. (rad/s**2)"
  EQUAL REAL A[MovilAxis]        "body axis absolute
                                   acceleration (m/s**2)"
  EQUAL REAL psi                 "yaw angle (rad)"
  EQUAL REAL theta               "pitch angle (rad)"
  EQUAL REAL phi                 "roll angle (rad)"
  EQUAL REAL R[EarthAxis]        "earth axis position (m)"
  EQUAL REAL Rdot[EarthAxis]     "earth axis velocity (m/s)"
END PORT
```

All these gates are defined and used in such a way that they run variables with units contained in the international system, as indicated in the comments in the program.

6.2 ATMOSPHERE MODEL

All the necessary elements are included in the file FLIGHT_f_Atm.el.

The model comprises all the functions necessary to evaluate the temperature, the pressure and the density of the air and the speed of sound as a function of the height.

In short, they are functions with a single input argument, ie the height of flight, that return the value of the property we want to calculate. The names that have been given to these functions are *TAtm*, *pAtm*, *rhoAtm* and *VsoundAtm*. The following is an example of these functions:

```
FUNCTION REAL rhoAtm (IN REAL altitude)
  DECLS
    REAL T          "temperature (K)"
    REAL p          "pressure (Pa)"
  BODY
    T = TAtm(altitude)
    p = pAtm(altitude)
  RETURN p / ( R * T)
END FUNCTION
```

7 MAIN COMPONENTS OF THE FLIGHT_SIM LIBRARY

These are the components which are included in all the characteristics that define our aircraft and which will give rise to the final F-16 model.

Before going any further, it should be noted that these components are generic and they can therefore be reused in any model that we may want to try to simulate, but there are others which are peculiar to the configuration of the F-16 and should be implemented for each case, even if they serve as an example of application.

7.1 "Frame" COMPONENT

This is the basic component of the library. It contains all the equations of the movement of a rigid solid as well as those needed to calculate the different state variables characteristic of the aircraft's movements and it is defined in the file FLIGHT_Frame.el.

This component is defined with one IN port *Forces* and one OUT port *State*. The values of the different forces that act on the solid are input via the IN port and the state of the solid is output via the OUT port.

```
COMPONENT Frame
  PORTS
    IN Forces forces_in     "external forces input"
    OUT State state_out     "aircraft state output"
```

Another noteworthy feature of this component is its data. With these data we can change the inertia properties of our aircraft (such as the mass and the moments of inertia) which makes it completely reusable.

```
  DATA
    REAL mass = 10000.   "mass (kg)"
    REAL Ixx  = 10000.   "moment of inertia Ix (kg*m**2)"
    REAL Iyy  = 100000.  "moment of inertia Iy (kg*m**2)"
    REAL Izz  = 100000.  "moment of inertia Iz (kg*m**2)"
    REAL Ixy  = 0.       "product of inertia Pxy (kg*m**2)"
    REAL Ixz  = 1000.    "product of inertia Pxz (kg*m**2)"
    REAL Iyz  = 0.       "product of inertia Pyz (kg*m**2)"
```

7.2 "F16Aerodynamics" COMPONENT

This is the component responsible for obtaining the aerodynamic forces that act on the F-16 model, which makes it a specific component, and it is defined in the file FLIGHT_F16Aerodynamics.el.

It contains all the equations necessary to calculate the forces based on the coefficients of forces and moments, as they were defined in the section which addresses the physical-mathematical model.

This component is defined with four IN ports, ie one *State* and three *analog_signal*, and two OUT ports, ie one *Forces* and the other *analog_signal*. The variables of the aeroplane's state and the deflections of the three control surfaces are input via the IN ports and the forces and the flight Mach are output via the OUT ports.

Given that all the parameters needed to obtain the value of the coefficients are determined by the aircraft's state, with the exception of the control surface deflections, this component will have three degrees of freedom: *de*, *da* and *dr*, which will fix three of the four control parameters of the aircraft.

As in the case of the *Frame* component, it is worth pointing out that its data include, in addition to the tables to calculate the coefficients, the values of the chord, the span and the surface of the wing, as well as two adimensional magnitudes which determine the nominal position of the centre of masses (X_R) and the real position (X_{CG}), the values of which will considerably condition the stability of the aircraft.

7.3 "F16Engine" COMPONENT

This component is responsible for obtaining the propulsive forces that act on the F-16 model, which means it is also a specific component, and it is defined in the file FLIGHT_F16Engine.el.

It contains all the equations necessary to obtain the forces based on the levels of power demanded (including the equations that calculate the characteristic time lags) and the flight conditions, as they were defined in the section which addresses the physical-mathematical model.

This component is defined with three IN ports, ie one *State* and two *analog_signal*, and one OUT port, ie *Forces*. The variables of the aircraft's state, the engine control parameter and the flight Mach are input via the IN port and the forces are output via the OUT port.

Likewise, given that all the parameters necessary to obtain the forces are determined by the aircraft's state, with the exception of the engine control parameter, this component will have only one degree of freedom: *p*, that which will fix the fourth of the aircraft's control parameters.

We will now go into a little more detail about the data of this component, as we have done with the previous components, which in this case include three throttle

tables for three throttle conditions (idle, military and maximum) as a function of flight height and Mach and the value of the engine angular moment.

7.4 "Trimmer" COMPONENT

With this component we can obtain the state of the aircraft for certain stationary flight conditions. Since the complete system has four degrees of freedom, it will be sufficient to impose four independent conditions, such as the flight coordinate (nil lateral acceleration) and the flight velocity coordinates, the constant climb angle and horizontal turn rate. The *Trimmer* is a generic component and valid, therefore, for any model. It is defined in the file FLIGHT_Trimmer.el.

This is the most exceptional component in this library, a statement which is substantiated by the need to calculate certain initial stationary conditions on which we begin to simulate any manoeuvre. The fact of the matter is that, because this is such a complex, unstable system, the determination of these initial conditions is no trivial task and it this component which is required to impose them.

The component contains the four equations necessary to impose the stationary flight conditions or to keep the aircraft control parameters constant and equal to those obtained when the initial conditions were calculated, according to the value taken by the overall *Trim* variable. This variable is assigned the value of 1 when a stationary condition is to be calculated and the value of 0 when the system is to be integrated. The influence that this variable has on the equations in the "*Stationary Conditions*" block can be seen in the code.

This component is defined with one IN port, ie *State* and four OUT ports, ie *analog_signal*. The variables of the aircraft's state are input via the IN port and the values of the four control parameters for the imposed stationary conditions are output via the OUT port.

In this component, it is through the three data which are defined as: *VT_req* (required aircraft velocity), *ClimbAngle_req* (required climb angle) and *dpsi_req* (required turn rate), that the initial flight conditions are controlled. This makes it possible for these values to be modified when it is required to define other components or each time it is required in an experiment.

