MULTI-DOMAIN MODELLING - AN ADVANTAGE IN MODELLING, SIMULATION AND CONTROL EDUCATION

Dejan Dovžan
Borut Zupančič
Gorazd Karer
Igor Škrjanc
Faculty of Electrical Engineering
University of Ljubljana
Tržaška 25
1000 Ljubljana, Slovenia
E-mail: dejan.dovzan@fe.uni-lj.si

KEYWORDS

Multi-domain modelling, object oriented modelling, helicopter, control systems

ABSTRACT

The paper deals with an important aspect of continuous systems modelling and simulation approaches, with possibilities for multi-domain modelling. The traditional approach is based on block oriented schemes in which causal relations play an important role. However this causality is artificially generated in order to fulfil appropriate conditions for simulation on conventional sequential computers. Fortunately new concepts which are based on object oriented approaches, physically oriented connections and algebraic manipulations enable the so called acausal modelling which can efficiently be used for multi-domain modelling. The advantages and disadvantages of traditional and more advanced approaches are discussed.

The described multi-domain approach is illustrated with modelling of a laboratory helicopter using a very popular multi-domain modelling environment Dymola with Modelica language. A multivariable controller was designed for a wide operating range. The control signals which drive the main and the tail rotors enable the desired pitch and rotation (azimuth) angles. The multi-domain object oriented approach enables an efficient coupling of elements from mechanical libraries and a block library for the implementation of control system.

INTRODUCTION

Standardization of languages for modelling and simulation was always very important in the history. However the last standard that was really accepted was CSSL standard (Strauss 1967). Nowadays perhaps the most promising activities are in conjunction with the so called Modelica activities (*www.modelica.org*). After an initiative of the Federation of European Simulation Societies EUROSIM and The Society for Computer Simulation SCS in the middle of nineties a new language Modelica (Modelica 2005, Fritzson 2004), which gives a hope to become a kind of international standard for model exchange, was defined.

A lack of object-oriented properties, which disables the reuse of already build models, is an important disadvantage of many modern modelling and simulation tools. Due to this reason some special-purpose tools were developed (for mechanical, electrical, chemical systems,…). In modelling of complex industrial systems however combinations of systems from different areas, i.e. multi-domain systems, are frequently needed (e.g. mechanical, electrical, hydraulic as well as control systems in mechatronics, particularly within automotive, aerospace and robotics applications). There were many attempts to connect different modelling tools, however the more efficient approach is to use tools which support multi-domain modelling (Van Beck and Roda 2000). Modelica with appropriate working environment e.g. Dymola (Cellier 1991, Dymola 2008) is probably the most promising multi-domain modelling environment based on object oriented approach and 'physical' way of connections between model components.

IMPORTANT FEATURES OF ADVANCED OO ENVIRONMENTS

Block oriented approach versus object oriented modelling and simulation approach

In order to allow the reuse of component models, the equations should be stated in a neutral form without consideration of computational order. This is so called acausal modelling (Zupančič et al. 2005, Zupančič and Sodja 2008). However most of the general-purpose simulation software on the market such as ACSL, Simulink,… assume that a system can be decomposed into block diagram structures with causal interactions (Matko et al. 1992, Cellier 1991). In this procedure state derivatives must be expressed which leads to the explicit state space form

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

$$\dot{x} = f(x,u,t) \qquad (1)$$
$$y = g(x,u,t)$$

which can be efficiently simulated with ODE solvers. u is an input, y is an output and x is a state. It is rare that a natural decomposition into subsystems leads to such a model. Often a significant effort in terms of analysis and analytical transformations is needed to obtain a problem in this form. It requires a lot of engineering skills and manpower and it is error-prone.

In Modelica and some other multi-domain modelling tools it is possible to write balance and other equations in their natural form as a system of differential-algebraic equations, DAE

$$0 = f(\dot{x},x,y,u,t) \qquad (2)$$

where x is the vector of unknowns that appears differentiated in the equation and y is the vector of unknowns that do not appear differentiated. Computer algebra is utilized to achieve as efficient simulation code as possible, similar when we convert the equations ‘manually’ into ODE form.

