Hardware-in-the-Loop Simulation and Control for Developing Very Large Wind Energy Systems

Rami Basilios* and Adrian Gambier*

**Fraunhofer IWES, Fraunhofer Institute for Wind Energy Systems, 27572 Bremerhaven, Germany (Tel: +49 471 14290-375; e-mail: adrian.gambier@iwes.fraunhofer.de)*

Work financed by Federal Ministry of Economic Affairs and Energy (BMWi)

Abstract: Nowadays, simulation is a very important tool in order to design control systems of large wind turbines due to the fact that large complex wind turbines are not available as experimental set-up and down scaled systems show a completely different behaviour as the large ones. On the other hand, digital simulation is not enough to test control algorithms since the analysis of controllers in a real-time environment is essential. Hence, the combination of wind turbine simulation and direct digital real-time control becomes significant and this leads to the concept of Hardware-in-the-Loop (HiL) simulation and control. The present contribution proposes a Hardware-in-the-Loop configuration for the real-time simulation and control of large-sized wind turbines, where a well-known simulation tool is integrated with a control hardware that is often used in real wind turbines. Software and hardware choices are analysed, the implemented architecture is described and satisfactory results of a numerical experiment based on a 20 MW wind turbine is presented.

Keywords: Large wind turbines, control of wind turbines, real-time simulation, Hardware-in-the-Loop

1. INTRODUCTION

The basic idea of Hardware-in-the-Loop simulation is to test control algorithms in an environment where not only purely mathematical models of components are used but also real physical components are embedded in the control loop. This concept provides a more realistic testing infrastructure, which is particular useful when these components are difficult to be accurately modelled or when it is impossible to reckon with a whole real system but some real components can be tested in a more general context. In addition, HiL simulation requires that the control system as well as the simulation performs under real-time conditions.

HiL simulations have been used for a very long time, for example in flight simulators (e.g. Evans and Schilling, (1984), Bailey and Doerr, (1996)), in the automotive industry (see Hanselmann, (1996), Kiffmeier, (1996)) and in power systems (as Viehweider et al., (2011), Faruque and Dinavahi, (2010)). HiL simulators have also been used to study power systems (grid phenomena, Roscoe et al., (2010), Viehweider et al., (2011), as well as wind energy systems from the electrical point of view, Steurer et al., (2004). More recently, HiL simulators for the study of nacelle test benches have been reported in Neshati et al., (2016) as well as in Leisten et al., (2017).

Regarding wind turbines, it is important to remark, that the increasing size and complexity of such systems requires more sophisticated control approaches. Due to the fact that large complex wind turbines are not available for testing and experiments as well as small scaled systems behave dynamically in a completely different manner, it is necessary to appeal to realistic simulation tools, like HiL simulators.

This is the main contribution of the present work, where the focus is set in the real-time testing of advanced control algorithms in a dedicated hardware for wind turbines taking into account high resolution dynamic models. The paper is organized as follows: In Section 2, concepts of simulation, real-time and Hardware-in-the-Loop are introduced according to the definitions used in the work. Section 3 is devoted to describe the used architecture including hardware and software. A numerical example is presented in Section 4 and simulation results are shown in Section 5. Finally, conclusions are drawn in Section 6.

2. REAL TIME SYSTEMS AND HARDWARE IN THE LOOP

Real time, modelling, simulation and hardware in the loop are much disseminated concepts that take different meanings and definitions depending on the used context and discipline. Therefore, meaning and concepts concerning to this work are introduced and clarified in the following in order to avoid misunderstandings and confusions.

2.1 Modelling and Simulation

In the sense of this work, a model is an abstract representation of a physical system or component concretized by mathematical equations. In particular, dynamic systems are modelled, in the case of systems with only continuous states, by a set of algebraic-differential equations, by a state machine (or other discrete formalism) in the case of only discrete states and by hybrid formalisms, like hybrid automata, in the case of systems with continuous and discrete states. Notice that time-discrete systems are also continuous systems represented by algebraic-difference equations.

It is important to remark that sometimes software implementations of mathematical models is also called "model". For instance, a Simulink-block-diagram implementing several physical components could also be named a "model" of these components. However, this meaning is not used in the present work.

Differential equations used for the modelling of a dynamic system can be linear, nonlinear and with partial derivatives. Wind energy systems include all these classes.

Simulation means here the numerical solution of a set of equations by using a computer program, which are representing a dynamic model of a real system. This program requires a “solver”, i.e., a software implementation of a numerical algorithm to solve differential equations as well as algebraic loops.

