Advanced modeling with a Symbolic based approach Application to the modeling, control design and real-time control and HIL simulation of a quadrotor helicopter

N. Gachadoit[1], A. El Hadri[2], A. Benallegue[2], A. Seba[3], B.Vidalie[1]

1. Maplesoft, 32 boulevard Colbert, 92160 Antony France, france@maplesoft.com
2. LISV, 10-12 avenue de l'Europe, 78140 Vélizy France
3. ISEP, 28 Rue Notre Dame des Champs 75006 Paris France

Abstract:

In this contribution we present a new type of modeling tool taking full advantage of symbolic computing. Physical components are described from block libraries, built directly from Modelica, now a well established consortium, and a fast developing physical language (see [1]). Directly from the block diagram description the analytical equations of the system are automatically generated and simplified with a powerful symbolic engine ([2],[3]).
On a quadrotor helicopter system, we show how the system is described and how easy it is to get access to the equations. With highly optimized equations, the model runs very fast, and shows very interesting performances in real time. The automatically generated equations are used in a math tool ([2]) to analyze the system, and carry out control design steps. Besides traditional analyses based on numerical computing, several symbolic techniques are used. Finally the model is tested on a real time experimental setup ([4],[5],[6]), showing good comparison.

Keywords: modeling, simulation, symbolic computing, Modelica, control design, real time, quadrotor, helicopter

1. Introduction

Quadrotor helicopters (also called quadricopters) have generated an increasing interest in the past years as they provide easier stability and have shown very interesting performances for drone activities. The French DGA and ONERA organize a joint challenge on Micro Drones to motivate innovation in the area, where Quadricopters show very good performances (see [7]). Quadricopters started to be used in Autonomous Unmanned Micro Aerial Vehicle applications with companies like MicroDrones since more than 5 years (see [8]). Also interesting projects have been started to share resources on related unmanned observation systems (see [9]). Building a reliable, easy to use model for such a new technology is critical to help better dimension future quadrotor helicopters.

The objective of this paper is to demonstrate on a quadrotor helicopter model the interest of a new kind of simulation tool based on symbolic computing technology ([2],[3]).
In a first step we will show how the model is described with MapleSim ([3]), a new physical multi-domain tool. Directly from Modelica components libraries the block diagram of the system is easily built. The system's equations are generated automatically from the block diagram description and simplified with the Maple symbolic engine ([2]). This results in a very efficient formulation, but also allows to access to the analytical equations of the system directly into Maple. Getting access to the equations allows any analysis or design work with all the math power of Maple ([2]). To show the benefits of getting the equations in a math tool we demonstrate all the steps of a control design. In particular we highlight the use and interest of the symbolic engine ([2]). One of the benefits is that the equations and model description are kept in a live, very easy to read technical document. This document is linked to the model and keeps all the appropriate knowledge on the model, the analysis, the design, and even the validation steps.

In a second step the model is validated against real data. For this the model was given to the LISV (Laboratoire d'Ingénierie des Systèmes de Versailles), a well respected research lab, with a high level background on control and real time management of quadrorotor helicopter, and fully equipped with a real time experiment. The objective was to have the model tested by an independent and renown institution, and build relevant real time comparison data.

1. Modeling the quadrotor helicopter

As its name says, a quadrotor helicopter is an aircraft equipped with 4 rotors.

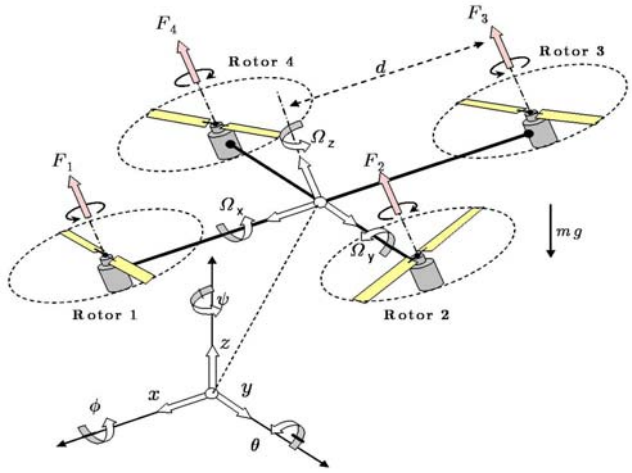

Fig 1 : Quadrotor helicopter [5]