Object orientation

We shall not talk about general concepts in OO programming where well known terms as encapsulation, data abstraction, inheritance, dynamic binding, … are used. From a modeller point of view, OO means that one can build a model similar to real system: taking a pump, a pipe, a valve, … and connecting them. For an efficient modelling, modelled systems are decomposed into subsystems (components), which are modelled as submodels and then hierarchically connected into a complete model. Modelling languages enable simple reuse of already build models. To reuse a certain model in other models it should be defined as a class. Model classes can be defined by physical laws (energy and mass balance equations and not necessarily with state space description (Equation 1**).** This contributes to a better understanding and reusability of models.

‘Physically’ oriented connections

The appropriate complexity of the implementation of the connections between model components (classes) is probably the most important property of multi-domain OO modelling tools. Connections between submodels are based on variables, which define proper relations and influences between movements, angles, currents, pressures, torques, forces, etc. It is similar as when real systems are built. Figure 1 shows how three mechanical rotational subsystems M1, M2 and M3 are connected. Several physical variables are presented in connections (connectors in Modelica):

τ_i	torques
θ_i	angles
ω_i	angular velocities
α_i	angular accelerations

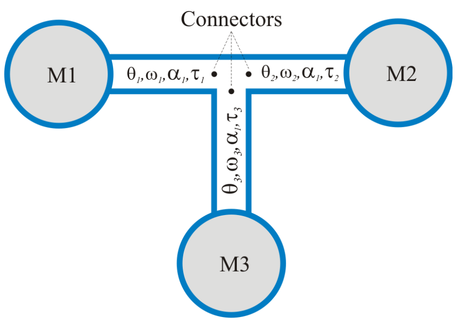
Connectors
M1
M2
M3
$\theta_1,\omega_1,\alpha_1,\tau_1$
$\theta_2,\omega_2,\alpha_1,\tau_2$
$\theta_3,\omega_3,\alpha_1,\tau_3$

Figure 1**:** Connection of three mechanical subsystems

In general there are two types of variables, which are defined in connectors of subsystems: variables that become equal in connection points, in our example angles, velocities, accelerations (ACROSS variables, also potential, temperature, pressure in other types of systems)

$$\theta_1 = \theta_2 = \theta_3 \qquad \omega_1 = \omega_2 = \omega_3 \qquad \alpha_1 = \alpha_2 = \alpha_3 \qquad (3)$$

and variables which sum equals zero (THROUGH variables, in our example this is torque, but also current, force etc. in other systems - prefix FLOW in Modelica)

$$\tau_1 + \tau_2 + \tau_3 = 0 \qquad (4)$$

CONNECTOR is a special structure in which all the variables are collected. Each CONNECTOR has a name, which is composed of a submodel name and a name of a particular connector. Connections in traditional block diagram simulation languages can be treated as a subset of connections introduced by connectors. Namely they possess only variables of the type ACROSS, which become equal in junction points.

Example of a connector definition in Modelica

```
connector flange
    Real theta,omega,alfa;   // Type across
     flow Real tau;          // Type through
end flange;
```

By joining connectors the submodels are connected. During processing the modelling tool automatically generates equations 3 and 4 from submodel connector definitions.

Summary of features of Multi-domain OO modelling and simulation environments

Some important features of modern multi-domain OO modelling and simulation environments (e.g. Dymola with Modelica) are:

- Modelling of various kinds of complex physical systems with object oriented approach.

- Multi-domain tools, equivalently usable for modelling of mechanical, electrical, chemical, thermo dynamical and other systems.
- Possibilities to reuse already built models.
- Acausal model building.
- Hierarchical structure of models.
- Description of processes through physical laws (differential equations) irrespective to the type and purpose of a model.
- Easy and efficient way for submodels connections through connectors (more general then input-output connections known in block oriented simulation tools).
- Symbolical and numerical solving of systems of equations - algebraic formula manipulation.