2.2 Real-time Control and Simulation

Several definitions of real-time systems can be found in the literature. The definition that accentuates the fact that time is a very important variable in the system, where timing constraints are associated with system tasks, is adopted here as

A real-time system is characterized by the fact that the correctness of a result depends not only on the logical correctness of the calculation but also upon the meeting of the previous defined deadline at which the result has to be made available.

The system tasks mentioned above are normally involved in the control activity or in the reaction to events that take place in the external world happening in “real time”. Hence, real-time tasks must be able to respond deterministically to scheduled tasks requests as well as to internal and external events, with which it is concerned (w.r.t Gambier, (2004)). Thus, the deterministic system response is related to time constraints associated to tasks in the form of deadlines, which have to be strictly met.

The implementation of multitasking real-time systems can be done from two different points of view: one approach is based on programming by using a concurrent real-time language, like Ada, and the other one is implemented by using a standard programming language and the real-time services are delegated to a *real-time operating system* (RTOS, Burns and Wellings, (2009)). Notice that RTOS and real-time systems are not equal concepts: A RTOS provides facilities, like multitasking (*i.e. concurrency* and *parallelism*), scheduling, inter-task communication mechanisms, etc., for implementing real-time systems. Well-known RTOS are for example QNX, Kim et al., (2010), VxWorks, Liu et al., (2017), Wind River Linux (also known as RT-Linux) and LynxOS, Garcia, (2017). On the other hand, there exist several custom dedicated implementations of real-time monitors in order to satisfy real-time requirements of particular hardware. This is the case, for instance, of manufacturers of PLC (Programming Logic Controllers) and Smartphones.

The concept of real-time changes slightly if it is applied to simulation. In Isermann et al., (1999), real-time simulation is defined as a simulation where input and output signals have the same time dependence as the real running system. However, this definition involves the previous one of real-time system because in order to satisfy the time dependence, the simulation has to be executed inside a real-time environment, i.e. the dynamic of the real system determines the maximum integration time-step for the solver and the solver is executed in a task with a deadline set at the end of the given integration step. Herewith, the synchronization between real time and simulation time is obtained.

In order to guarantee determinism, the integration step has to be fix in order to avoid iterations and recalculations that normally take place in the algorithms with adaptive integration steps making unpredictable the whole computational time.

On the other hand, the time-step should be smaller or even negligible compared to the system dynamics, i.e. the maximum natural frequency, in order to satisfy stability conditions Khaled-El Feki, (2014). A particular problem appears when fast and slow transients are mixed in the model. In such a case, the step-size has to be chosen small enough to capture the fast behaviour but it will increase the computational burden during the integration of slow dynamics. In Balla, (2011), the time-step is chosen according to

$$h = 1/(\pi f_{\max}), \tag{1}$$

where $f_{\max} = \omega_{\max}/(2\pi)$ and $\omega_{\max}$ is the undamped maximum natural frequency of the system.

Since the real-time simulation is combined here with the real-time control, the time-step has to be in compliance with the sampling time. According to Shannon’s theorem, the theoretic sampling time must be smaller than $1/(2\,f_{\max})$. However, for practical applications is well-know that the sampling time should be (see e.g. Åström and Wittenmark, (1997))

$$T_s = 1/(\alpha f_{\max}), \tag{2}$$

where α can be chosen, for example, as 5 or 10. Thus, the time-step should be smaller than the sampling time in order to assume the time as “continuous”. Accepting that the solver should yield several values of the solution in a sampling period, the time-step should be

$$h = \gamma\, T_s = \gamma/(\alpha f_{\max}), \tag{3}$$

where γ is often select as 0.1. Hence, the fix integration time-step should be about (α/γ) times smaller (e.g. 100) than the smallest time constant of the simulating dynamic system in order to maintain the emulation of the continuous time. This value is however very small and leads to a very high computational load. Since this is actually empiric and depends on the application, it is possible to optimize the time-step in order to find a compromise between stability, accuracy and time consumption.

Notice that dynamic errors produced by the solver are proportional to h^n, i.e. $O(h^n)$, where n is the order of the numerical integration algorithms. One of the most popular algorithms is the predictor-corrector fourth-order Adams-Bashforth-Moulton method (ABM4). This method requires the calculation of the derivatives in step n+1 with the inputs $\mathbf{u}_{n+1}$, which are not available in the case of real-time operation, Howe, (1989). An algorithm without this limitation is the Adams-Bashforth method (AB4), Howe, (1991). In addition, the ABM method have been adapted for real-time applications in Howe, (1989).

A similar analysis is given in the literature also for Runge-Kutta based algorithms. Hence, it is important to be careful in the selection of numerical integration algorithms, if they should be used in a real-time environment.

