Implementing the wind speed estimation algorithm in the target embedded control system

Ivana Ivanović, Mario Barišić, Zlatka Tečec Ribarić
Power Electronics and Control Department
KONČAR - Electrical Engineering Institute
Zagreb, Croatia
ivanovic@koncar-institut.hr, mbarisic@koncar-institut.hr, ztecec@koncar-institut.hr

***Abstract*—To improve power production, wind turbine control system use information regarding the wind speed. Unfortunately, wind speed can't be enough accurately measured and measurement is not a good representative of the rotor effective wind speed. Therefore, wind speed estimation is necessary. This paper describes an algorithm for the estimation of the effective wind speed, based on the aerodynamic torque. It is assumed that the rotor speed and generator torque are measured. Wind speed enters into the generator dynamics through a highly nonlinear function, so there is a difficult problem of estimation of a nonlinearly parameterized system. To solve this problem secant method proved to be suitable. The algorithm is implemented in a conventional wind turbine control system, in the processor module, which contains a 32-bit microcontroller. The algorithm is implemented in fixed-point arithmetic using 16 bits with sign and has been validated by comparison with the Matlab simulation results. The benefit of this estimation algorithm is reflected in the anemometer fault detection. Therefore, the system availability is increased up to 0.5-1%.**

Keywords*— wind turbine control; wind speed estimation; fault detection; embedded control systems

I. INTRODUCTION

Wind turbines are very complex systems that are expected to work autonomously and to produce energy in a very different operating conditions defined primarily by wind speed. Therefore, control systems of modern wind turbines have to be very sophisticated. Control systems are realized digitally, using microcontrollers, PLCs or industrial PCs. Wind turbine control system must perform the following functions: control, sequence control, diagnostic, protection, chronological recording and archiving of process values and discrete system events, communication with a remote control centers.

Control tasks of wind turbine control system are to maintain process variables in desired range. Regulated process variables are: generator speed, nacelle direction, temperature and hydraulic pressure.

In order to achieve process variables regulation and wind turbine sequence control, control system has to manage the following subsystems of wind turbines: pitch system, frequency converter, excitation, power grid connection modes, rotor brake and hydraulic brake system, lubrication system and light warning obstruction.

The most important process variables are rotor speed and generator power. Controlling these values presents central, safety critical part of the system [1], [2], [3]. Other wind turbine operating variables are controlled by relatively simple logic. Limits of certain variables are defined in the control system. When certain value exceeds a set limit, control system is trying to correct it with appropriate action. Beside the described tasks, wind turbine control system continuously monitors the safety of all the system components, such as sensors, actuators and control hardware. Also, it monitors the weather conditions, especially wind speed and wind direction measurements. Normal operating state of wind turbine is a state when all components of the system are correct, and all process and atmospheric values are within acceptable limits. Some component error or a process variable deviation from desired range indicates emergency system state which requires appropriate action. Malfunction of some system component almost always causes wind turbine stop. Deviation of some process variables from set limits may immediately cause the system stop but also can be tolerated for a certain time in which the control system tries to set process variables to its limits.

Wind is a very complex phenomenon, and its behavior is difficult to describe because of the constant turbulence and unpredictable fluctuations. Wind speed can't be enough accurately and quickly measured because anemometers for wind speed measuring are on top of the nacelle, a few meters behind the rotor and measuring signal is always delayed. Besides that, measured wind is significantly degenerated (filtered) by passing through the blades and can't represent wind that determines the aerodynamic conditions on the rotor. Therefore, wind speed estimation is necessary.

Anemometers defect almost always causes the wind turbine stop. If wind speed estimator exists, in case of wind measurement error, control system use estimated instead of measured wind speed and wind turbine doesn't necessarily have to stop. That increases the availability of the system [4], [5].

In this paper, wind speed estimation is based on aerodynamic torque calculation and is described in third section. The algorithm is suitable for text-oriented programming, but some control systems avoid it and use graphical programming because of the programming simplicity, traceability and determination.

The paper is organized as follows. The second section describes the commercial wind turbine control system in which the algorithm is implemented and validated. It also describes the concept of programming environment based on the block diagrams in accordance with IEC 61131-3, which is used for the algorithm implementation.