Modelica – OO modelling standard

Modelica (Modelica 2005, Fritzson 2004) is a modelling language which supports both high level modelling using pre-prepared complex model components and detailed modelling by equations. Graphical diagram layer and textual layer can be efficiently combined. The basic construct in Modelica is class. There are several types of classes with several restrictions: class, model, block, function, package, type etc. There are already several pre-prepared packages or libraries in standard Modelica library, e.g. for block diagrams, for electrical and electronic systems, for mechanical systems, for thermal and some others. Many other libraries are public domain libraries. In the past we developed a Modelica process systems library (Zupančič et al. 2005). However in this paper we focus to the mechanical systems which are even more appropriate for theoretical modelling as process systems (systems with level, flow, pressure, temperature, flow, PH, ..) and therefore very convenient to show the advantages of multi-domain object oriented acausal modelling.

EXAMPLE: MODELLING OF A LABORATORY HELICOPTER

The advantage of the multi-domain OO modelling will be shown with a modelling and control of a laboratory helicopter device. Laboratory set-ups, which model real processes, and mathematical models have a significant role in efficient control design and education. The CE150 (Figure 2) is a laboratory helicopter made by Humusoft (Humosoft 2002). It is used for studying system dynamics and control engineering principles and enables a wide range of practical experiments. The goal of modelling and identification is to prepare a basis for the students' laboratory assignments, such as designing a multivariable controller that ensures satisfactory control in the wide operating range. There are two well known modelling approaches: theoretical and experimental. In helicopter modelling both approaches were combined. The mathematical model is described in details in Karer and Zupančič 2006.

The laboratory helicopter set-up comprises a helicopter body carrying two motors, which drive the main and the tail rotors and a servomechanism, which shifts the centre of gravity by moving a weight along the helicopter's horizontal axis. The helicopter body is mounted to the stand so that two degrees of freedom are enabled:

- the rotation around the horizontal axis (pitch angle - y_ψ) and
- the rotation around the vertical axis (rotation angle or azimuth - y_φ).

Figure 2 depicts the helicopter as it finally appears in animation scheme in Dymola/Modelca environment. The model can be described as a nonlinear multivariable system with three inputs: the voltage driving the main rotor motor (u_1), the voltage driving the tail rotor motor (u_2) and the voltage which drives the servomotor for positioning of the weight. In our case only the first two inputs are taken into control. The weight is positioned to a fixed place, defining the appropriate mechanical characteristic of the helicopter. The system has two outputs: pitch angle (y_ψ) and rotation angle (y_φ). Forces which effect the helicopter body F_{main} , F_{react} and F_{tail} will be explained later.

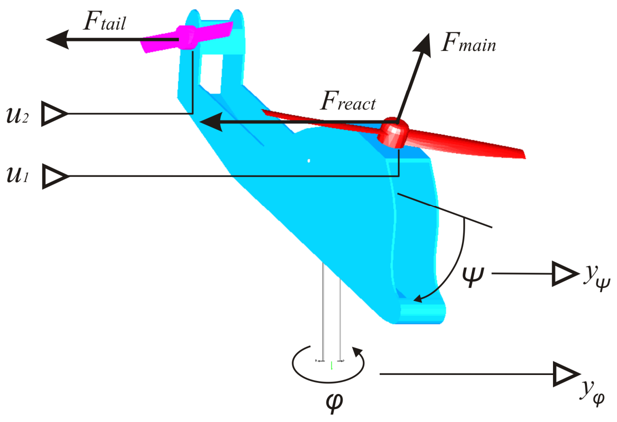

Figure 2: Laboratory helicopter with forces affecting the body and control input/output signals

Multi-domain approach is useful because we combine two different areas: mechanical systems and control systems which are usually modelled with block diagrams. We chose *Modelica* (Modelica 2005, Fritzson 2004) as mechanical systems and block diagrams are included in the Standard Modelica library. *Dymola* environment (Dymola 2008) which supports *Modelica* was used. For animation the body of the helicopter and both rotors were drawn with program *SolidEdge* and with *DxfExport* add-in exported into *dxf* file format. *Dymola* uses *dxf* files to create an animation of model components. The helicopter body and the stand were modelled with the standard *Modelica* libraries *Mechanics\MultyBody and Mechanics\Rotational.*

The designed mechanical model was then used in Matlab/Simulink environment as Dymola block. Namely

Matlab/Simulink environment is very efficient for control system design as Control, Optimization or some other tools (toolbox) can be used. The control scheme designed in Matlab/Simulink was then implemented also in the *Dymola/Modelica* environment using standard *Modelica* libraries *Block\Continuous*, *Block\Sources* and *Block\Math*.

