AiMT Advances in Military Technology
Vol. 9, No. 1, June 2014

Real-time Flight Model for Embedded Simulator

P. Frantis[1*] and A. Cuzzolin[2]

[1] *University of Defence, Czech Republic*
[2] *Ecole de l'Air, France*

The manuscript was received on 22 October 2013 and was accepted after revision for publication on 20 April 2014.

Abstract:

The paper deals with development of a simple flight model suitable to be implemented as a C++ algorithm for real-time usage. This model is used for embedded flight simulator in a synthetic vision system. The developed flight model is validated by comparing it longitudinal and lateral responses with typical response modes of an airplane.

Keywords:

Simulation, flight model, real-time, programming

1. Introduction

Nowadays, the producers of miniaturized, low power embedded systems added support of accelerated 3D graphic that allows more complex visualization for embedded applications. In our department there were two projects focused on synthetic vision systems and their application in airplane avionics recently solved. The real-time 3D visualization systems based on digital geographic data was developed during the work on these projects. This 3D visualization system was connected with the Inertial Navigation System (INS) unit and it allows in-flight visualization of avionics data and 3D terrain. The architecture of the synthetic vision system is modular (Fig. 1). The synthetic vision system consists of 3 independent modules that are connected by software buses [4].

The 3D terrain visualization core processes airplane position and orientation data to visualize 3D environment based on digital geographic data.

The Avionics module visualizes various avionics instruments based on data from INS module.

[*] *Corresponding author: Communication and Information Systems Department, University of Defence, Kounicova 44, Brno, Czech Republic, +420 973 442348, petr.frantis@unob.cz*

The INS module is the interface layer to the used INS unit. The synthetic vision system can use various INS hardware units so this layer provides a unified interface to these unit and works like a hardware abstraction layer.

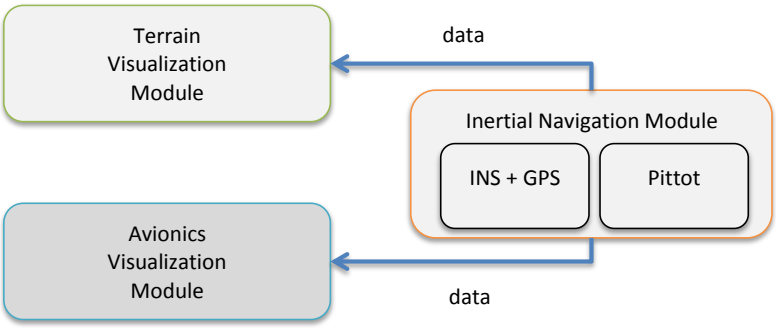

Fig. 1 Modular architecture of synthetic vision system

This synthetic vision system underwent flight tests that proved its functionality and potential for additional development. One of the possible improvements was add an embedded training feature that would allow flying the planned route on the ground before the actual flight or using this system as simplified flight simulator. Thanks to the module based architecture the adding of flight simulation feature can be done by just replacing the INS module with the new Flight Simulation module (Fig. 2). The Flight Simulation module can be implemented as a simulated INS unit that processes flight controls positions and uses built-in simulation model the produce simulated INS output data. Using this architecture the rest of the synthetic vision system does not need to be changed so the embedded flight simulation feature can be implemented at minimal cost.

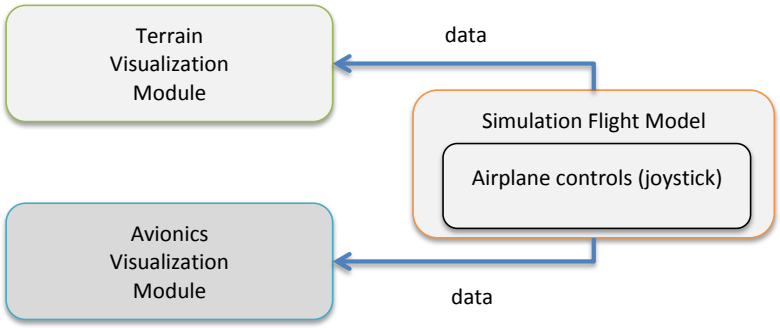

Fig. 2 Flight simulation module in synthetic vision system

2. Flight Model

In the paper the following notation will be used:

$R_0(O,\overrightarrow{x_0}, \overrightarrow{y_0}, \overrightarrow{z_0})$	Earth frame of reference
$R_a(G,\overrightarrow{x_a}, \overrightarrow{y_a}, \overrightarrow{z_a})$	Aerodynamic frame of reference
$R_b(G,\overrightarrow{x_b}, \overrightarrow{y_b}, \overrightarrow{z_b})$	Plane frame of reference
O	A fixed point relative to earth
G	Centre of mass of the plane
$\vec{V}$	Velocity of G relative to the atmosphere
α	Angle of attack
β	Angle of sideslip
T	Transformation matrix $R_a \rightarrow R_b$
x, y, z	Coordinates of the plane in R_0
ψ	Azimuth angle
θ	Elevation angle
ϕ	Bank angle
R	Transformation matrix $R_a \rightarrow R_0$
$\overrightarrow{\Omega_{b/0}}$	Angular velocity of the plane relative to the earth
(p, q, r)	Coordinates of $\overrightarrow{\Omega_{b/0}}$ in R_b basis
$\overrightarrow{F_P}$	Weight
$\overrightarrow{F_T}$	Thrust
F_0	Maximum thrust at the sea level
$\overrightarrow{F_a}$	Aerodynamic force
(X, Y, Z)	Coordinates of $\overrightarrow{F_a}$ in R_b basis
$\overrightarrow{M_a}$	Moments in G due to the aerodynamic force
(L, M, M)	Coordinates of $\overrightarrow{M_a}$ in R_b basis
ρ	Density of air
ρ_0	Density of air at the sea level
m	Mass of the plane
I	Inertia matrix
A, B, C, E	Inertial momentums
S	Wing area
l	Mean aerodynamic chord
C_x	Drag coefficient
C_{x0}	Zero-lift drag coefficient
k_i	Induced drag coefficient
C_y	Lateral coefficient
C_z	Lift coefficient
α_0	Zero-lift angle of attack
C_l	Rolling moment coefficient
C_m	Pitching moment coefficient
C_n	Yawing moment coefficient

$C_{i\alpha}$, $C_{i\beta}$	Gradient of force or moment coefficients
C_{ip}, C_{iq}, C_{ir}	Rolling, pitching, yawing damping coefficients
$C_{i\delta l}$, $C_{i\delta m}$, $C_{i\delta n}$	Coefficients representing the influences of the control surfaces
δl	Position of the roll control (angle)
δm	Position of the pitch control (angle)
δn	Position of the yaw control (angle)
δx	Position of the throttle in percent
ω_{PO}	SPPO pulsation (cf. 3.1.1.)
T_{PO}	SPPO period
T_{PH}	Phugoid period
τ_{RS}	Roll subsidence time constant
ω_{SO}	Sideslip oscillation pulsation
T_{SO}	Sideslip oscillation period
τ_{SM}	Spiral mode time constant

$i \in \{z, y, l, m, n\}$, All angles are in radians.

The flight model is based on these hypotheses:

- Earth is fixed and flat. Thus the earth frame of reference is a Galilean reference frame.
- Gravity is constant relative to the altitude. Thus is g a constant vector. In what follows, we will consider that:

$$g = \|\vec{g}\| = 9.81 \text{ m/s}^2$$

- The plane mass and inertial momentums are constant. This hypothesis would not be valid if we studied a cruise flight, due to fuel consumption.
- The plane is rigid and symmetrical (relatively to geometry and mass). Thus the inertia matrix can be written as follows in the plane frame of reference:

$$I = \begin{pmatrix} A & 0 & -E \\ 0 & B & 0 \\ -E & 0 & C \end{pmatrix}_{R_b} \quad (1)$$