```
COMPONENT Trimmer
  PORTS
    IN State state_in
    OUT analog_signal s_elevator
    OUT analog_signal s_aileron
    OUT analog_signal s_rudder
    OUT analog_signal s_throttle
  DATA
    REAL VT_req = 100.          "required aircraft velocity (m/s)"
    REAL ClimbAngle_req = 0.    "required climb angle (rad)"
    REAL dpsi_req  = 0.         "required turn rate (rad/s)"
  DECLS
    REAL alpha                  "angle of attack (rad)"
    REAL beta                   "sideslip angle (rad)"
    REAL VT                     "aircraft velocity (m/s)"
    REAL V[MovilAxis]           "body axis aircraft velocity (m/s)"
    REAL W[MovilAxis]           "body axis angular velocity (m/s)"
    REAL psi                    "yaw angle (rad)"
    REAL theta                  "pitch angle (rad)"
    REAL phi                    "roll angle (rad)"
    REAL ClimbAngle             "climb angle (rad)"
    REAL dpsi                   "derivative of the yaw angle (rad/s)"
    REAL elevator               "deflection of the elevator (rad)"
    REAL aileron                "deflection of the ailerons (rad)"
    REAL rudder                 "deflection of the rudder (rad)"
    REAL throttle               "throttle position [0., 1.] ()"
  CONTINUOUS
    alpha = state_in.alpha
    beta = state_in.beta
    VT = state_in.VT
    V = state_in.V
    W = state_in.W
    psi = state_in.psi
    theta = state_in.theta
    phi = state_in.phi
      -- Climb angle for coordinated flight
    sin(ClimbAngle) = cos(alpha) * cos(beta) * sin(theta) - \
      ( sin(beta) * sin(phi) + sin(alpha) * cos(beta) * cos(phi) ) * \
      cos(theta)
      -- Derivative of the yaw angle
    dpsi = ( W[y] * sin(phi) + W[z] * cos(phi) ) / cos(theta)

      -- Stationary Conditions
    elevator' + Trim * ( ClimbAngle_req - ClimbAngle ) = 0.
    aileron' + Trim * ( dpsi_req - dpsi ) = 0.
    rudder' + Trim * ( W[x] * V[z] - W[z] * V[x] + \
      g * sin(phi) * cos(theta) ) = 0.
    throttle' + Trim * ( VT_req - VT ) = 0.

      -- Outputs Assignation
    s_elevator.signal = elevator
    s_aileron.signal = aileron
    s_rudder.signal = rudder
    s_throttle.signal = throttle
END COMPONENT
```

7.5 "Actuator" COMPONENT

This component is implemented to simulate the behaviour of the control surfaces and therefore contains all the equations necessary to model the time lag of the first order and to establish to limits of deflection and the velocity of deflection. It is defined in the file FLIGHT_actuators.el.

This component is defined with one IN port and one OUT port, both *analog_signal*. The values of the required deflections are input via the IN port and the real value of the deflection, including the limitations of the actuator, are output via the OUT port.

Once again, the data are a noteworthy feature of the component because it is through these that the opposite of the characteristic response time of the actuator (*Rtau*), the deflection limit (*DL*) deflection rate limit (*RL*) are established.

8 CONSTRUCTION OF THE "F16" COMPONENT FROM THE FLIGHT_SIM LIBRARY

We are now in a position to connect all the components defined up till now (*Frame*, *F16Aerodynamics*, *F16Engine*, *Trimmer* and *Actuator*). These components will result in the final F-16 model and they are defined in the file FLIGHT_F16.el.

This component is defined with four *analog_signal* IN ports and one *State* OUT port. The values of the control variables are input via the IN port and the aircraft's state is output via the OUT port.

In addition, the code must include all the components that are going to be used from this library or from other libraries.

The moment that this component is implemented is when the values of the model have to be assigned to the generic component data from the library in order to make them particular to the F-16 model, ie, those of the components *Frame*, *Trimmer* and *Actuator*. How this is achieved can be seen in the "*TOPOLOGY*" block of the code.