The overall scheme of the top level model in Dymola/Modelica is shown in Figure 3. The scheme is very clear. It consists of the coordinate system definition, the stand model, the helicopter body, the tail and main rotor model (rotor means motor and propeller) and the controller with two reference signals for pitch and rotation angle.

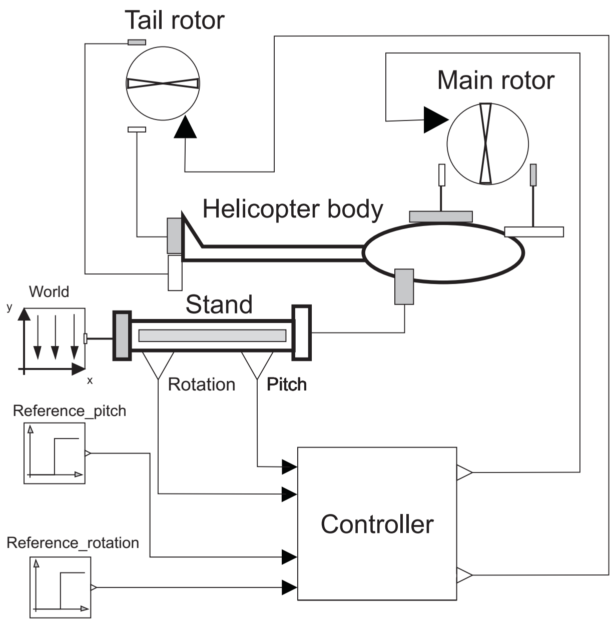

Figure 3: The top level model of the control system

Coordinate system

Coordinate system orientation is such that the gravitational force is in the direction of negative y axis. It is defined by the *Modelica* class *world*.

Model of the stand

The stand (see Figure 4) is modelled with translations (pure transformations of coordinates, straight lines between *a* and *b* connectors) and two bearings (actuated revolute joints *Revol_ext* and *Revol_ext1* in *Modelica*). Rotational axis of the first bearing is y axis (rotation in the horizontal plane) and of the second bearing is z axis (rotation in the vertical plane). The used actuated revolute joints were actually extended with additional outputs in order to get more simple and useful information about the pitch and rotation angles. Namely the connectors of revolute joint have a complex information about movements and velocities with three components in (*x,y,z*) coordinate system. In this way the angle sensors of the set-up were modelled. The bearing friction was modelled using *bearingFriction* blocks, where characteristics of the friction are described with lookup tables. The functions for both friction characteristics were defined using relation

$$\tau_{friction} = B_1\omega + B_2 sign(\omega) \tag{5}$$

in which linear and Coulomb part are taken into account. The relation (5) was implemented with lookup tables in both bearing friction classes.

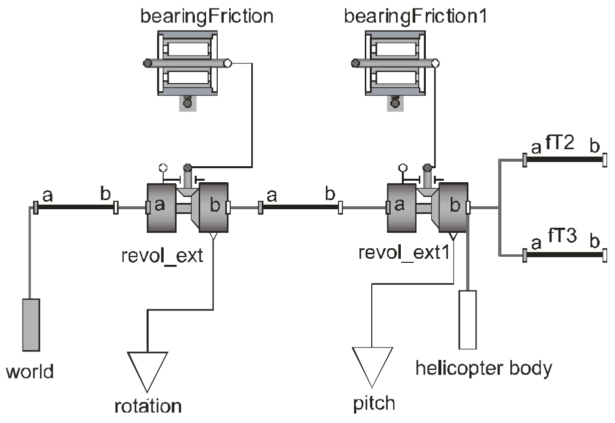

Figure 4: The stand model

The coordinate system is connected to the connector *world* and the body of the helicopter is connected to the connector *helicopter body*. Translations *fT2* and *fT3* do not influence the dynamics. They are inserted for more realistic animation for the rotational axis in *z* direction.