Two pairs of rotors (see Fig. 1) turn in opposite direction in order to balance the moments, and by varying the rotation speeds the system can be oriented and positioned as needed. For example when applying the same rotor speeds altogether with the same quantity, the lift forces will change the altitude z of the system. The yaw angle ψ is obtained by speeding up the clockwise motors or slowing down depending on the desired angle direction. The motion direction according (x, y) axes depends on the direction of tilt angles (pitch angle θ and roll angle φ), whether they are positive or negative.
The dynamical model of the quadrotor helicopter has six outputs {x, y, z, φ, θ, ψ} while it has only four independent inputs. Therefore the quadrotor is an under-actuated system ([4], [6]). It is not possible to control all of the states at the same time.

In the system we studied, each rotor is driven by a DC motor. The body of the system is supposed to be rigid.
In MapleSim, the body of the quadrotor is represented via a single 3D rigid body, taken from the multibody library. The position of each rotor is materialized via a local frame, defined with respect to the frame of the quadricopter rigid body (Figure 2).
The DC engines are built from the electrical and mechanical libraries in MapleSim. The components available in MapleSim libraries are based on Modelica. The modeling principle for the physical parts of the system is based on an acausal modeling paradigm. For each component MapleSim is able to get the equations from the related Modelica code. When a link is drawn between two components, MapleSim, is aware of the through and across variables related to the components, and generates the equivalent of equilibrium equations.

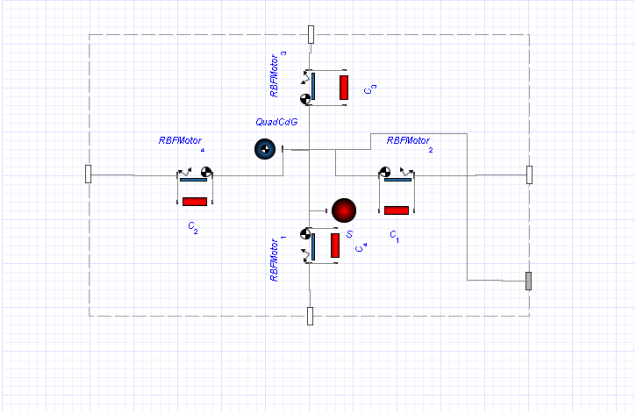

Fig 2 : Rigid body and rotor positions

As MapleSim uses the full power of symbolic computing available in Maple, it is also unit aware, meaning that it is able to know the physical dimensions of a quantity. The result is that it checks for the physical dimensions and guaranties the consistency of the model. The same comment applies to the units applied to a typical model, where MapleSim takes care of any conversion as appropriate.

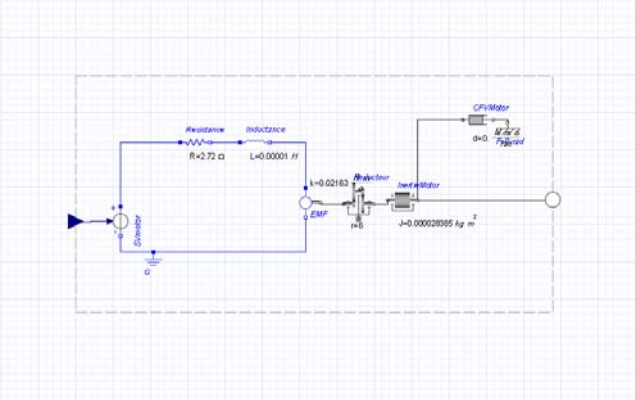

Fig 3 : Electrical drive

As we can see in Figure 3, the electrical drive is modeled with a LR electrical circuit and a mechanical part including a gear, an inertia and damping. Thanks to the acausal modeling paradigm, the model description is very close to the real system.