```
COMPONENT F16
  PORTS
    IN analog_signal s_elevator
    IN analog_signal s_aileron
    IN analog_signal s_rudder
    IN analog_signal s_throttle
    OUT State state_out

  TOPOLOGY
    Frame Frame(
      mass = 9071.8,
      Ixx = 28384.,
      Iyy = 166832.,
      Izz = 188610.,
      Ixy = 0.,
      Ixz = 2935.,
      Iyz = 0.)

    F16Aerodynamics Aero

    F16Engine Engine

    Actuator El_Actuator(
      Rtau = 20.2,
      RL = 60.,
      DL = 25.)

    Actuator Ail_Actuator(
      Rtau = 20.2,
      RL = 80.,
      DL = 21.5)

    Actuator Rdr_Actuator(
      Rtau = 20.2,
      RL = 120.,
      DL = 30.)

    Trimmer Trimmer(
      VT_req = 100.,
      ClimbAngle_req = 0.,
      dpsi_req = 0.)

    S_summing Sum1

    S_summing Sum2

    S_summing Sum3

    S_summing Sum4

    -- Components connections
  CONNECT s_elevator TO Sum1.s_in_1
  CONNECT s_aileron TO Sum2.s_in_1
  CONNECT s_rudder TO Sum3.s_in_1
  CONNECT s_throttle TO Sum4.s_in_1

  CONNECT Trimmer.s_elevator TO Sum1.s_in_2
  CONNECT Trimmer.s_aileron TO Sum2.s_in_2
  CONNECT Trimmer.s_rudder TO Sum3.s_in_2
  CONNECT Trimmer.s_throttle TO Sum4.s_in_2

  CONNECT Sum1.s_out TO El_Actuator.s_in
  CONNECT Sum2.s_out TO Ail_Actuator.s_in
  CONNECT Sum3.s_out TO Rdr_Actuator.s_in
  CONNECT Sum4.s_out TO Engine.s_throttle

  CONNECT El_Actuator.s_out TO Aero.s_elevator
  CONNECT Ail_Actuator.s_out TO Aero.s_aileron
  CONNECT Rdr_Actuator.s_out TO Aero.s_rudder

  CONNECT Aero.s_Mach TO Engine.s_Mach

  CONNECT Aero.forces_out, Engine.forces_out TO \
    Frame.forces_in

  CONNECT Frame.state_out TO Trimmer.state_in, \
    Aero.state_in, Engine.state_in, state_out