Finally, an AB4 algorithm will produce errors in the order of

$$O(h^4) = \gamma^4/(\alpha^4 f_{\max}^4), \tag{4}$$

where $\pi \leq \gamma/\alpha \leq 100$ is an empiric value. Thus, from (1) and (2) it follows

$$10^{-8}/f_{\max}^4 \leq O(h^4) \leq 10^{-2}/f_{\max}^4\,. \tag{5}$$

In this way, a compromise value for α can be chosen such that for a given maximum natural frequency an accepted error is obtained.

2.2 Hardware in the Loop

There are several ways to interpret the idea of Hardware-in-the-Loop. A detailed general description about the topic can be found in Sarhadi and Yousefpour, (2015), Bacic, (2005), Bélanger et al., (2010). In the following, only the concept as used in this work is presented. The start point is a real computer-controlled system as shown in Fig. 1, i.e., a real plant, a controller implemented on a dedicated computer and the interfaces. Notice that wide arrows are used here to represent digital data with parallel representation of bits and thin arrows symbolize analog signals.

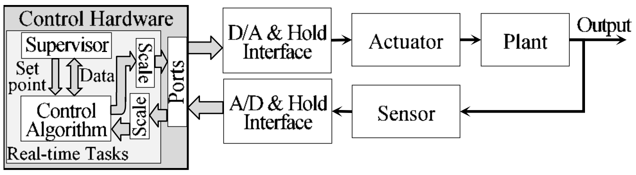

Fig. 1. Scheme of a real control loop

If the control loop of Fig. 1 is completely implemented in a digital computer for simulation purposes, the block diagram looks like Fig. 2.

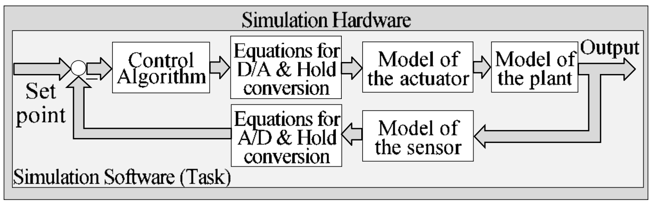

Fig. 2. Scheme of a simulated control loop

The chain actuator-plant-sensor of Fig. 1 can be replaced by the corresponding models of Fig. 2 such that Fig. 1 becomes a Software-in-the-Loop scheme. On the other hand, if all blocks associated with the control system of Fig. 2 are replaced by the control hardware of Fig. 1, the result is a Hardware-in-the-Loop scheme. Notice that both obtained schemes result in an identic configuration and its name depends on the point of view (see Fig. 3).

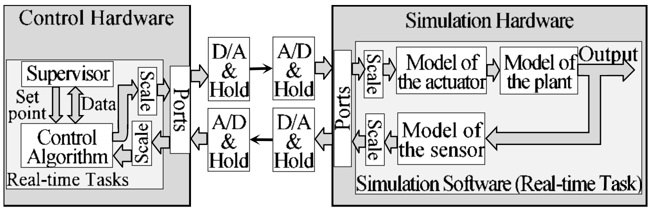

Fig. 3. Hardware-in-the-Loop control system configuration

It is important to remark that due to the fact that the control hardware and the simulation system run in different machines, both has to be synchronized. Hence, the simulation must run inside a real time task. This can be named real-time simulation and therefore the algorithm for the numerical integration should satisfy the conditions described subsection 2.2. This aspect is treated in details in the next section.

If only the model of a component, as e.g. the actuator, is replaced by a real actuator in Fig. 2, then it is a clear case of HiL. However, the software has to manage the actuator power and in this case the scheme is called sometimes Power Hardware-in-the-Loop (PHiL, see e.g. Bouscayrol, (2008)) and an example of this is given in Fig. 4.

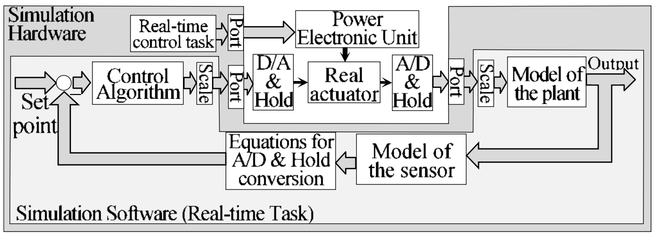

Fig. 4. Example of a Power Hardware-in-the-Loop configuration

In the present work, the configuration used is that given in Fig. 3 and it will be called Hardware-in-the Loop because the maximum interest is to study the control hardware, the control algorithms and the real-time problem. A configuration like Fig. 4 is planned for the future.

3. HARDWARE-IN-THE-LOOP ARCHITECTURE

Following the concepts introduced in the previous section, the implemented architecture and the simulation environment are described in the following.