Once the DC drive has been defined, it is put in a hierarchy. The base mount of the rotor is represented by a 3D rigid body. The blade is also represented by a rigid body. They are linked with a cylindrical joint. To represent the lift force, a signal driven force driver is connected to the base mount component. The force is computed with respects to the rotation speed of the DC drive (See Figure 4).
The base itself is connected to the relevant local frame on the quadricopter body. The resistance of the rotor to the air is modeled as a torque function of the rotor speed.

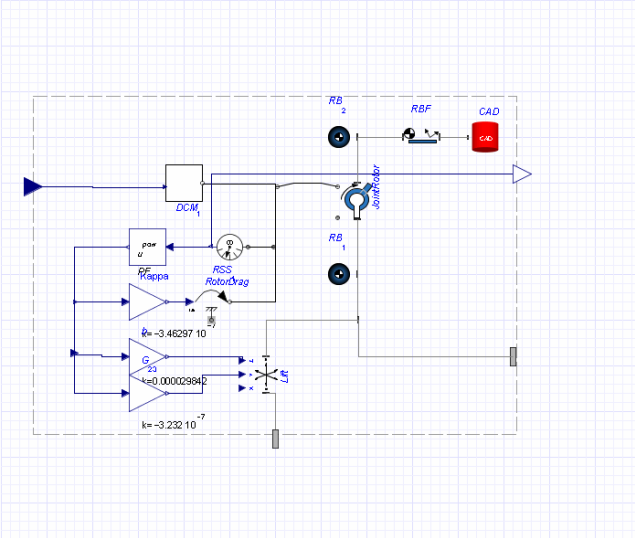

Fig. 4 Rotor mechanics

The overall rotor model is put into a hierarchy and duplicated three times, so that each rotor be represented and connected to the related local frame on the quadrotor helicopter body.

The remainder of the model,consists in describing the control layers. We will show later how they were designed from the dynamical equations of the quadrotor helicopter.

To get an overview of the overall modeling process, let us assume we completed the control design step and start running the simulation.

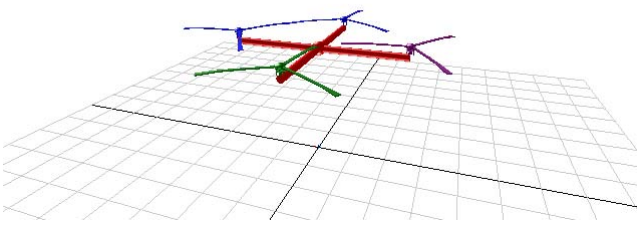

Fig. 5 3D animation

Before we start the simulation, we set its duration and select a numerical solver. MapleSim uses the numerical power of Maple combined with its very powerful symbolic engine. More precisely physical multi-domain system models often result in high index Differential Algebraic Equations (DAE). To cope with this type of equations MapleSim takes advantage of symbolic computing to reduce the index order.

When the simulation is launched, MapleSim starts by generating automatically the equations of the system. At the simulation stage the equations do not need to be displayed. The user is still provided with general data on the number of generated equations before simplification and after simplification. It is also possible to ask a compiled version of the model, so that it may be run faster. For the quadrotor helicopter the number of equations generated is of 955. After symbolic simplification it is of 99, and after index reduction the resulting number of equations is of 23. The benefit is that those equations represent exactly the same behavior as the “unsimplified” formulation.
To have an idea of what kind of symbolic techniques are used to simplify equations, it is interesting to look at what MapleSim does to get access to the exact, highly optimized set of equations, for multi-body mechanical systems. For this, MapleSim parses the multi-body model, and uses advanced simplification techniques, using graph theory. A good explanation on these simplification techniques can be found in [10], [11]. At the end of the simulation, besides rich plot facilities, the user gets also access to a 3D animation of the simulation (Fig. 5).
At this stage the benefit of symbolic computing resides in the size of the simplified formulation, and on the index reduction for Differential Algebraic Equations (DAE).
The other very important benefit is that the equations can be accessed from Maple, a powerful mathematic tool. We will illustrate this added value in next step, on the design of a controller on the quadrotor helicopter.