END COMPONENT
```

It can be seen in the code how easily the different library components fit together to construct the *F-16* component and, therefore, how easy it would be to model any other aircraft with the extensive reuse of the code.

These interconnections are schematically illustrated in Figure 4, where the four IN ports are located to the left of the figure and the OUT port is located to the right. It may be observed that the input control signals are added to those obtained in the *Trimmer* component, that is say, to those of the stationary flight. In the next step, the three control parameters *de*, *da* and *dr*, after being input to each of the *Actuator* components, are input to the *F16Aerodynamics* component to calculate the aerodynamic forces. Meanwhile, because the *F16Engine* component already includes the actuator model, the control parameter ***p*** is directly input to it to calculate the propulsive forces. In this way, the output of both components provide the *Frame* component which completes the system state. Since several components need to know the aircraft's state at each instant, the outputs from the *Frame* components go to the rest, as in the case of the *F16Engine* component which needs to know at each instant the flight Mach, which is provided by the *F16Aerodynamics* component.

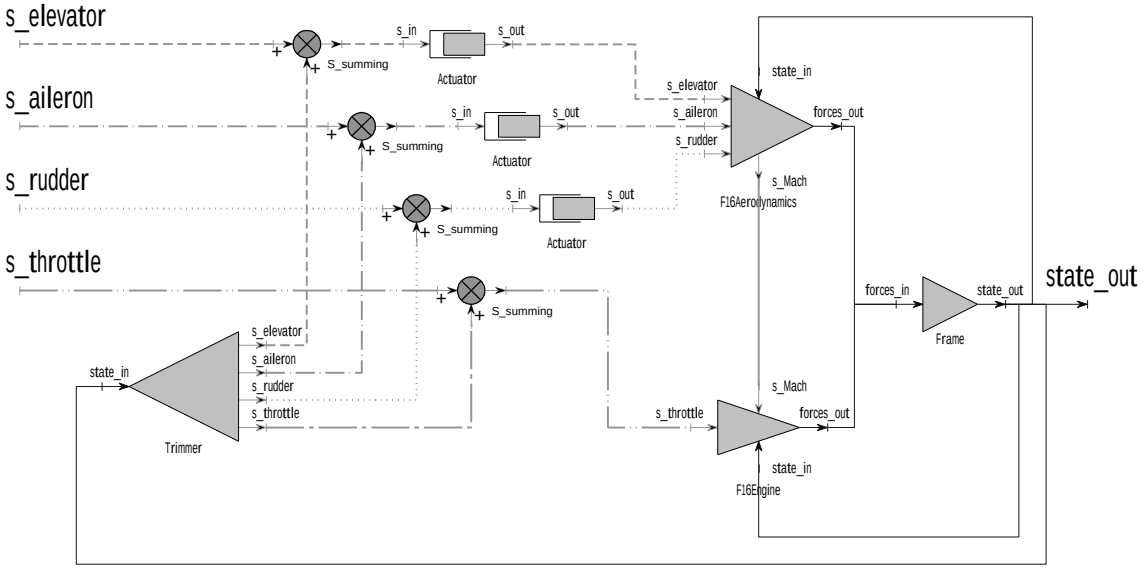

Figure 4: Connection of components for the construction of the F-16 model

9 MODELLING THE AUTOPILOTS

To implement an autopilot model in EcosimPro we have to follow some steps which are very similar to those we followed when we constructed the *F-16* model. This is why it is necessary to have an overview of the components required and how they are interconnected.

We must first remember that, just like the complete control system, the autopilots are specific to each system and must be implemented for each particular aircraft. However, those which are described below serve as an example for the subsequent development of models for other aircraft.

We will need some components which are common to any control system and defined in the control library (multipliers, integrators, adders and subtractors, etc) and others which belong to the FLIGHT_SIM library. Some of these components respond to different actuator models, as in the case of the *ARI* (Aileron-Rudder Interconnect) component which is defined in the file FLIGHT_actuators.el, and others respond to different sensor models which allow us to obtain some aircraft state variables like the angle of attack sensor (AOASensor), the sideslip angle sensor (SDLSensor), the gyroscope sensor (GyroSensor) and the accelerometer sensor (AccelSensor), which are defined in the file FLIGHT_sensors.el. It is over these variables that control will be exercised.

In keeping with the overview we have established, the following is a detailed description of two characteristic autopilots. The first, denominated *F16qAutopilot*, is a longitudinal autopilot which will control the angular pitch velocity (q). The second, denominated *F16prAutopilot*, is a lateral-directional autopilot which will control the angular roll velocity (p) and keep the angular yaw velocity (r) at nil. Both components are defined in the file FLIGHT_autopilots.el.

9.1 "F16qAutopilot" COMPONENT

This component is defined with two IN ports, ie *State* and *analog_signal*, and one OUT port, ie *analog_signal*. The values of the aircraft's state and the signal for angular pitch velocity are input via the IN port and the signal for the deflection angle of the elevator is output via the OUT port.

Figure 5 illustrates a detailed sketch of the control system in which the two IN ports are located on the left and the OUT port on the right.

It can be seen how the state variables are input to the different sensors and how the elevator deflection is calculated.

This model contains some of the components belonging to the CONTROL library and others that belong to the FLIGHT_SIM library. Those from the control library are typical components such as filters, integrators, multipliers, adders, etc. Those from the FLIGHT_SIM are, in this case, a sensor which measures the aircraft's angle of attack (AOASensor) and a gyroscope which measures the angular pitch velocity (GyroSensor).

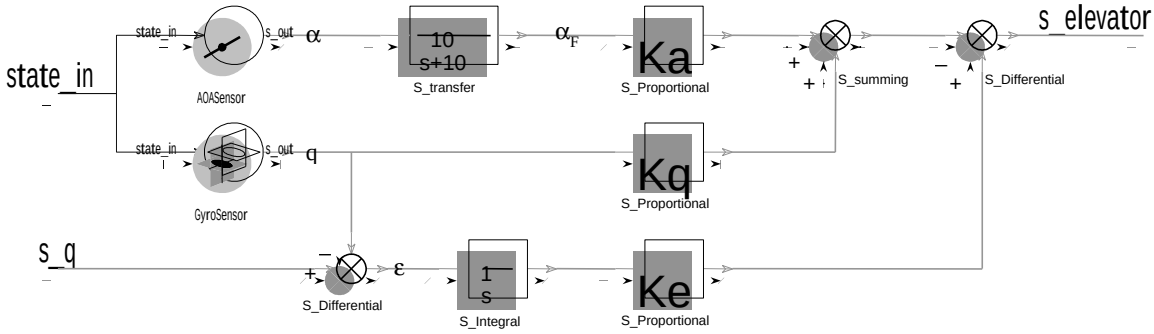

Figure 5: Connection of components for the construction of the longitudinal autopilot

9.2 "F16prAutopilot" COMPONENT

This component is defined with three IN ports, ie one *State* and two *analog_signal*, and two *analog_signal* OUT ports. The values of the aircraft's state and the pilot-controlled angular roll and yaw velocities are input via the IN port and the controlled deflection angles of the ailerons and the rudder are output via the OUT port.

Figure 6 illustrates a detailed sketch of the control system where the three input ports are located on the left and the two output ports are located on the right.

We can again see how the state variables are input to the different sensors to obtain the feedback signals from the autopilot and the controls signals to the control system.

In this model, only the different components will be made to stand out. Thos pertaining to the control library are of the same type and do not require a great deal of attention. On the other hand, there are some new components from the FLIGHT_SIM library such as an accelerometer which measures the aircraft's lateral acceleration (AccelSensor) and an actuator which is denominated ARI (Aileron-Rudder Interconnection) which is normally associated with a typically hydraulic or electrical system (K_{ARI}), in addition to the two gyroscopes which in this case measure the angular pitch and yaw velocities of the aircraft (GyroSensor).

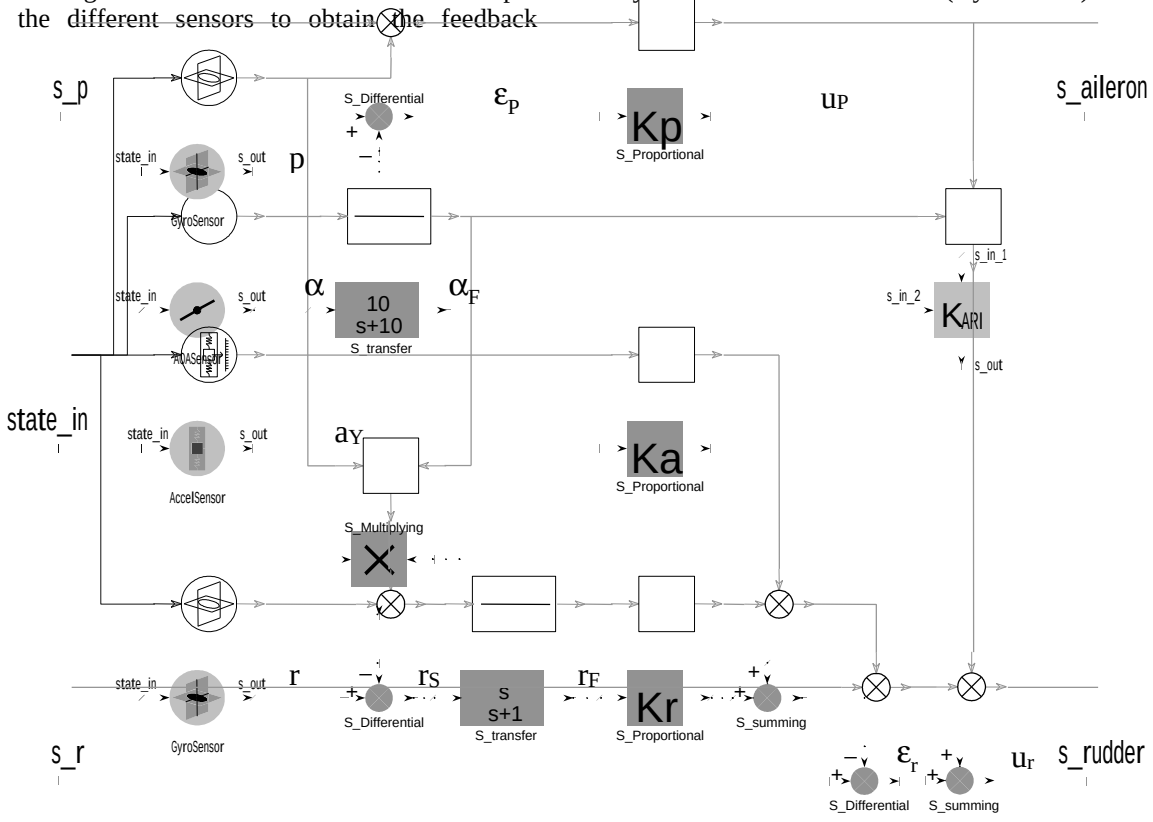

Figure 6: Connection of components for the construction of the lateral-directional autopilot

10 CONSTRUCTION OF THE "F16_Autopiloted" COMPONENT FROM THE FLIGHT_SIM LIBRARY

We are now ready to connect the two autopilots to the F-16 model (component *F16*) and this will give us the final F-16 model with longitudinal and lateral-directional l, which is defined in the file FLIGHT_F16_Autopiloted.el.

This component is defined with a single *analog_signal* IN port because there is only one remaining aircraft control parameter free, that of the engine. The engine power control values are input via this IN port.

Figure 7 below illustrates how the different components are connected.

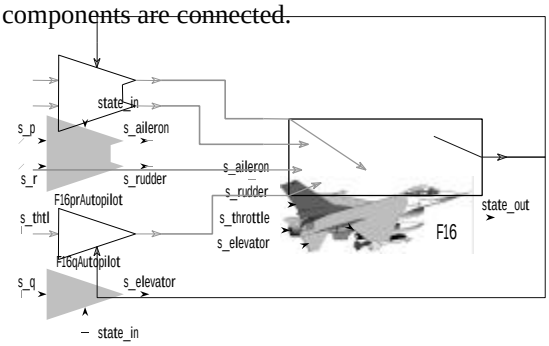

Figure 7: Connection of components for the construction of the F-16 model with longitudinal and lateral-directional control

11 SIMULATION WITH ECOSIMPRO OF MODELS FROM THE FLIGHT_SIM LIBRARY

All the previous work has led to the obtainment of two models (two components) with which we will be able to carry out a series of interesting experiments to demonstrate the simulation capabilities and the possible applications in the field of flight mechanics that the EcosimPro tool offers. On the one hand, with the *F16* component we will be able to simulate the aircraft's behaviour and, on the other, with the *F16_Autopiloted* component, the aircraft's behaviour under the control of both autopilots (longitudinal and lateral-directional).

Taking into account the F-16 model chosen and the capabilities we have implemented, this library facilitates the simulation of stationary and non-stationary manoeuvres, some examples of which are given below. To be exact, we will obtain curves of the aircraft's most characteristic actuations (for different flight conditions and modes) and its response curves to the control parameters (elevator, ailerons, rudder and power), as well as the different manoeuvres governed by the two autopilots.

11.1 F-16 PERFORMANCE CURVES

The aim is to obtain a series of state or control variables for different symmetrical and stationary flight conditions for different values of the free parameters. For example, power or elevator deflection curves required for the horizontal, symmetrical and stationary flight conditions on a vertical plane based on the aircraft's speed, or for the coordinated turn conditions on a horizontal plane based on the turn rate for a given aircraft speed.

All these results are obtained by calculating the steady states of the *F16* component for different values of the *Trimmer* data, so that the desired stationary flight conditions are imposed. We can see in the first example how easily this is done in an experiment carried out with EcosimPro