Model of the helicopter body

The central part of this model class (see Figure 5) is the model class *BodyShape* from the *MultyBody* library which is suitable for rigid body modelling. We have to specify the vector from frame *a* to frame *b* resolved in frame *a,* the vector from frame *a* to the centre of mass, resolved in frame *a* and the mass of the rigid body. Beside we have to specify also the inertia tensor with J_{xx}, J_{yy}, J_{zz}, J_{yx}, J_{zx}, J_{zy}. The inertia tensor has to be defined with respect to a coordinate system that is in the origin of the centre of mass of the body. In our example we had to identify and took into account J_{yy} and J_{zz}. With the series of translations (*fTy, fTrx, fTry, fTrz*) we adequately moved the rotors from the rigid body connectors so that the generated forces produced by rotors had the right levers. In *BodyShape* model class the animation shape can be selected. In our case the shape was drawn with a program *SolidEdge*. The drawing was exported to the *dxf* file format. This file was then chosen in the animation options of the *BodyShape* class. Unfortunately Dymola cannot interpret the dimensions of objects in *dxf* files. Therefore we added to the model a set of translations for the proper animation (the encircled part in Figure 5).

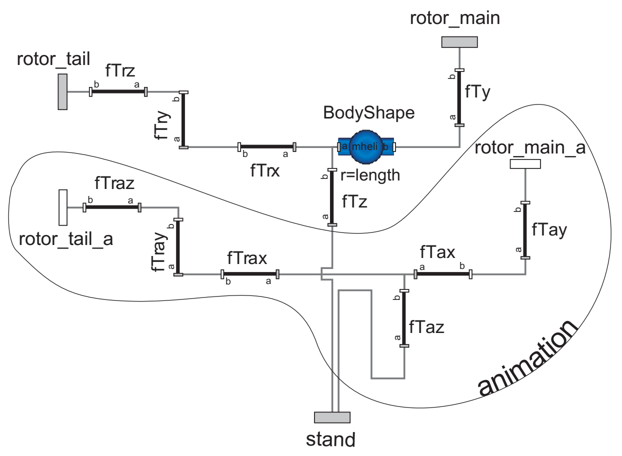

Figure 5: The helicopter body model

Determination of parameters

Modelling in Dymola/Modelica environment is actually a kind of theoretical modelling with a strong support of Modelica libraries. However many parameters have to be experimentally determined. Some experiments were very simple geometrical measurements, some others were more sophisticated. The approaches and the parameter values are presented in more details in Karer and Zupančič 2006. In this paper we shall briefly described only some more sophisticated procedures.

The centre of gravity was determined with some simple mechanical experiments for a predetermined position of the weight which enable the changeable dynamic properties.

For both bearings the constants B_1 and B_2, the moment of inertia tensor J of the regid body were derived with the minimization of criterion function (6), which takes into account the integral of the square difference between the real response y_{real} and the model response y_{mod} with different experiments with appropriate input signals u_1 and u_2 and appropriate initial conditions.

$$\min_{J,B_1,B_2} \int_0^{t_{\max}} (y_{real}(t,u_1,u_2) - y_{\mathrm{mod}}(t,u_1,u_2))^2 dt \qquad (6)$$

Nelder-Mead optimization method in Matlab was used.

Dynamics in one rotational direction (in one plane) was determined when the movements in another plane was disabled. For rotation around around horizontal axis (movements in vertical plane) we obtain

$J_{zz}=2.43\cdot 10^{-3}\ kg\ m^2$
$B_{1z}=1.53\cdot 10^{-4}\ kg\ m^2 s^{-1}$
$B_{2z}=1.03\cdot 10^{-4}\ kg\ m^2 s^{-1}$

and for rotations around vertical axis (movements in horizontal plane)

$J_{yy}=2.02\cdot 10^{-3}\ kg\ m^2$
$B_{1y}=5.52\cdot 10^{-4}\ kg\ m^2 s^{-1}$
$B_{2y}=5.96\cdot 10^{-4}.\ kg\ m^2 s^{-1}$