2. Designing a LQG controller

We will not describe Maple in detail here (see [2]). It is just necessary to know that Maple has both a very powerful symbolic engine, and provides state of the art numerics. It also provides a technical document interface, which allows to “play” in a very interactive way with the equations, and read the equations in a natural format (eg the way mathematics appear in a scientific book). This provides a way to both read the analytical equations of a given system, manipulate them, and then apply any needed calculus as appropriate.
The equations of the quadrotor helicopter are recovered in Maple via a simple template. In Figure 6 we show a screen shot of the equations recovered in Maple.

Even if they were simplified, they are very large. To achieve the same optimized formulation by hand it would have taken a lot of time and errors. When they are in Maple the equations can be manipulated with a comprehensive set of symbolic commands. As an example, model can be differentiated per any of the parameters. Knowing the analytical form of the

differentiate per a parameter is very useful to know if the model is sensitive to changes on this parameter. This technique, also called sensitivity analysis, is very useful to define efficient plans of experiment.

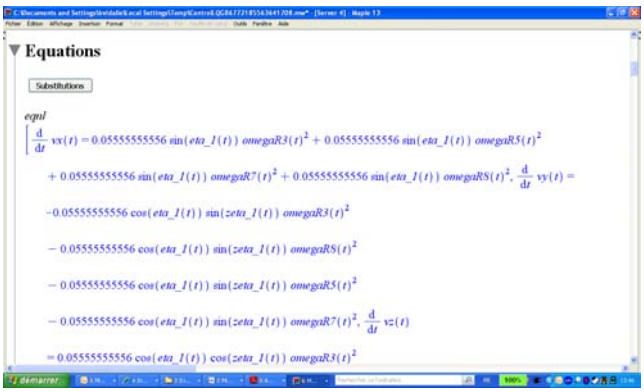

Fig. 6 : Screenshot of model equations in Maple

To start let us define an equilibrium point, we will use to linearize our system. We consider the stationary state as the equilibrium, that is when the applied propeller speeds, maintain a given altitude. To get this equilibrium point we compute the command needed to equilibrate the lift force and gravity.

```
sol1 := solve( [op( evalf ( subs( vx(t) = 0, vy(t) = 0, vz(t)
    = 0, wx(t) = 0, wy(t) = 0, wz(t) = 0, x(t) = 0, y(t)
    = 0, z(t) = 0, zeta_1 (t) = 0, eta_1 (t) = 0, xi_1 (t)
    = 0, map( rhs, eqnl ) ) ) ), omegaR3 (t)
    = omegaR5 (t), omegaR3 (t) = omegaR7 (t),
    omegaR3 (t) = omegaR8 (t) ], [ RI1(t), RI2(t), RI3(t),
    RI4(t), omegaR3 (t), omegaR5 (t), omegaR7 (t),
    omegaR8 (t), i1 (t), i2 (t), i3 (t), i4 (t) ]) [ 1 ]
```

Fig. 7 : Solve directly on system's equations

Figure 7 displays the command used in Maple for this task. The important point is that no formulation by hand is needed for this first calculus. We will not recall this fact for all subsequent steps, but it is critical to reduce the design cycle time and the number of possible errors. If it brings a huge added value during the first design, it is even easier when needing to modify the model and replay an overall design automatically. This is achieved through the access to symbolic equations of the system, but also with the document interface (see Figure 6), and a full Math language (more than 4000 math and graphics commands, see [2]).
The calculated setting point (see Figure 8) is then used to linearize the system.
Maple provides a standard format and a comprehensive set of tools to work on dynamical systems representations and produce analyses. This includes representation through Differential Equations, State Space, Transfer Function, Zero Poles. Taking advantage of symbolic computing, it is possible to start by simply describing a system through its differential algebraic equations.

```
RI1(t) = 176.2662883 , RI2(t) = 176.2662883 , RI3(t)
    = 176.2662883 , RI4(t) = 176.2662883 , omegaR3 (t)
    = 6.644170377 , omegaR5 (t) = 6.644170377 ,
  omegaR7 (t) = 6.644170377 , omegaR8 (t)
    = 6.644170377 , i1 (t) = 19.42332334 , i2 (t)
    = 19.42332334 , i3 (t) = 19.42332334 , i4 (t)
    = 19.42332334
```