```
-------------------------------------------------------------------------
-- Steady state conditions:
-------------------------------------------------------------------------
--   VT = [40 m/s, 240 m/s]
--   ClimbAngle = 0 rad
--   psi' = 0 rad/s
-------------------------------------------------------------------------
  Trim = 1
  Trimmer.ClimbAngle_req = 0.
  Trimmer.dpsi_req = 0.
  FOR(Trimmer.VT_req = 40.; Trimmer.VT_req < 240.; \
      Trimmer.VT_req = Trimmer.VT_req + 5.)
    STEADY()
  END FOR
```

In the first step we fix the value of the *Trim* variable to calculate the stationary flight conditions. In the second step we fix the horizontal flight conditions on a vertical plane. And the third step gives us the results for the different aircraft velocities within a range of 40 and 240 m/s.

This is how we achieve the following F-16 actuation curves for horizontal, symmetrical and stationary flight on a vertical plane. See Figures 8 and 9.

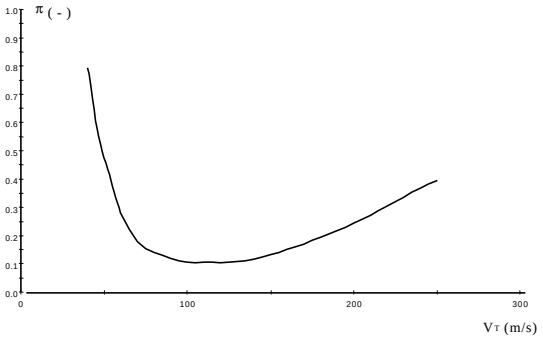

Figure 8: Power curve for horizontal, symmetrical and stationary flight on a vertical plane

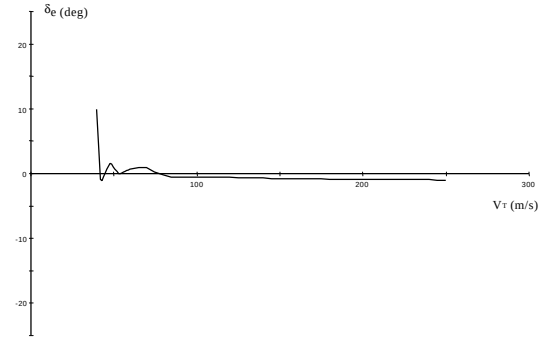

Figure 9: Elevator deflection curve for horizontal, symmetrical and stationary flight on a vertical plane

Drawing an analogy for the coordinated turn conditions on a horizontal plane, we obtain the load and the roll angle curves; see Figures 10 and 11.

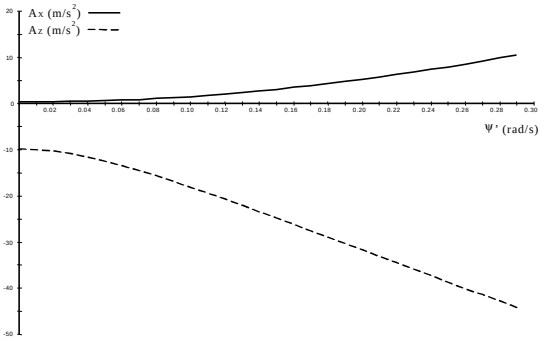

Figure 10: Load curve for coordinated turn on a vertical plane

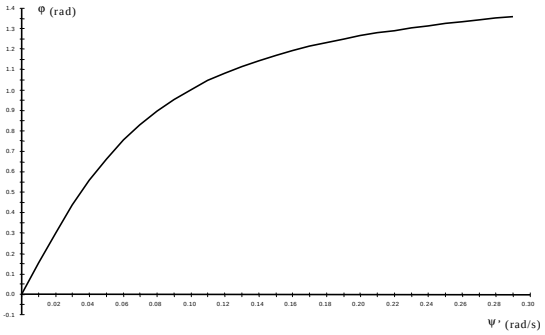

Figure 11: Roll angle curve for coordinated turn on a vertical plane

11.2 F-16 RESPONSE CURVES

The aim of these experiments is to obtain the response of the F-16 a different disturbances in the aircraft's control parameters, characteristic response times, maximum response peaks, and so on.

To achieve this, starting from a determined stationary flight condition, we must subject the F16 component to these disturbances in the control parameters (boundary conditions) and propagate the model.

The initial stationary conditions are obtained in the same way we have obtained them in the previous experiments. So what we will see now is only how to implement the boundary conditions to get an example of disturbance in the form of a *doublet* on the control surface of the elevator.