Model of rotors

Each rotor generates two forces that affect the movements of the helicopter body (see Figure 2). The first force is the 'useful' propulsation force in the direction of rotational axis (F_{main} of the main rotor and F_{tail} of the tail rotor). The second force is a reaction force, which is perpendicular to the rotational axis and acts as a kind of disturbance being a consequence of the rotor rotations and air resistance (F_{react} of the main rotor, the reaction force of the tail rotor is neglected). The reaction force of the main rotor must be compensated by the propulsation force of the tail rotor. There is also a gyroscopic effect caused by rotation of the main rotor which effects the additional torque around the horizontal axis which was neglected in our study. From this short discussion we conclude that we need a multivariable control approach as each control input influences both outputs (especially u_1).

The propulsation forces can be modelled mathematically using appropriate equations for interactions among the air, the shape and rotational speed of the rotor or as in our case they can be modelled experimentally. To model the forces we used experimentally obtained transfer function (7) of second order with double pole and the output quadratic nonlinearity (10)

$$\frac{U_m(s)}{U(s)} = \frac{1}{(Ts+1)^2} \qquad (7)$$

$$F_{prop} = a u_m^{\ 2} + b u_m \qquad (8)$$

In Equation (7) U is the voltage driving the rotor motor (U_1 or U_2) and F_{prop} in Equation (10) is the useful force of the rotor (F_{main} or F_{tail}). Parameters a and b are gained from the static characteristic.

We modelled the reaction force similar to the propulsion force except that we introduced more complex numerator dynamics (11)

$$\frac{U_m(s)}{U(s)} = \frac{T_{1r}s^2 + T_{2r}s + 1}{(Ts+1)^2} \qquad (9)$$

$$F_{react} = a_r u_m^{\ 2} + b_r u_m \qquad (10)$$

For the main rotor the parameters are: *a=2.488 N, b=0.992 N, T=0.094 s,* T_{1r}*=0.0017 s,* T_{2r}*=0.1908 s,* a_r*=0.296 N,* b_r*=0.2 N* and for tail rotor: *a=0.792 N, b=0.316 N, T=0.094s.*

The block diagram, which realizes the forces for the main rotor (Eqs. 7-10) is shown in Figure 6.

Figure 7 describes the model of the rotor. The part that is outside the encircled area is actually the part described in Figure 6 which calculates the forces. The encircled part is for the animation purposes. For this purpose we model a bearing and a rotor with no mass.

These parts do not affect the model dynamics. The rotational speed of the rotor is proportional to the voltage on the motor.

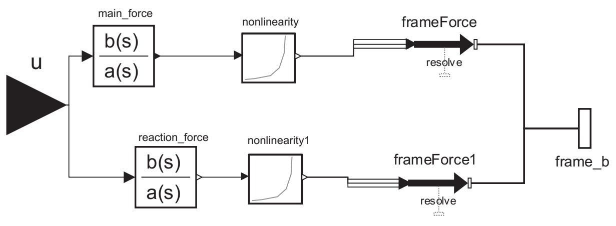

Figure 6: Block diagram that calculates two forces for the main rotor

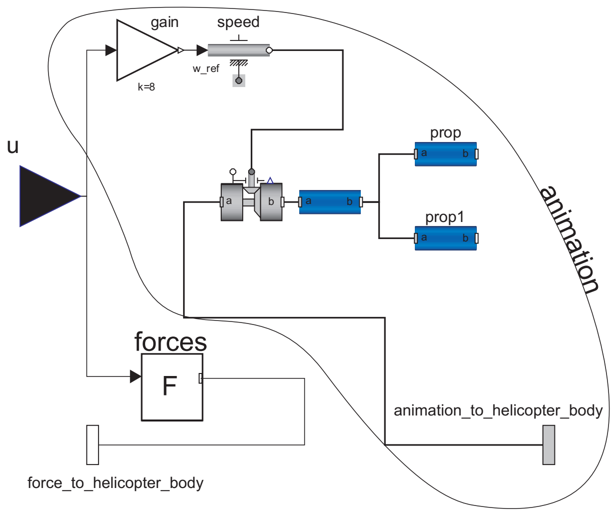

Figure 7: The model of rotors

Controller

The control system was designed using Matlab/Simulink environment. In Simulink the overall mechanical model was presented with the so called Dymola block. The simulation scheme is depicted in Figure 8. The input u_1 into the helicopter model is the voltage driving the front rotor and the input u_2 is the voltage driving the tail rotor. These are control signals. Output y_1 is the pitch angle and y_2 is the rotation angle. These are controlled signals.