- The direction of thrust is the axis of the fuselage, $\overrightarrow{x_b}$, and its application point belongs to the line (G, $\overrightarrow{x_b}$). Especially it supposes that the thrust is symmetrical (in the case of a multiengine plane).
- The density of air,ρ , is constant relatively to the altitude and time in what follows, we will consider that: ρ = 1.255 kg/m^3.
- There is no wind thus $\vec{V}$, the velocity of G relative to the atmosphere, is also the velocity of G relative to the earth.

2.1. Algorithm

As a reminder, the aim of this project is to simulate flight by implementing a real-time flight model algorithm in a synthetic vision program. Thus, the coordinates of the plane in the earth frame of reference – *x*, *y*, *z* – and the angular position of the plane –

ψ ,θ , Φ – are the outputs needed. Obviously, the inputs are the control surfaces and throttle positions – δl, δm , δn , δx.

Here are the equations that will be used to estimate the outputs [3]:

$$\begin{cases} m\left[\dot{V}\cos\alpha\cos\beta - V\left(\dot{\alpha}\sin\alpha\cos\beta + \dot{\beta}\cos\alpha\sin\beta\right) + V(q\sin\alpha\cos\beta - r\sin\beta)\right] = \\ = X + F_T - mg\sin\theta \\ m\left[\dot{V}\sin\beta + V\dot{\beta}\cos\beta + V(r\cos\alpha\cos\beta - p\sin\alpha\cos\beta)\right] \\ = Y + mg\cos\theta\sin\phi \\ m\left[\dot{V}\sin\alpha\cos\beta + V\left(\dot{\alpha}\cos\alpha\cos\beta - \dot{\beta}\sin\alpha\sin\beta\right) + V(p\sin\beta - q\cos\alpha\cos\beta)\right] = \\ = Z + mg\cos\theta\cos\phi \end{cases} \quad (2)$$

$$\begin{cases} A\dot{p} - E\dot{r} + (C - B)qr - Epq = L \\ B\dot{q} + (A - C)rp + E(p^2 - r^2) = M \\ C\dot{r} - E\dot{p} + (B - A)pq + Erq = N \end{cases} \quad (3)$$

$$\begin{cases} \dot{\theta} = q\cos\phi - r\sin\phi \\ \dot{\phi} = p + \tan\theta\,(q\sin\phi + r\cos\phi) \\ \dot{\psi} = \dfrac{q\sin\phi + r\cos\phi}{\cos\phi} \end{cases} \quad (4)$$

The Eq. (2) is describing movement according to centre of gravity; the Eq. (3) describes movement around centre of gravity. The Eq. (4) illustrates the relationship between positional angles and angular velocities. As this is a highly coupled system, there is no way to solve it analytically without more hypotheses [2, 3, 5]. The C++ language will be used to estimate it by computing it in a short time period (integration method).

2.2. Integration Method

The principle is to use the previous known value to calculate the next one thanks to the derivate. The shorter the period of time is the more accurate results are obtained.

Firstly, the Eqs (1) and (2) can be rewritten as follows:

$$\begin{cases} \dot{V} = \dfrac{1}{\cos\alpha\cos\beta}\left[\dfrac{X + F_T}{m} - g\sin\theta - V(q\sin\alpha\cos\beta - r\sin\beta)\right] \\ \dot{\beta} = \dfrac{1}{V\cos\beta}\left[\dfrac{Y}{m} + g\cos\theta\sin\phi - V\left(\dot{V}\sin\beta - r\cos\alpha\cos\beta - p\sin\alpha\cos\beta\right)\right] \\ \dot{\alpha} = \dfrac{1}{V\cos\alpha\cos\beta}\left[\dfrac{Z}{m} + g\cos\theta\cos\phi - \dot{V}\sin\alpha\cos\beta + V\dot{\beta}\sin\alpha\sin\beta - \right. \\ \left. - V(p\sin\beta - q\cos\alpha\cos\beta)\right] \end{cases} \quad (5)$$