```
BOUNDS    -- set expressions for boundary variables
  s_elevator.signal = 2.*step(TIME, 1.0, newInt()) - \
    4.*step(TIME, 1.5, newInt()) + 2.*step(TIME, 2.0, newInt())
  s_aileron.signal = 0
  s_rudder.signal = 0
  s_throttle.signal = 0
```

In Figures 12 and 13 we can see the real surface deflection (including the behaviour of the actuator) and the response of the angles of pitch (θ) and attack (α).

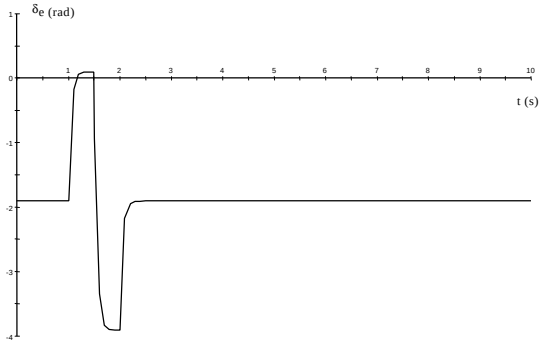

Figure 12: Deflection of the control surface (elevator)

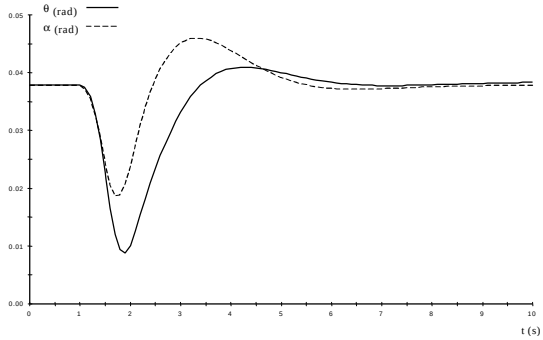

Figure 13: Response curve of the angles of pitch and attack

This can be done in the same way for the remaining state variables and for the remaining control parameters of the aircraft. We will then obtain equivalent response curves without any extra effort, thanks to the reuse of the code.

11.3 MANOEUVRES CONTROLLED BY THE F-16 AUTOPILOTS

Lastly, we will see two different examples of manoeuvres with the use of the autopilots. The first involves only longitudinal control and therefore takes place on a vertical plane. The second involves

both and the final trajectory turns out to be three-dimensional.

Figure 14 shows a graph with the trajectory of the first manoeuvre. The vertical axis represents the height of the aircraft and the horizontal axis represents the horizontal displacement of the aircraft.

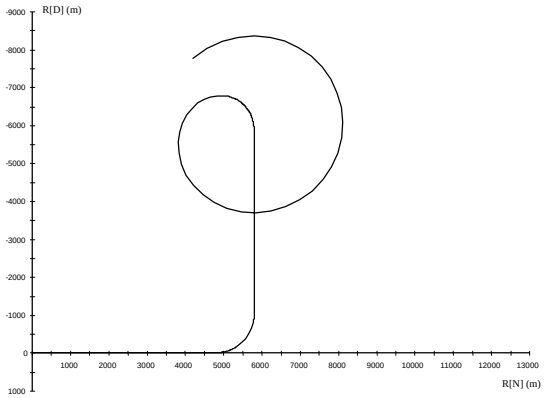

Figure 14: Trajectory of the manoeuvre executed with the longitudinal control of the aircraft

To achieve this manoeuvre, the autopilot is required to carry out the following control sequence: 5s of stationary horizontal flight ($q_c = 0$), 10s at 8.7º/s of pitch rate ($q_c = 8.7$), 30s of stationary vertical flight ($q_c = 0$) and 50s at 10º/s of pitch rate ($q_c = 10$). It is also established that at the instant the pitch manoeuvre starts, the engine control parameter is fixed at 100%.

This is implemented with the same ease that we have seen in the experiments carried out up till now.