Fig. 8 Computed equilibrium point

Assuming it has the relevant form (eg to be displayed as a transfer function, system has to be linear), the system's representation can be switched from one another representation using directly the symbolic equations. This means that physical parameters used in one form (usually the starting point is a differential equation set), will show accordingly in other forms. This also applies when switching from continuous to discrete representations. For our design we take the linearized system in the State Space representation form. The resulting dynamical system has 19 outputs, 4 inputs and 20 states.

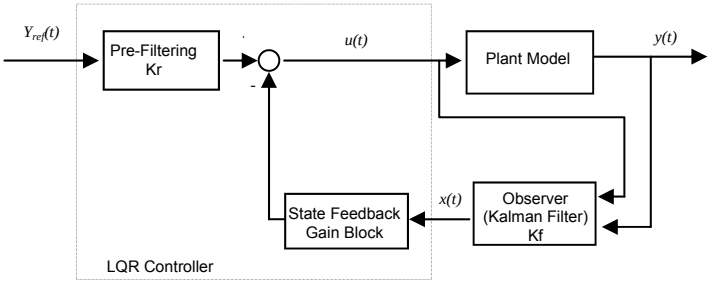

Fig. 9 : Controller structure

To control the system, we first build a standard state feedback gain, assuming that we are able to measure all states. As the number of possible measures is less than the number of actual states in the system, we use an observer to estimate the lacking values. The design is then done in two parts :

- Design of an LQR controller
- Design of a Kalman filter

To design the LQR controller, we just need to call a command in Maple, directly on the automatically obtained linearized state space representation (see Figure 10).

```
Kc, Kr := Control:-LQR(syslin1, 15,
    controlled_outputs = [11, 12, 13, 17, 18,
    19])
```

Fig . 10 : LQR control design

When the control law has been computed it is stored in a state space representation. Directly in Maple it is then possible to analyse the closed loop response of

the system with a single command, taking the plant and the gain control.

```
DynamicSystems:-ResponsePlot ( sysSFcl, [ 0, 0, 50, 0, 0,
    0 ], duration = 20, output = [ zout ], numpoints = 200,
    dsolveargs = [ maxfun = 1000000 ], gridlines = true )
```