$$\begin{cases} \dot{p} = \dfrac{1}{A - \dfrac{E^2}{C}} \left\{ L + \dfrac{E}{C}[N + (A - B)pq - Erq)] + (B - C)qr + Epq \right\} \\ \dot{q} = \dfrac{1}{B}[M + (C - A)rp - E(p^2 - r^2)] \\ \dot{r} = \dfrac{1}{C}[N + E\dot{p} + (A - B)pq - Erq] \end{cases} \tag{6}$$

Assuming $\alpha, \beta \ll 1$, $(\dot{\alpha} \sin\alpha \cos\beta + \dot{\beta} \cos\alpha \sin\beta) \approx 0$. This is why this term does not appear anymore in $\dot{V}$ equation.

Moreover, whatever differentiable scalar function *f*:

$$\dot{f}(t_n) = \lim_{\Delta t \to 0} \frac{f(t_n + \Delta t) - f(t_n)}{\Delta t} \tag{7}$$

Hence, assuming $\Delta t \ll 1$,

$$\dot{f}(t_n) \approx \frac{f(t_n + \Delta t) - f(t_n)}{\Delta t} \quad \text{i.e.} \quad f(t_n + \Delta t) \approx \dot{f}(t_n)\Delta t + f(t_n) \tag{8}$$

This simple Euler method has been chosen for its simplicity and easy implementation in programming language for the real-time application. The above equation is computed every rendered frame so the Δt is very small (less than 16 ms). This simple method gave us stable results even for high speed airplanes (tested Mirage model). But in general it is intended for subsonic aircrafts (L-39, L-159).

With all these elements, we can easily write the C++ algorithm which will provide us the coordinates and angular positions of the plane in the real time [1].

2.2. The C++ algorithm

Definitions

At first it is necessary to define the aircraft structure that defines all the features of the plane, from its mass to its aerodynamic coefficients.

Secondly it is necessary to define a position structure. The position structure gathers the coordinates of the plane in the earth frame of reference – *x ,y, z* – and the angular position of the plane – ψ, θ, ϕ – but also other values needed to calculate the next position: the angle of attack α, the angle of sideslip β, the coordinates of $\overrightarrow{\Omega_{b/0}}$ in the plane frame of reference – *p, q, r* – and the velocity $\vec{V}$.

The main function that solves the flight model equation is the evolution function. According to the Eq. (8), it returns the next value thanks to the previous one, the derivate and the time period (Euler method).

The first four parameters of evolution function represent the position of control surfaces and throttle at a given moment. The first one *P* is the position of the plane at this same moment. The second one is the time period Δt and third one is the type of aircraft used. And finally *P2* is the position of the plane at the next moment $(t + \Delta t)$.

Calculation of the new angular position and the new angular velocity

First of all, we have to calculate the forces and moments applied to the plane at the moment *t*. Then, as explained in paragraph 2.1, we can estimate the new values of ψ, θ, ϕ, using Eq. (6).

Calculation of the new coordinates

We have to apply the transformation from the plane to the earth frame of reference *R*. The velocity will be calculated in the earth frame of reference. Then the new coordinates are estimated according to the Eq. (8).

Calculation of the new velocity

Finally, the Eq. (5) provides us the features of the new velocity that we will be needed for the next call of evolution.

3. Evaluation of the Model

The C++ algorithm, developed in the previous section has been implemented in the synthetic vision system and thus, using a joystick, it becomes a simulator. Even if we can experiment that it reacts qualitatively like a real plane, we have to confirm it quantitatively. This is why we are going to analyse the responses of our real-time flight model algorithm. We choose the well documented Airbus A300 features.