```
BOUNDS    -- set expressions for boundary variables
  prAutopilot.s_p.signal = 0.
  qAutopilot.s_q.signal = 8.7*( step(TIME, 30., newInt()) - \
    step(TIME, 40., newInt()) ) + 10.*step(TIME, 70., newInt())
  prAutopilot.s_r.signal = 0.
  s_thtl.signal = ( 1. - F16.Trimmer.throttle ) * \
    step (TIME, 30., newInt())
```

Some interesting results obtained from this experiment, and which show additional library capabilities, can be seen in Figures 15 and 16. The first figure shows the aircraft's response to the control exercised by the autopilot, and how the autopilot obtains the pilot-controlled pitch rates. The second figure shows the values taken by the control parameters to achieve the required manoeuvre (particularly the elevator deflection and the engine control parameter).

Exactly as we have done for these magnitudes, so can any other state variable be immediately represented.

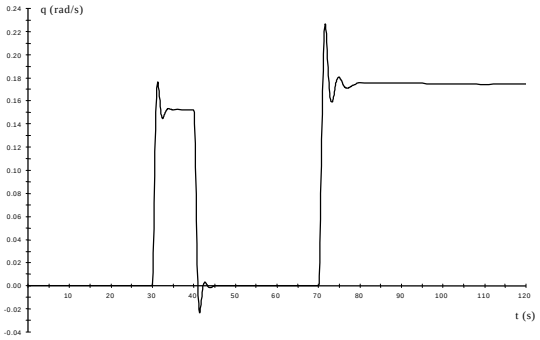

Figure 15: Angular pitch rate obtained by the autopilot

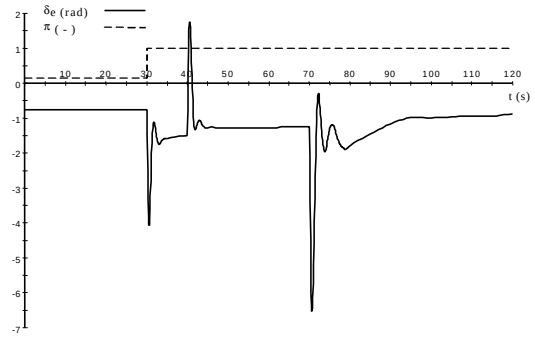

Figure 16: Elevator deflection and engine control parameter necessary for the manoeuvre

Figure 17 shows the trajectory resulting from the second manoeuvre. The vertical axis represents the height of the aircraft and the lateral displacement, and the horizontal axis represents the horizontal displacement of the aircraft.

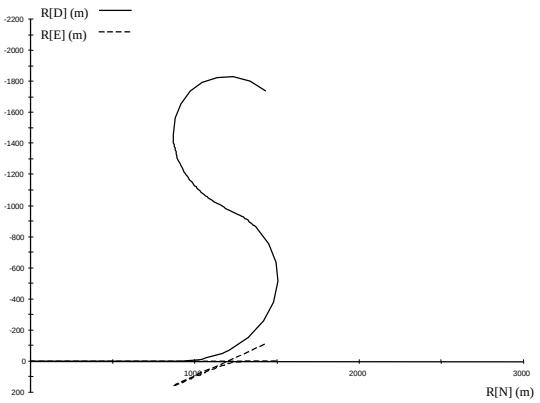

Figure 17: Trajectory of the manoeuvre executed with the aircraft's lateral-directional control

To achieve this manoeuvre, the autopilot is required to carry out the following control sequence: 5s of stationary horizontal flight ($q_c = 0$, $p_c = 0$) and 30s at 15º/s of pitch rate ($q_c = 15$). When the aircraft is almost at the highest *looping* point, it is also required to execute a 180º roll for 2s at 150º/s of roll rate ($p_c = 150$). Lastly, as in the previous case, it is established that at the instant the pitch manoeuvre starts, the engine control parameter is fixed at 100%.

In this case, the lateral autopilot control will be used over the angular yaw rate to prevent sideslipping during the manoeuvre ($r_c = 0$).

All this can be implemented with exactly the same ease as we have seen in all the experiments up till now.