3.1 System Architecture

The general system architecture consists of a simulation workstation with interfaces and real time capacity, where the aeroelastic code simulating the wind turbine dynamic behaviour runs, and a distributed hardware for the implementation of the control system. The scheme is illustrated in Fig. 5.

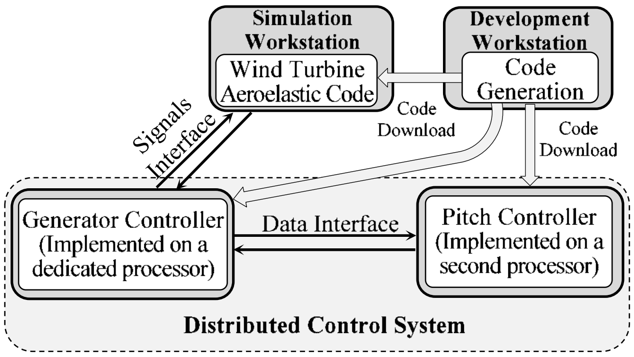
Simulation Workstation
Wind Turbine Aeroelastic Code
Development Workstation
Code Generation
Code Download
Code Download
Signals Interface
Generator Controller (Implemented on a dedicated processor)
Data Interface
Pitch Controller (Implemented on a second processor)
Distributed Control System

Fig. 5. Proposed general HiL architecture for the large wind turbine

3.2 Software Implementation

The simulation of wind turbines in time domain can be undertaken by using different software tools. Thus, commercial software are, for example, HAWC2 from the Denmark Technical University, Larsen and Hansen, (2014), Bladed from DNV GL, Bossanyi, (2003), and Cp-Lambda from Politecnico di Milano, Bottasso and Croce, (2009). Open source tools are for instance QBlade from the Technical University of Berlin, Pechlivanoglou et al., (2010) and FAST from National Renewable Energy Laboratory, Jonkman and Buhl Jr., (2005).

FAST is provided including the source code and therefore is attractive for the current implementation. In particular, it is useful the provided Matlab/Simulink interface because tools like Simulink Coder, Simulink Real Time and Simulink Desktop Real-Time can be used to execute FAST in a soft real time task with integration steps up to 1 ms. In addition, an AB4 algorithm is implemented as solver. Thus, the wind turbine can be simulated in a real time like environment despite the time constraints imposed to the simulation conditions.

3.2 Hardware Implementation

The necessary hardware consists basically of at least two computational units, one to host the real-time simulation and the other one as digital controller, and the corresponding interfaces. As simulator a Windows® Workstation with an Intel® Core™ i7 Extreme edition processor is used. There are many interface cards in the market. However, the necessary conditions for this configuration are: to support by the Simulink Desktop Target and a high number of A/D, D/A and I/O channels. The final decision felt upon two card MF634 from Humusoft® completing 16 channels of each type.

Although there exist many options for the control hardware, three requirements are decisive for the selection: support of Simulink development based on a blockset and code generation for the target, standard hardware used in real wind turbines and well-known hard real-time operating systems (HRTOS). All these prerequisites are satisfied by the M1 industrial platform of Bachmann®, which consists of several modules (e.g. MC210 and MX213 as computational units and AIO216 and DIO248 for the interface channels). As HRTOS, VxWorks from Wind River is used.

Finally, signal conditioning adapters are necessary to connect the TLL signal level of the MF634 and the ±24V of the Bachmann modules.

3.3 Final Implemented Configuration

The final configuration is shown in Fig. 6.

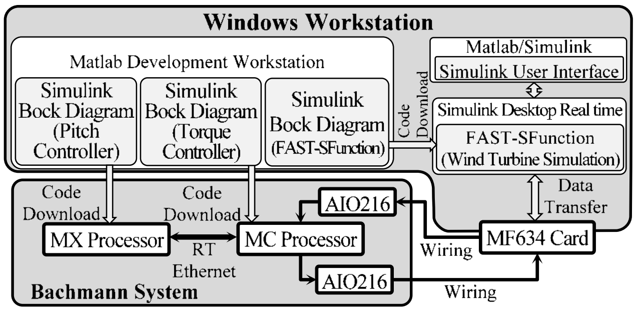
Windows Workstation
Matlab Development Workstation
Simulink Bock Diagram (Pitch Controller)
Simulink Bock Diagram (Torque Controller)
Simulink Bock Diagram (FAST-SFunction)
Code Download
Matlab/Simulink
Simulink User Interface
Simulink Desktop Real time
FAST-SFunction (Wind Turbine Simulation)
Data Transfer
Code Download
Code Download
MX Processor
RT Ethernet
MC Processor
AIO216
AIO216
Wiring
MF634 Card
Wiring
Bachmann System