The third section describes wind speed estimation algorithm, based on the aerodynamic torque calculation [6]. Based on the algorithm described in the third section, the fourth section explains implementation of wind speed estimator in the target embedded control system. Also describes the algorithm modifications customized to the programming environment based on block diagrams and fixed-point arithmetic. The results of simulations and comparison of measured and estimated wind speed are presented in the fifth section. At the end, conclusion and outlooks for future work are given.

II. KONWECS WIND CONTROL SYSTEM

A. Hardware architecture

KONwecs Wind Control System is the main control system for the wind turbine. Apart from local turbine functions, it also supports power plant and remote access functions. The main supported functions are: speed and power control, torque control, pitch control, brake control, yaw control, sequence control, vibration monitoring, control of cooling system of power converter, fault-ride-through during disturbances, remote control, event recorder, disturbance recorder, communication with other subsystems within the tower and communication with other turbines within the wind farm. The core of the system consists of three independent mounting racks:

- tower base rack mounted in the tower base cubicle,
- nacelle rack mounted in the nacelle cubicle,
- monitoring rack mounted in the nacelle cubicle.

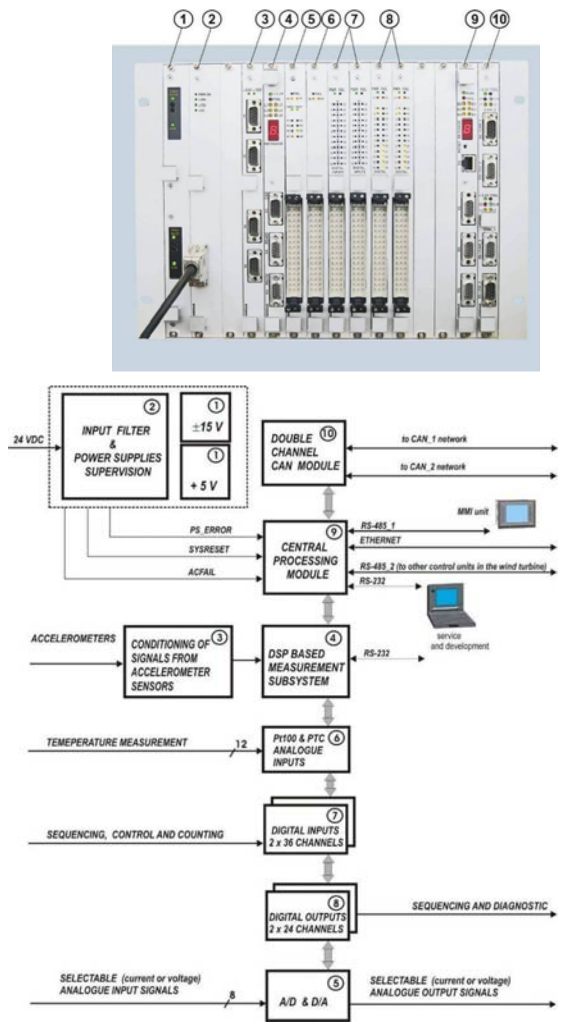
The simplified block diagram of the tower base rack and the front view photo is given in Fig. 1. The following processors have been deployed in this rack: one 32-bit μC, eight 8-bit μCs, and one 32 bits DSP.

Nacelle Control Unit shown in Fig. 1 contains the following HW components:

①, WCSPS, +5V power supply and ±15V power supply integrated into one component;

②, FILCMPS, filters input voltage, monitors outputs of power supplies, performs power-on sequencing and generates error signals in the case of power supply failure;

③, ACCAI, 4 channels for conditioning of signals obtained from vibration (accelerometer) sensors;



① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩
24 VDC
INPUT FILTER & POWER SUPPLIES SUPERVISION ②
±15 V ①
+5 V ①
DOUBLE CHANNEL CAN MODULE ⑩
to CAN_1 network
to CAN_2 network
PS_ERROR
SYSRESET
ACFAIL
CENTRAL PROCESSING MODULE ⑨
RS-485_1
MMI unit
ETHERNET
RS-485_2 (to other control units in the wind turbine)
RS-232
service and development
ACCELEROMETERS
CONDITIONING OF SIGNALS FROM ACCELEROMETER SENSORS ③
DSP BASED MEASUREMENT SUBSYSTEM ④
RS-232
TEMEPERATURE MEASUREMENT
12
Pt100 & PTC ANALOGUE INPUTS ⑥
SEQUENCING, CONTROL AND COUNTING
DIGITAL INPUTS 2 x 36 CHANNELS ⑦
DIGITAL OUTPUTS 2 x 24 CHANNELS ⑧
SEQUENCING AND DIAGNOSTIC
SELECTABLE (current or voltage) ANALOGUE INPUT SIGNALS
8
A/D & D/A ⑤
SELECTABLE (current or voltage) ANALOGUE OUTPUT SIGNALS

Fig. 1. Components deployed and block diagram of Nacelle Control Unit

④, DSPM6, DSP based module for implementing vibration measurement subsystem. It is connected to I/O channel that enables communication with CPM7 or CPM8;

⑤, AIAOGI, general-purpose analogue/digital and digital/analogue conversion module. A/D section consists of 8 galvanically isolated voltage or current (selectable) input channels. D/A section contains 2 galvanically isolated voltage or current (selectable) output channels. It is connected to I/O channel;

⑥, Pt100AI, temperature measurement module, connected to I/O channel. It supports up to 12 Pt100 input channels and 4 PTC input channels;

⑦, DIGIN6, digital inputs module, connected to I/O channel. It supports up to 32 opto-isolated digital inputs and 4 opto-isolated counter inputs;

⑧, DIGOUT5, digital outputs module, connected to I/O channel. It supports up to 16 solid state relay outputs and up to 8 outputs based on classical mechanical relays;

⑨, CPM7 or CPM8, central processing module performs the majority of control, sequencing, diagnostic and communication functions;

⑩, DCAN3, double channel CAN communication module, based on two local microcontrollers. The module is connected to the VMEbus that enables communication with CPM7 or CPM8.

Remaining mounting racks (tower base and monitoring racks) are very similar to the main nacelle rack [7], where the main part of wind turbine control algorithms are implemented and also wind speed estimation algorithm, described in fourth section.

B. Software architecture

The component-based software is adapted to the modular hardware architecture. It therefore enables the development of simple as well as complex control applications. System software and application IDE (integrated development environment) are proprietary solutions where IDE is based on block diagrams. IDE is always based on the same graphical utilities, regardless of the processor type.

Software architecture is divided to system software, integrated development environment for application programming (IDE) and control program, Fig. 2.

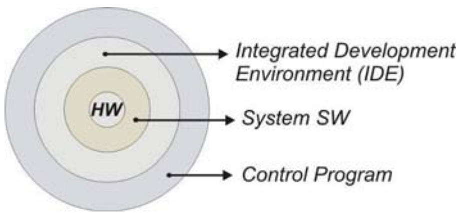


Fig. 2. Software architecture

System software comprises real-time kernel and system programs. System software consists of interrupt driven real-time kernel, monitor, data transmission software and diagnostic software.

Real-time kernel handles tasks according to the preemptive fixed priority scheduling policy. This means the kernel provides the environment in which every separate executable piece of code, i.e. each periodic task and each aperiodic task is assigned a priority. Scheduler determines when a task context switching (the process of saving the state of the currently running task and then loading the state of the task to be run next) should take place and which task should be run next according to the assigned priorities. Therefore, the task with lower priority can be preempted with the higher priority task. The actual prioritizing of the process interrupt tasks is handled by the processor's internal hardware, while the prioritizing of the cyclic tasks is handled in software [8].

Application program is executed under the control of system software.

Development environment for application programming consists of databases containing SW components, graphical tools and development framework.

Control program is the union of application program developed by application engineer and system software.

Integrated development environment for application programming is used for developing, testing and documenting the application (user) program with the aid of the block diagram, user oriented, graphical language.