```
BOUNDS    -- set expressions for boundary variables
  prAutopilot.s_p.signal = 150.*( step(TIME, 15., newInt()) - \
    step(TIME, 17., newInt()) )
  qAutopilot.s_q.signal = DtoR * 15. * step(TIME, 5., newInt())
  prAutopilot.s_r.signal = 0.
  s_thtl.signal = ( 1. - F16.Trimmer.throttle ) * \
    step (TIME, 5., newInt())
```

Some interesting results obtained from this experiment can be seen in Figures 18 and 19. The first figure shows the aircraft's response to the control exercised by the autopilots and how it tries to obtain the required angular rates. The second figure shows the values taken by the control parameters to achieve the required manoeuvre.

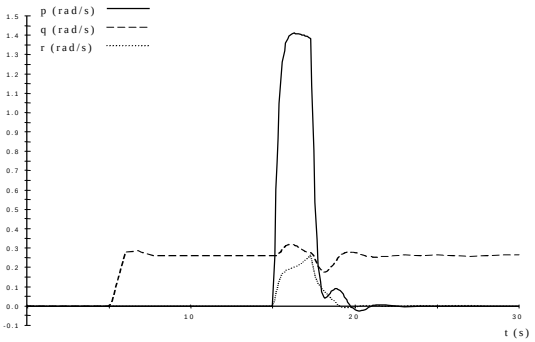

Figure 18: Angular rates obtained by the autopilot

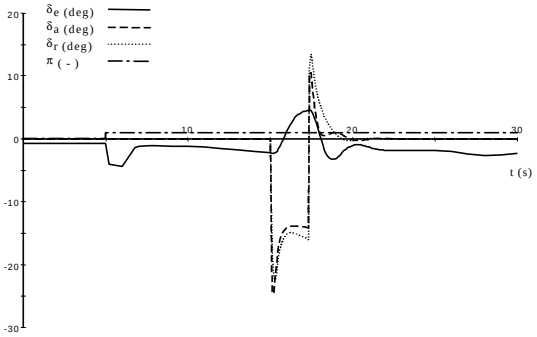

Figure 19: Aircraft control parameters necessary for the manoeuvre

With these experiments we have been through all the components developed for flight mechanics models which have been implemented in the FLIGHT_SIM library, and well as through all the possible applications that EcosimPro offers in this field. The following is a list of more capabilities that could be added, many of which are already being included in the library.

New Capabilities for the Library

- Improve the *Frame* component so that the aircraft's attitude can be integrated using Euler angles or quaternions
- Develop a model which included the effects of the atmosphere in movement (winds, gusts, turbulence, etc)
- Further detail in the actuator and sensor models
- Develop models for other aircraft, such as helicopters and missiles
- Develop components to represent the most typical autopilots

Conclusions

With the FLIGHT_SIM library we can quickly and easily build aircraft models which can be used to calculate the stationary actuations or the transient response of the aircraft. This capability offered by EcosimPro facilitates the use of a single model, thus reducing the effort put into building models and the work involved in their maintenance.

The autopilots are simulated in much the same way it is represented for aircraft models, using the components from the CONTROL library (previously incorporated in EcosimPro) together with those from the FLIGHT_SIM library.

Acknowledgements

The authors thank Pedro Cobas for all his advice.

References

[1] Stevens, B. L. & Lewis, F. L., (1992) "Aircraft control and simulation", John Wiley & Sons, Inc, USA.

[2] Martínez García, J. J. and Gómez Tierno, M. A., (2000) "Apuntes de mecánica del vuelo", Publications by the E. T. S. I. Aeronáuticos, Spain.