For the evaluation purposes the joystick control is not used. Firstly the plane position is initialized in a longitudinal equilibrium i.e. a straight and level flight (except for the spiral mode). Then in a loop is executed the flight model function with the inputs corresponding to the mode that is evaluated. Finally the result values from each time step are transcribed from a data text file and the values are represented by a curve.

There were evaluated these flight modes:

- Longitudinal modes
  - Short period oscillation (SPPO)
  - Phugoid
- Lateral modes
  - Roll subsidence
  - Sideslip oscillation and Dutch rol Roll l
  - Spiral mode

Due to the paper length restriction just the Spiral mode will be explained here.

3.1. Spiral Mode

The spiral mode represents evolution of the bank angle ϕ when $\phi_0 \neq 0$ and $\delta l = 0$. Either this mode is stable and converges to zero, or it is unstable and the bank angle increases. Assuming $\theta \approx 0 \approx q$, theory shows that the time constant of the spiral mode is:

$$\tau_{SM} = \left(\frac{C_{lp}C_{n\beta} - C_{l\beta}C_{np}}{C_{lr}C_{n\beta} - C_{l\beta}C_{nr}}\right)\frac{V_0}{g} \tag{8}$$

The Fig. 3 shows the response of our flight model algorithm, with the following inputs (angles in radians).

Tab. 1 Response of flight model

ϕ_0	p_0	V_0	$\delta m = \delta m_0$	δl	$\delta x = \delta x_0$
10°	0	$100\ m/s$	-0.212	0	0.344

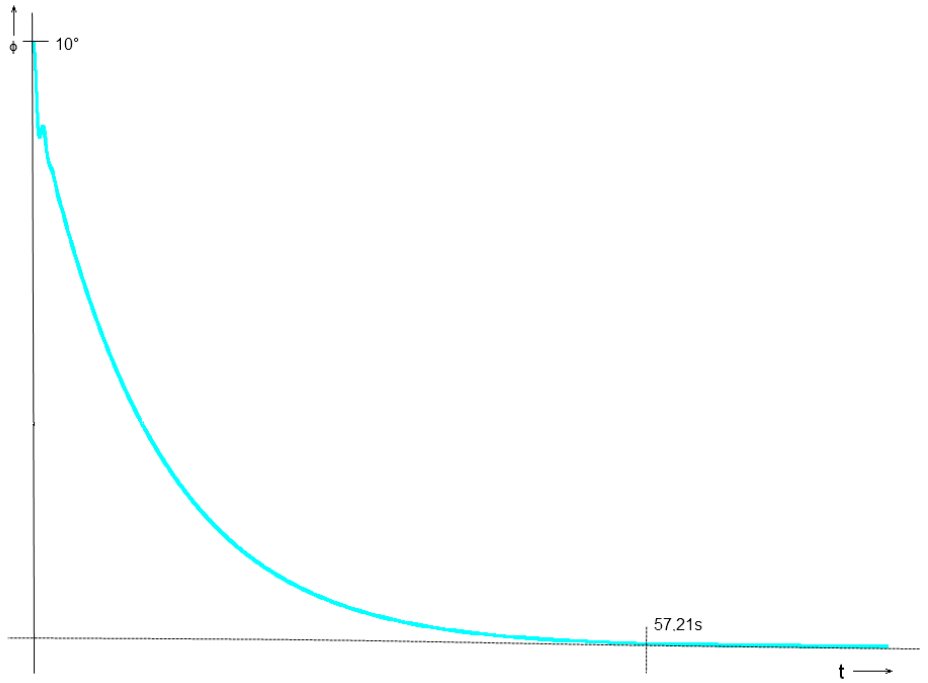

Fig. 3 Spiral mode with $\phi_0 = 10°$

Thus, we can compare the response of our model with the theoretical result:

Tab. 2 Comparison of results

RS time constant	Theory	Flight model algorithm	Difference
τ_{SM}	53.86 s	57.21 s	6.2 %

Time constant express the minimal acceptable time to double the initial roll angle.