Fig. 6. Implementation of the HiL architecture

3.4 Task Synchronization

As it is previously mentioned simulation and control tasks have to be synchronized. This is not difficult to do on the control hardware because VxWorks provides facilities for this. On the simulation side, the synchronization is not simple because the running software does not foresee methods for real-time operation. In a first approach, the problem is solved by using the simulation time, that FAST provides as output variable. This variable is a monotonic increasing piecewise constant signal, whose constant period is equal to the integral time-step. The signal is then passed by an edge detector in order to generate a pulse train, which, in turn, is transferred to the real-time control hardware as wake-up signal in order to activate the waiting control task. The procedure is illustrated in Fig. 7.

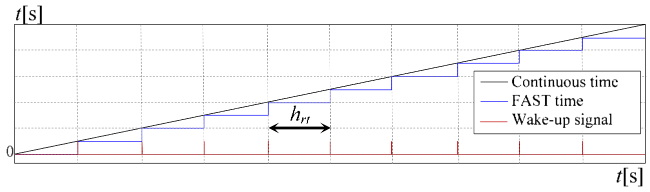

Fig. 7. Synchronization procedure for the HiL system

The necessary real time required by the solver to compute the solution of an integration time-step h_{FAST} is noted as h_{rt}. In general, the real-time system must satisfy $h_{rt} < h_{FAST}$.

Notice that the solver must reckon with the input signals at the begin of the integration step, which are maintained constant from time-step to time-step until the sampling period finishes and new values of the input signals are required at the begin of the next sampling period. Therefore, it is essential that the whole sequence of time-steps finishes inside the sampling time but letting an enough free margin such that the controller can compute the next values of the control signals before the next simulation step begin. On the other hand, the values of the output signals are delivered by the solver at the end of $m\,h_{rt}$ steps. Hence, these output values are already available for the control signal calculation when the control tasks are triggered by the wake-up signal. Control tasks have to deliver the control signals in a time defined by T_s - $m\ h_{rt}$. This analysis is summarized in Fig. 8.

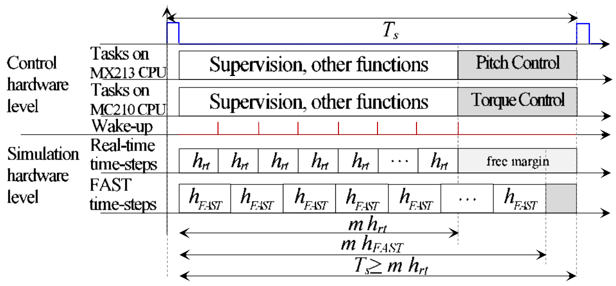

Fig. 8. Illustrative time sequences of the HiL system

3.5 Pitch Control System

At present, only a collective pitch control system (CPC) has been implemented. It is based on a PID controller with an anti-windup mechanism for magnitude and rate saturations with back calculation formulated as

$$U(s) = K_p E(s) + \frac{1}{s}\left[K_i E(s) - K_a[U(s) - U_a(s)]\right] + \frac{K_d\, s}{1 + T_d\, s} E(s) \qquad (6)$$

and illustrated in Fig. 9. The controller is implemented as a time-discrete system for the defined sampling time.

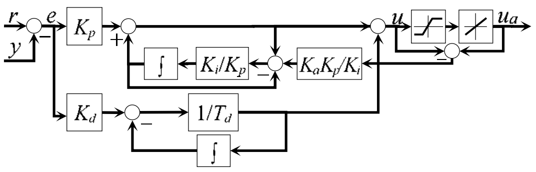

Fig. 9. Automatic reset configuration with magnitude and rate anti-windup with back calculation

3.6 Torque Control System

In Region II, i.e. where wind speed in under rated, the optimal tracking control is obtained by manipulating the electromagnetic torque. For this work, the very simple control law given by

$$T_g = K_1 \omega_g^2 + K_2 \dot{\omega}_g \qquad (7)$$

is used, where K_1 is selected for maximum power extraction and K_2 for rotor inertia reduction (see Gambier and Meng, (2019)).

4. NUMERICAL EXAMPLE

The Hardware-in-the-Loop architecture has been tested with a numerical example based on a reference wind turbine. These aspects are presented in the following subsections.

4.1 Description of the Wind Turbine

The 20 MW reference wind turbine used here is proposed first in Ashuri et al., (2016). This is a conventional three-bladed, horizontal axis, clockwise, upwind, variable-speed and variable-pitch machine. The rotor has a diameter of 276 m, where the blades have a length of 135 m and the hub a diameter of 6 m. Each blade has a mass of 259 tonnes and is divided in 20 sections with six different airfoils. The maximum chord of the blade is 10 m and the tip deflection in the fore-aft direction is 18.1 m. Finally, the whole rotor has a mass of 839.3 tonnes and a second mass moment of inertia of 2.92 $\times 10^9$ kg m^2.