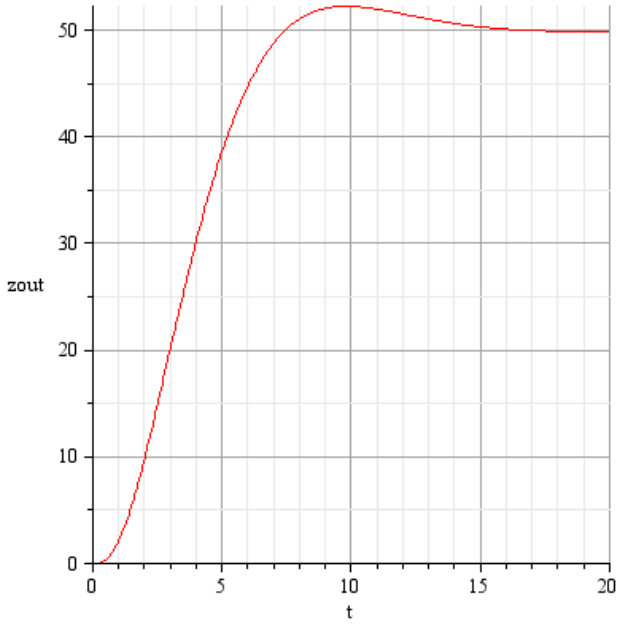

Fig. 11 Closed loop step response of the LQR

The further steps of the design, consist in designing a Kalman filter in Maple, and check the overall closed loop system. One of the math challenges of control design is to solve the Riccati equations. The precision used for this calculus may be critical. Symbolic computing allows to adjust the numerical calculus precision at any arbitrary level. The actual solving process is done using numerical techniques relying on arbitrary numerical precision. The end result is that the precision can be set to the needed level and better solutions may be achieved.
The overall controller (including a Kalman Filter) is defined in a state space representation. When it has been analyzed in Maple (eg with a closed loop step response), it is automatically sent to MapleSim.
When it has been imported in MapleSim, the controller is then tested in simulation on the full model.
Assuming that the controller design is completed, the next step in the process is to test the model or the controller in a real configuration.
For this MapleSim is able to generate automatically C code. It also provides links to Real time platforms, from providers like dSPACE or National Instruments. In our next step, we will get the model used on a real system.

3. Testing the model on a real system

The model was provided to the LISV team, with all means to test and analyze it through numerical simulations, but also through analyses of the equations of the system in Maple and MapleSim.

The LISV already produced important contributions on quadrotor helicopters and their control ([4],[5],[6]). They used their fully equipped real time platform with a quadricopter to carry out the hereafter comparison tests.

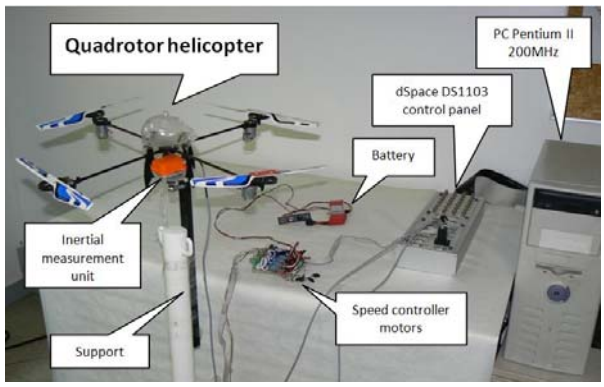

Fig. 12 : Quadricopter real time platform

The real time system used is based essentially on a dSPACE DS1103 card being used to generate the control signals and the sensor data acquisition used in the control loop. The flying machine used is a mini rotorcraft with four rotors (Draganflyer IV without the electronics control) manufactured by Draganfly Innovations, Inc.(http://www.rctoys.com). (Physical characteristics in table 1).

Weight (including the support)	400g
Blade diameter	29cm
Blade step	11cm
Drive to C.G. distance	20.5cm
Drive reduction rate	1 : 6

Table 1

The objective of this experiment is to safely test the proposed attitude controller. For this a stationary ball joint base is used (Figure 12). This base gives the aircraft unrestricted yaw movement, and around ±40° of pitch and roll, while restricting the aircraft to a fixed point in the three-dimensional space ([6]). To measure the aircraft angles and the angular velocity an Inertial Measurement Unit was connected to the dSPACE DS1103 serial communication port. The four DC permanent-magnet mini motors are current amplified with intelligent microcomputer speed controllers of type IMCS 25 and driven by PWM signals.

First step consisted in entering the physical parameters on the actual system in the MapleSim model. When this was done, the model was tested

with a control strategy developed by LISV ([6]), first in simulation, and second in real time.

The comparison is done based on real time measurements imported in MapleSim and compared to the closed loop simulated response.