Fig. 3 describes steps during the application program development. Databases contain SW components in two main environments: code environment and graphical environment. Graphical environment databases are libraries that contain elements and modules, i.e. components on different hierarchical levels. Code environment databases are elements source code library, modules source code library, modules relocatable object code library. The interface between graphical representation and source code is achieved by means of graphical postprocessor.

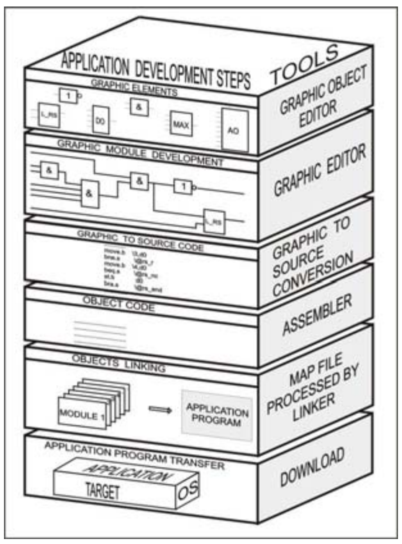


Fig. 3. An example of graphic symbols (SW components) connection made by graphical editor

III. Wind Speed Estimation Based On The Aerodinamyc Torque

Wind speed estimation is based on the principle equation of motion (1), which describes the rotor speed changes as a result of differences between aerodynamic torque M_a on the wind turbine shaft and electromagnetic generator torque M_g. Aerodynamic torque on the wind turbine shaft can be described by means of torque coefficient C_q [2], according to the (2).

$$M_a - M_g = J\frac{d\omega}{dt}, \qquad (1)$$

$$M_a = C_q(\lambda,\beta)\cdot\frac{1}{2}\rho_z\pi R^3\left(v_w - \dot{x}_{t\_nod}\right)^2 . \qquad (2)$$

Substituting expressions (2) at (1) follows:

$$f(v_w) = C_q(\lambda,\beta)\cdot\frac{1}{2}\rho_z\pi R^3\left(v_w - \dot{x}_{t\_nod}\right)^2 - M_g - J\dot{\omega} = \quad (3)$$

where v_w is estimated effective wind speed, J is total moment of inertia of rotor and generator, $\dot{\omega}$ is rotor acceleration, R is rotor radius, ρ_z is air density and $\dot{x}_{t\_nod}$ is tower nodding speed.

The above expression is an implicit nonlinear equation of the wind speed, because the wind speed is contained in tip speed ratio $\lambda = \omega R/v_w$ [2], which determines the torque coefficient C_q. For such an equation is not possible to find an explicit analytical solution. Equation (3) can be solved by using a numerical method to determine zero points of the function $f(v_w)$. Secant method [9] proved to be suitable to solve the equation (3). Function (3) is a higher order function, and certainly has a more than one zero points but only one matches the required wind speed. To avoid convergence of the algorithm in one of the zero points which do not correspond to the actual wind speed, before estimation algorithm starting it is necessary to "position" near the required speed. It is possible to do by using measured pitch angle and its familiar (static) correlation of the wind speed. The wind speed estimation from the implicit equation (3) by using the secant method can be described as an iterative algorithm given in Tab. 1. In the described algorithm, rotor acceleration is approximated by the central difference:

$$\dot{\omega}[kT] \approx \frac{\omega[(k+1)T] - \omega[(k-1)T]}{2T} \quad (4)$$

TABLE I. WIND SPEED ESTIMATION ALGORITHM

1. Assume two initial values of wind speed, $v_{w,1}$ i $v_{w,2}$ based on the pitch angle, and calculate functions $f(v_{w,1})$ and $f(v_{w,2})$;
2. Set step i to the value $i = 2$;
3. Determine the following wind speed value according to the relation
   $$v_{w,i+1} = v_{w,i} - \frac{v_{w,i} - v_{w,i-1}}{f(v_{w,i}) - f(v_{w,i-1})}\cdot f(v_{w,i});$$
4. Check the solution convergence which means to verify if the difference between last two wind speed estimated values is less than a predefined tolerance ε. If $|v_{w,i+1} - v_{w,i}| < \varepsilon$ is true the process is completed;
5. If the required accuracy ε is not satisfied, calculate $f(v_{w,i+1})$, increase i for 1 and repeat step 3.