The nacelle has a mass of 252.8 tonnes and it is mounted on a tower of 160.2 m hub height and 10 m diameter on the bottom as well as 6.2 m on the top. In addition, the tower consists of 22 sections. The first natural frequency of the tower is 0.1561 Hz.

The drive train consists of a low speed shaft of 159.1 tonnes, a gearbox of 161.9 tonnes and a ratio of 164:1. The high-speed shaft and the generator adds a mass of 59.8 tonnes. The drive train is characterized by an equivalent spring constant of 6.94$\times 10^9$ Nm/rad and a damping constant of 4.97 $\times 10^7$ Nm/(rad/s).

For a generator efficiency of 94.4%, a rated electrical power of 20 MW corresponds a rated mechanical power of 21.19 MW. The maximum power factor $C_{p,max}$ = 0.47268 is reached at a tip-speed ratio of 9.51. The gearbox ratio yields a rated generator speed of 1173.7 rpm for a rated rotor speed of 7.16 rpm, such that for the rated wind speed of 10.715 m/s is obtained.

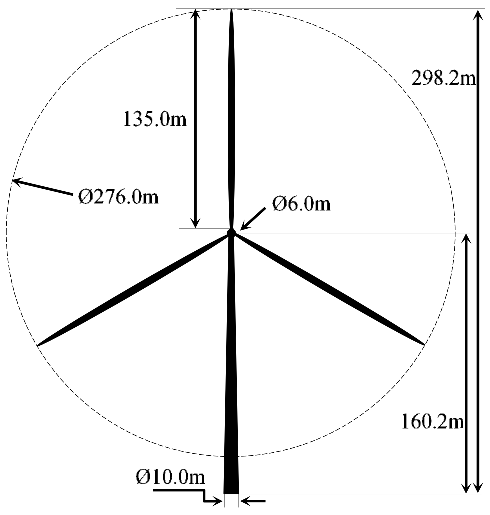

Fig. 10. Descriptive scheme of the 20 MW wind turbine

The reference turbine has 16 DOF (degrees of freedom), i.e. 32th order, and the natural frequencies are in the range of

$$0.0174 \le \omega_n \le 11.7596 . \qquad (8)$$

The maximum natural frequency is given by f_{max}=11.7596/(2π) (f_{max}=1.8716) such that the sampling time is T_s = 0.05 sec. Accepting an integration time-step of h_{FAST} = 0.0125, the value for γ (according to (3)) is 0.25. Thus, four integration time-steps can be completed in a sampling period. Since the real computational time for an integration time-step is 2.6 ms (h_{rt} = 0.0026) 42.6 ms are available for the control signal calculation. Errors of the AB4 algorithm are in the order of $O(h^4)$ = 3.18 10^{-8} (w.r.t. eq. (4)).

A control system design for this reference turbine is proposed in Gambier and Meng, (2019).

4.2 Experimental Setup

For this example, the wind turbine is operated in Region III for an effective wind speed varying between 11 and 25 m/s including tower shadow and variable turbulences between 5 and 20%. The profile is shown in Fig. 11.

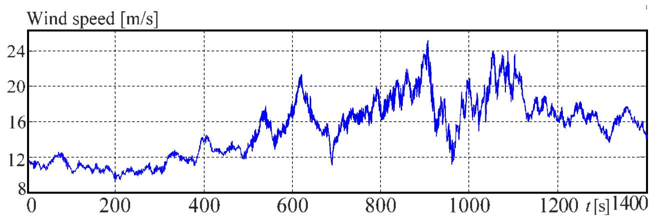

Fig. 11. Wind speed profile for simulation purposes in FAST

The simulation time was set to 1400 sec (23.3 min).

5. SIMULATION RESULTS

The main objective of this simulation experiment is to verify that the implemented HiL system work correctly. That is, the simulation of a large wind turbine by using a high-resolution model can run in a real-time environment, the communication between simulation hardware and control hardware is correct and maintains the real-time conditions and the available time on the control hardware is enough to compute the control signals. This can be observed in the simulation results (Fig. 12 and Fig. 13). This is so because on the contrary the large machine behaves with very large oscillations or even becomes unstable.