Both control loops are combinations of the feedforward control and the PID feedback control with windup protection. Feedforward signals are obtained from the reference signals *Ref1* and *Ref2* which are processed with two nonlinear functions which are realized with two lookup tables. The feedforward signals bring the controlled signals into the vicinity of the reference signals. Fine corrections and steady state error eliminations are achieved with appropriately tuned PID controllers.

As the emphasize of this paper is given to the multi-domain object oriented modelling approach we shall not describe the controller tuning procedure and the appropriate parameters of the control system.

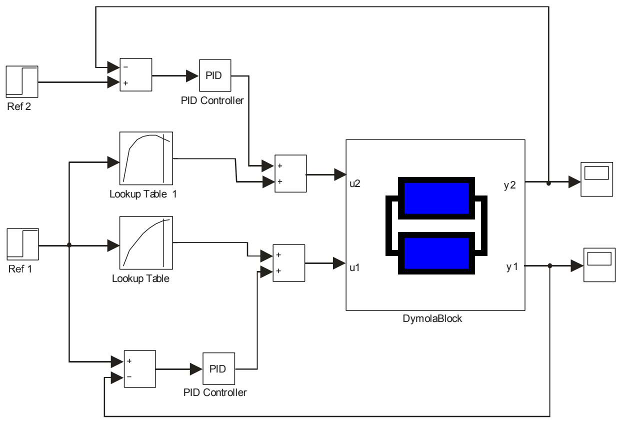

Figure 8: The control system in Matlab/Simulink environment

Figure 9 shows the time responses of the pitch and rotation signals. The reference of the pitch angle was chosen between -1.2 radians and 0.3 radians and the reference of the rotation angle was chosen between 0 an 3 radians. We can observe that the control system behaviour is satisfactory in spite of significant changes.

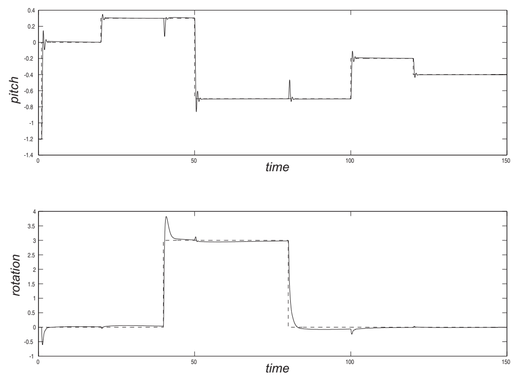

Figure 9: Controlled signals (pitch and rotation angles) and appropriate reference changes (dashed lines)

CONCLUSION

Modelling and simulation is extremely important subject in all control engineering courses. Namely it is much easier, cheaper and safer to experiment in simulation environment as on real processes. However traditional block oriented causal approaches should be extended by more advanced object oriented acausal approaches and tools which are especially efficient in multi-domain applications.

In Karer and Zupančič 2006 the traditional modelling approach using causal traditional block diagram approach was used. In this paper however a more advances multi-domain and object oriented acausal modelling was used. There is no important difference when modelling control schemes. However there is a big difference when modelling physical systems. The structure of the block diagram model in Matlab/Simulink does not reflect the topology of the physical system. It is not possible to transparently observe the stand, the helicopter body, the rotors, the revolute joints … There is a fundamental limitation of block diagram modelling. The blocks have an unidirectional data flow from inputs to outputs. This is the reason why objects mentioned above cannot be dealt with directly. It is also the reason why some parameters of the different components appear in mixed expressions in the Matlab/Simulink model. And the developed structure can be more or less used only for the configuration it was developed. The effort to produce the Simulink simulation model is incomparable with the Modelica model. One needs much more time and much more modelling knowledge for Simulink based approach. There is no useful textual layer behind the diagram layer which is the important disadvantage of the Simulink model. So it is problematic to deal with complex and sophisticated models. The documentation is very difficult and inappropriate.