When calculating $f(v_w)$, it is necessary to determinate torque coefficient C_q, based on the tip speed ratio λ and pitch angle β. Typical dependence of torque coefficient upon tip speed ratio with pitch angle used as a parameter is shown in Fig. 4.

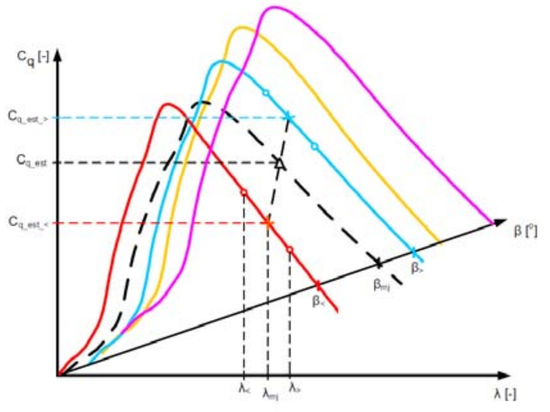


Fig. 4. The idea of three-dimensional interpolation to determine the current value of the torque coefficient C_q

IV. IMPLEMENTATION OF WIND SPEED ESTIMATION ALGORITHM IN KONWECS

The algorithm described in the previous section is implemented in the target embedded control system, precisely in the processor module, which contains a 32-bit microcontroller. The algorithm is implemented in fixed-point arithmetic using 16 bits with sign [11]. Thus, the all SI values are represented by integers in range from -32768 to 32767.

The third step of the algorithm includes multiplication and division, and also calculation of $f(v_{w,i})$ and $f(v_{w,i-1})$. Because of scaling of all units and adjustments to integer arithmetic, all these operations cause a precision loss. Therefore, the algorithm from Tab. 1 is modified and given in the Tab. 2.

TABLE II. MODIFIED WIND SPEED ESTIMATION ALGORITHM

1. Assume two initial values of wind speed, v_{w11} i v_{w21};
2. Check the solution convergence according to the $|v_{w1i} - v_{w2i}| < \varepsilon$. If the required accuracy is satisfied, the process is completed;
3. Calculate $v_{w,i} = (v_{w1i} + v_{w2i})/2$;
4. If the required accuracy is not satisfied, calculate $f(v_{w,i})$. If $f(v_{w,i}) \geq 0$, set $v_{w1i+1} = v_{w,i}$, and if $f(v_{w,i}) < 0$ set $v_{w2i+1} = v_{w,i}$.
5. Increase step i for 1 and repeat step 2.

Application program IDE does not support *for*, *while* or *repeat* loops, therefore, wind speed estimation can't be realized using the described iterative procedure.

As noted in the second section, the processes are arranged by tasks, depending on their priorities. Higher priority processes are performed on faster tasks, while lower priority processes are performed on slower tasks. Control algorithms are performed on the task T3 = 100 [ms]. Without using the loops, wind speed estimator can be realized on the task T0 = 1 [ms]. In this case, between each change of the process variables that are required for the estimation, estimation process is repeated up to 100 times. In practice, 100 steps are always enough for the estimation algorithm. Nevertheless, this method is not suitable for use, because number of iterations is limited and besides that, performing on the T0 task could cause CPU overload.

Another way of implementation is by using *if-end* loop to realize *while* loop. *If-end* loop work in a way that, if the execution condition of IF element is *true*, all elements with order of executions (OE) between order of execution of IF and END elements are performed. If the condition is *false*, END element is executed immediately. In this algorithm, *if-end* loop is not used in the usual way, because END element is executed before IF element (it has smaller order of execution). So, END element is performed first. Then steps 2.-5. from Tab. 2 are executed. If the required accuracy has been satisfied, the execution condition of IF element is set to *false*, and the process is completed. If the required accuracy is not satisfied, the execution condition of *if* loop is still *true*, so *if* element is executed, afterwards *end* element is performed and the process is repeated. In the case that the required accuracy is never being satisfied, this algorithm would go into an infinite loop. To avoid this, there is another condition that interrupts execution. Execution is terminated if the number of iterations is greater than *n*.