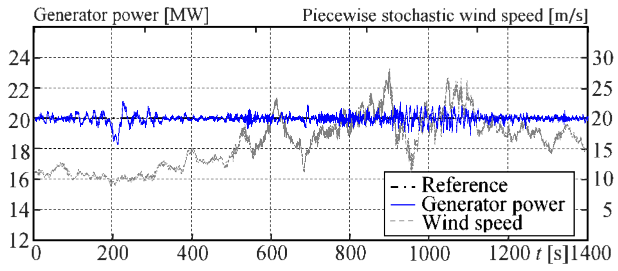

Fig. 12. Generator power maintained constant at 20 MW

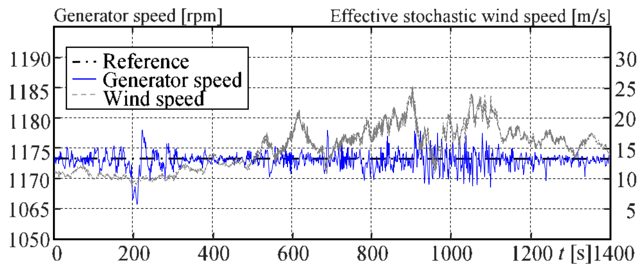

Fig. 13. Generator speed maintained constant at 1173.7 rpm under stochastic wind

Torque control and pitch control are designed for the above-mentioned objective and are still able to be optimized and improved. Additional control loops as the active tower damping control (ADTC) as used in Gambier, (2017) as well as individual pitch control implemented in Behera and Gambier, (2018) can be added in a near future.

Finally, the control signals computed in the MX213 CPU with rate limited anti windup are transferred to the simulation hardware in order to maintain constant the power when the wind speed is over rated. This signal is presented in Fig. 14.

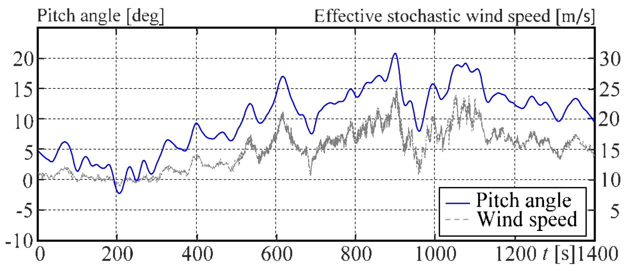

Fig. 14. Pitch angle signal provided by the controller

7. CONCLUSIONS

In this work, a Hardware-in-the-Loop architecture for the real-time simulation and control of wind turbines combined with a typical control hardware, which is often used in wind energy systems, is presented. Software characteristics and numerical aspects related to the solver implemented in FAST are analysed from the execution point of view under time-constraints.

It could be verified that the implemented system satisfied all requirements. The distributed real-time control system reduces, on one hand, the computational load in the calculation of the control signals allowing more sophisticated control algorithms and, on the other hand, introduces hardware redundancy such that reconfiguration and fault-tolerant control approaches can be studied in a real-time environment.

REFERENCES

Ashuri, T., Martins, J. R. R. A., Zaaijer, M. B., van Kuik, G. A. M. and van Bussel, G. J. W., (2016), 'Aeroservoelastic design definition of a 20 MW common research wind turbine model', *Wind Energy*, vol. 19, no. 11, pp. 2071-2087.

Åström, K. and Wittenmark, B., (1997), *Computer controlled systems*, 2nd edn, Prentice Hall International.

Bacic, M., (2005), 'On hardware-in-the-loop simulation', *44th IEEE Conference on Decision and Control and European Control Conference 2005*, IEEE, Seville.

Bailey, M. and Doerr, J., (1996), 'Contributions of hardware-in-the-loop simulations to Navy test and evaluation', *Proceedings of the Society of Photo-optical Instrumentation Engineers*, vol. 2741, pp. 33-43.

Balla, J., (2011), ' “Dynamics of mounted automatic cannon on track vehicle”, ', *International Journal ofMathematicalmodels and methods in applied sciences*, vol. 5, pp. 423-432.

Behera, A. and Gambier, A., (2018), 'Integrated pitch control system design of a wind turbine by using multi-objective optimization', *IFAC PaperOnLine*, vol. 51, no. 28, pp. 1033–1039.

Bélanger, J., Venne, P. and Paquin, J.-N., (2010), 'The what, where and why of real-time simulation', *PES General Meeting*, IEEE.

Bossanyi, E. A., (2003), 'GH Bladed, Version 3.6 User Manual', Garrad Hassan & Partners Limited, Bristol.

Bottasso, C. L. and Croce, A., (2009), 'Cp-Lambda user manual', Dipartimento di Ingnegneria Aerospaziale, Politecnico di Milano, Milano.

Bouscayrol, A., (2008), 'Different types of Hardware-In-the-Loop simulation for electric drives', *IEEE International Symposium on Industrial Electronics*, IEEE, Cambridge.