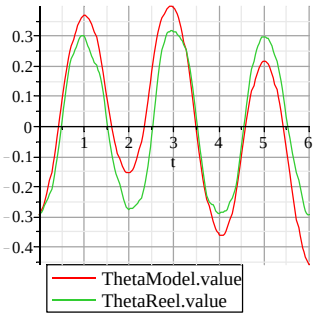

Fig. 13 Pitch Angle comparison

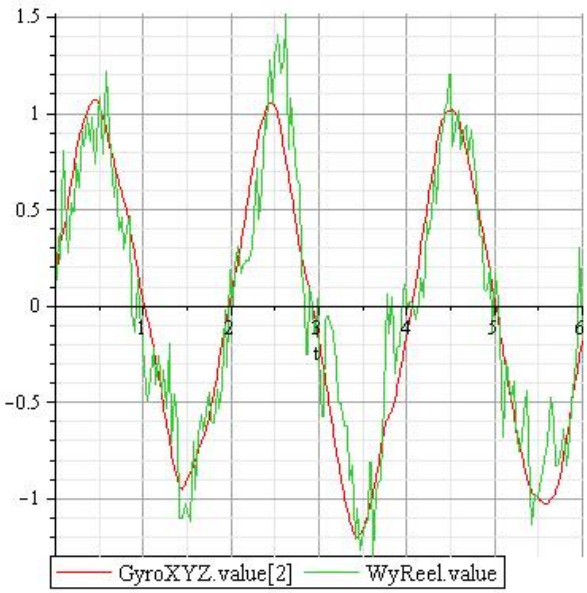

Fig 14 Pitch Angular Speed comparison

In Figures 13 and 14, the measured (in green), and simulated (in red) values on pitch angle are compared. The model shows a close behavior to the actual real time system.

4. Conclusion

Directly from a graphical and natural representation of the system, the analytical equations of the a quadrotor helicopter have been automatically generated and simplified with one of the most powerful symbolic engine available today (see [2]). The result is a very clean and fast simulation. Using advanced symbolic computing to produce the formulation, enables to build the exact representation with an optimized set of state variables and number of operations. This brings a direct advantage on the simulation time, but also helps to provide very fast and efficient real time models for HIL (Hardware In the Loop) simulations. Getting access to the equations of the system from a strong math engine, with both symbolic and numeric capabilities, allows carrying out advanced analyses and designs very fast.

All the steps of the design were not only saved in a Maple document, but were also commented in order to keep all the knowledge on the model, and on the design steps, in a very readable and interactive format. In the end we hope this project will provide to the many and very active quadricopter projects, a new set of methodologies and tools for their designs, and also new simulation and analysis technologies, which will enable more innovations in future projects.

This project has also provided a very challenging and interesting system, to illustrate the added value of a new modeling technique, based on symbolic computing; showing advances in modeling, analysis, design and knowledge capture.

5. References

[1] http://www.modelica.org/
[2] Maple user documentation (http://www.maplesoft.com/documentation_center)
[3] MapleSim user documentation (http://www.maplesoft.com/documentation_center)
[4] T. Madani and A. Benallegue "Control of a Quadrotor Mini-Helicopter via Full State Backstepping Technique" Proc. of IEEE Conference on Decision and Control (CDC'2006), San Diego, California, USA, December 13-15, 2006.
[5] L. Derafa, T. Madani and A. Benallegue "Dynamic Modelling and Experimental Identification of Four Rotors Helicopter Parameters" Proc. of IEEE Conf. on Industrial Technology (ICIT'06), Mumbai, India, December 15-17, 2006.
[6] L. Derafa, L. Fridman, A. Benallegue and A. Ouldali "Super Twisting Control Algorithm for the Attitude Tracking of a Four Rotors UAV" Accepted in the 11th International Workshop on Variable Structure Systems (VSS'10), Mexico City from June 26th till June 28th, 2010.
[7] http://www.onera.fr/actualites/concours-drones/index.php
[8] http://www.microdrones.com
[9] http://uavp.ch/moin
[10] Kevin Morency, John McPhee, and Chad Schmitke "Symbolic Modelling of Vehicle Dynamics: A Maple Implementation", University of Waterloo, Systems Design Engineering, May 18 2005
[11] John McPHEE, C. SCHMITKE and S. REDMOND: "Dynamic modelling of Mechatronic Multibody Systems with Symbolic Computing and Linear Graph Theory" Math. Computer Modelling Dynam. Syst., Vol. 10,no. 1,pp.1-23, 2004
[11] Control Toolbox For Maple, (http://www.control-toolbox.com/)