3.2 Discussion of Results

Accordingly, the analysis of the longitudinal and lateral modes (only spiral mode is described in this paper) confirms that the algorithm developed responds as an actual aircraft. Indeed the five typical response modes have been pointed out. The differences with theoretical values are between 4 and 8.2 percent.

4. Conclusion

This project was aimed to develop a real-time flight model algorithm that can be used as an embedded simulator in a synthetic vision system (Fig. 4). The real-time issue is important so this why we have not used a detailed aerodynamic model that would have

implied lengthy calculation. Moreover, one key point was the model used to describe aerodynamic forces and moments. Indeed, the strong hypotheses, which conduct to the linearization of aerodynamic coefficients this flight model is only valid in a small range of angles of attack.

The C++ algorithm was written using the equations and the basic integration method by working on a short period of time. Thus it provides the coordinates of the plane in the earth frame of reference and the angular position of the plane in response to the control surfaces and throttle positions.

This algorithm has been finally evaluated by analysing its longitudinal and lateral responses to basic inputs. Typical response modes have been highlighted. Thus we can affirm that we have simulated the motion of an actual plane. However, as explained above, this flight model algorithm is not valid for extreme angles of attack. As a result, it does not demonstrate stall phenomenon for instance. Widen the flight envelope simulated, especially to extreme angles of attack, non-linear aerodynamic phenomena should be taken into account. This would require a more complex model regarding aerodynamic coefficient.

Lastly, set apart the aerodynamic issues, we could study the issue of latest fighter jets. Indeed, in order to maximize their manoeuvrability, these aircrafts are built unstable. This is thanks to their fly by wire technology that pilots manage to control them. Thus, another module should be implemented in the program in order to simulate what happens between the joystick orders and the controls motion.

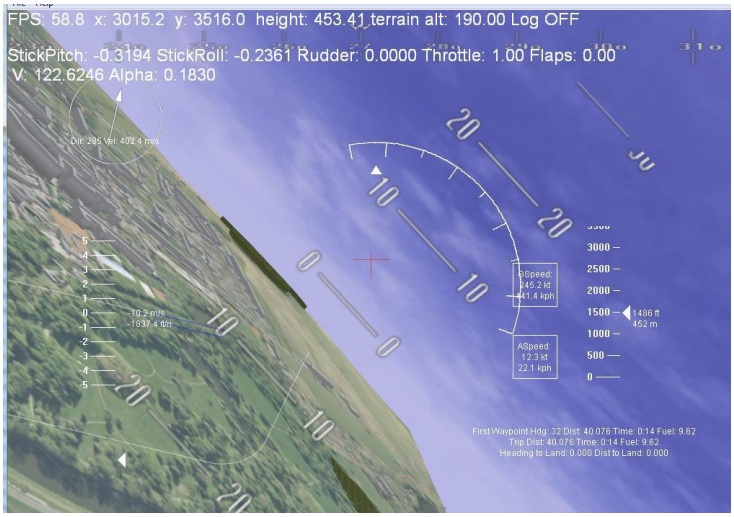

Fig. 4 Using of the flight model in simulator

References

[1] BEZET, BOCHET. *Language C* (in French). Ecole de l'Air, Octobre 2011.

[2] BOIFFIER, J-L. *Dynamique du Vol* (in French). SupAéro, 2001.

[3] BOVET, L. *Dynamique du vol de l'avion* (in French). Ecole de l'Air, Juillet 2006.

[4] FRANTIŠ, P. *Multipurpose low-cost synthetic vision system.* 29th Digital Avionics Systems Conference, October 2010.

[5] VEYSSET, F. *Modélisation et identification de comportements de l'avion en vol turbulent par modèle à retards* (in French). Ecole Centrale de Lille, 2006.

Acknowledgement

This paper was carried out during the "Learning Through Research" internship as a results of the EUAFA programme between French Air Force Academy and University of Defence.