Burns, A. and Wellings, A., (2009), *Real-time systems and programming languages*, Fourth Edition edn, Addison Wesley, Essex.

Evans, M. B. and Schilling, L. J., (1984), 'The role of simulation in the development and flight test of the HiMAT vehicle', NASA, NASA, Hampton.

Faruque, M. O. and Dinavahi, V., (2010), 'Hardware-in-the-loop simulation of power electronic systems using adaptive discretization', *IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1146-1158.

Gambier, A., (2004), 'Real-time control systems: a tutorial', *Asian Control Conference*, ACA, Melbourne.

Gambier, A., (2017), 'Simultaneous design of pitch control and active tower damping of a wind turbine by using multi-objective optimization', IEEE, Kohala Coast.

Gambier, A. and Meng, F., (2019), 'Control system design for a 20 MW reference wind turbine', IEEE, Hong Kong.

Garcia, L. W., (2017), 'Real-time operating systems. Case study: LynxOS vs. VxWorks', Florida Atlantic University, Boca Raton.

Hanselmann, H., (1996), 'Hardware-in-the-Loop simulation testing and its integration into a CACSD toolset', *1996 IEEE International Symposium on Computer-Aided Control System Design*, IEEE, Dearborn.

Howe, R. M., (1989), 'An improved numerical integration method for flight simulation', *AIAA Flight Simulation Technologies Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Washington D.C.

Howe, R. M., (1991), 'A new family of real-time redictor-corrector integration algorithms', *Simulation*, pp. 177-186.

Isermann, R., Schaffnit, J. and Sinsel, S., (1999), 'Hardware-in-the-loop simulation for the design and testing of engine-control systems', *Control Engineering Practice*, vol. 7, no. 5, pp. 643-653.

Jonkman, J. M. and Buhl Jr., L. M., (2005), 'FAST User's Guide', NREL, Battelle.

Khaled-El Feki, A. B., (2014), 'Distributed real-time simulation of numerical models: application to power-train', Université de Grenoble, NNT: 2014GRENT033, Université de Grenoble, Grenoble.

Kiffmeier, U., (1996), 'A Hardware-in-the-Loop testbench for ABS controllers', S.G.E., Genoa.

Kim, J. Y., Lee, Y. J., Cheon, S. W., Lee, J. S. and Kwon, K. C., (2010), 'A Commercial-off-the-shelf(COTS) dedication of a QNX real time operating system (RTOS)', IEEE, Mumbai.

Larsen, T. J. and Hansen, A. M., (2014), 'How 2 HAWC2, the user‘s manual, v.4.5', Risø, Roskilde.

Leisten, C., Jassmann, U., Balshüsemann, J., Hakenberg, M. and Abel, D., (2017), 'Design and analysis of a MPC-based mechanical hardware-in-the-loop system for full-scale wind turbine system test benches', *IFAC-PapersOnLine*, vol. 50, pp. 10985–10991.

Liu, J., Gao, X., Jiang, B., Yang, S. and Zhang, Z., (2017), 'Deterministic replay for multi-core VxWorks applications', IEEE, Beijing.

Neshati, M., Zuga, A., Jersch, T. and Wenske, J., (2016), 'Hardware-in-the-loop drive train control for realistic emulation of rotor torque in a full-scale wind turbine nacelle test rig', *2016 European Control Conference*, EUCA, Aalborg.

Pechlivanoglou, G., Marten, D., Nayeri, C. N. and Paschereit, C. O., (2010), 'Integration of a wind turbine blade design tool in xfoil/xflr5', DEWEK, Bremen.

Roscoe, A. J., Mackay, A., Burt, G. M. and McDonald, J. R., (2010), 'Architecture of a network-in-the-loop environment for characterizing AC power-system behavior', *IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1245-1253.

Sarhadi, P. and Yousefpour, S., (2015), 'State of the art: hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software', *International Journal on Dynamics and Control*, vol. 3, pp. 470-479.

Steurer, M., Li, H., Woodruff, S., Shi, K. and Zhang, D., (2004), 'Development of a unified design, test, and research platform for wind energy systems based on hardware-in-the-loop real-time simulation', *IEEE Power Electronics Specialists Conference*, IEEE, Aachen.

Viehweider, A., Lauss, G. and Lehfuss, F., (2011), 'Stabilization of power hardware-in-the-loop simulations of electric energy systems', *Simulation Modelling Practice and Theory*, vol. 19, no. 7, pp. 1699-1708.

Viehweider, A., Lehfuss, F. and Lauss, G., (2011), 'Power hardware-in the-loop simulations for distributed generation', *International Conference on Electricity Distribution*, CIRED, Frankfort.