In Modelica models (Figs. 3, 4, 5), the connections between physical system and computer model are very transparent. All the components are fully reusable in other configurations. The combination of textual and diagram programming is efficient. So Modelica can be used for very complex problems and is also superior for model documentation.

We conclude that Matlab/Simulink is superior for the design and implementation of control schemes as it has more facilities especially in conjunction with some toolboxes – e.g. Control System Toolbox, Optimization Toolbox. Modelica is superior when modelling physical systems when the concept of algebraic manipulation and specially defined connectors bring many advantages. This approach is also very useful in education. We propose to start modelling courses with OO acausal approach especially when one can deal with implemented libraries which do not demand a deep theoretical background. And it can motivate students much more than the low level Simulink approach.

AUTHOR BIOGRAPHY

DEJAN DOVŽAN received the university degree from electrical engineering at the University of Ljubljana, Slovenia in 2008. Currently he is the Ph.D. student. The main research area is: intelligent methods in modelling and control.
http://msc.fe.uni-lj.si/Staff.asp

BORUT ZUPANČIČ received Ph.D. in electrical engineering from the University of Ljubljana, Slovenia. He is full professor from 2000. His major research interests are: object oriented modelling, hybrid systems, modelling, simulation and control of conditions in buildings. He was the president of EUROSIM (Federation of European Simulation Societies) in the period 2004-07. He is author of app. 200 conference papers and 40 papers in scientific journals, co-author of one international book (published by Elsevier) from the area of modelling and simulation.

GORAZD KARER received the university degree from electrical engineering at the University of Ljubljana, Slovenia in 2004. Currently he is the Ph.D. student. His research interests are in hybrid systems and model predictive control. He is author of 6 papers in scientific journals.

IGOR ŠKRJANC received Ph.D. in electrical engineering from the University of Ljubljana, Slovenia, in 1996 and became full professor in 2008. His main research interests are: adaptive, predictive, fuzzy, and fuzzy adaptive control systems. He is author of app. 150 conference papers and 65 papers in scientific journals.

REFERENCES

Van Beek, D.A., J.E. Rooda. 2000. "Multi-domain modelling, simulation, and control". *Proceedings of 4th International Conference on Automation of Mixed Processes: Hybrid Dynamical Systems* (ADPM2000), Dortmund, pp. 139-146.

Cellier, F.E. 1991. "*Continuous System Modeling.*" Springer Verlag, New York.

Dymola, Multi-Engineering Modelling and Simulation. 2008. Users manual, Ver. 7.0, Dessault System, Dynasim AB, Sweden, Lund.

Fritzson, P. 2004. "*Principles of Object Oriented Modelling and Simulation with Modelica 2.1*", IEEE Press, John Wiley&Sons Inc., Publication, USA.

Humusoft, CE 150 Helicopter Model. 2002. User's manual, Humusoft, Prague.

Karer, G, Zupančič B. 2006. "Modelling and Identification of a Laboratory Helicopter". *Proceedings of the 5th MATHMOD conference*. Vienna, vol. 2, 9 pages.

Matko, D., R. Karba and B. Zupančič. 1992. "*Simulation and Modelling of Continuous Systems: A Case Study Approach*". Prentice Hall Int., New York.

Modelica - A Unified Object-Oriented Language for Physical System Modelling. Feb. 2005. Language specification, Ver. 2.2, Modelica Association, http://www.modelica.org/documents /ModelicaSpec22.pdf

Strauss, J.C. 1967. "The SCi Continuous System Simulation Language". *Simulation*, no.9, 281-303.

Zupančič, B., G. Zauner, F. Breitenecker. 2005. "Modeling and Simulation in Process Technology with Modelica", *Proceedings of the 2005 Summer Computer Simulation Conference*, Cherry Hill, NJ.

Zupančič, B., A. Sodja. 2008. "Object Oriented Modelling of Variable Envelope Properties in Buildings", *WSEAS Trans. on Systems and Control*, Issue 12, Volume 3. http://www.wseas.us/ e-library/transactions/control/2008/28-731.pdf