In practice, approximately 13 steps are enough to perform the algorithm to an accuracy of $1/362 = 2.76 \cdot 10^{-3}$. Greater precision can't be satisfied because wind speed is scaled in a way that 1 unit in IDE represents 1/362 [m/s].

V. ALGORITHM VALIDATION

The validity of the algorithm is tested on two examples of turbulent wind, measured with anemometers that measure wind speed on the real wind turbine, Fig. 5 and Fig. 6. On both figures (Fig. 5 and Fig. 6), the first and second graphs show the comparison of the measured and estimated current wind speed. Wind on first graph is estimated by using the algorithm described in the third section (Tab. 1) and implemented in Matlab m-function. Function inputs are measured pitch angle, rotor speed and generator torque. Estimator of the wind speed shown on the second graph, is implemented in the target embedded control system, by using the algorithm described in the fourth section (Tab. 2). The third graph shows the estimation of a 30-seconds average value, also implemented in the target embedded control system.

Both figures show very good matching between the actual wind speed and the estimated one, realized in Matlab, which confirms the algorithm validity. Slightly worse behavior is evident in the results of estimator realized on the target system. That is expected, according to the implementation limits, mentioned in the fourth section.

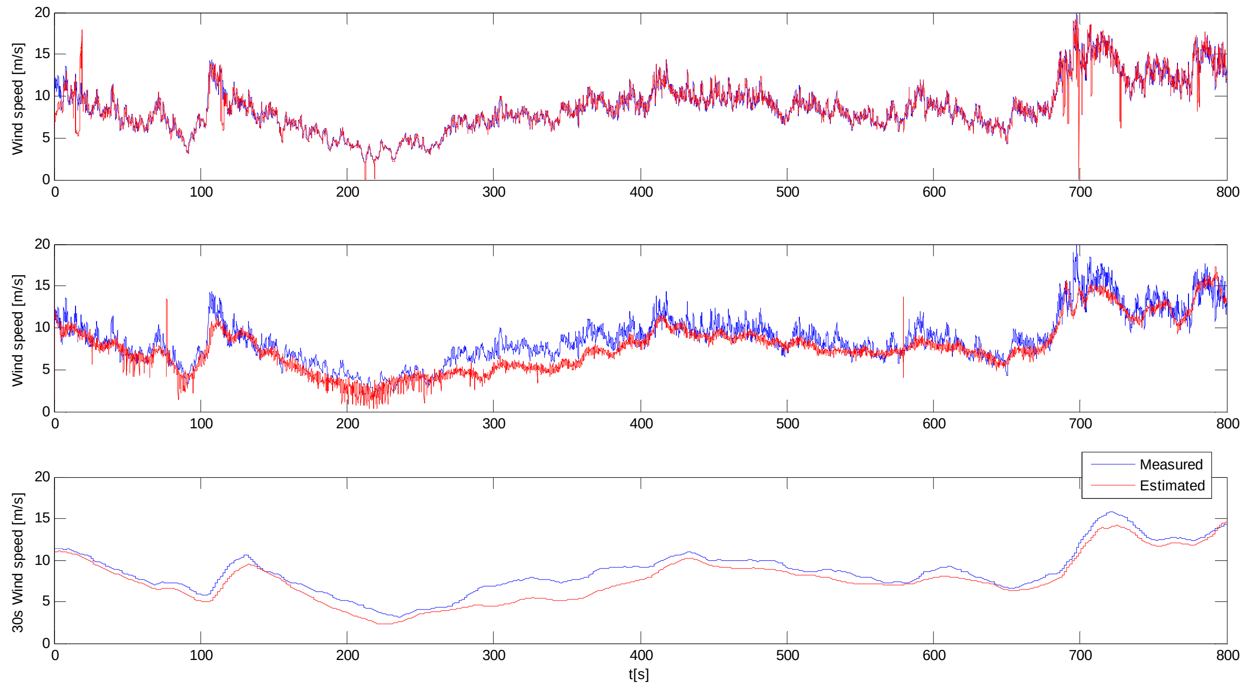


Fig. 5. Comparison of measured and estimated wind speed for the first example of turbulent wind

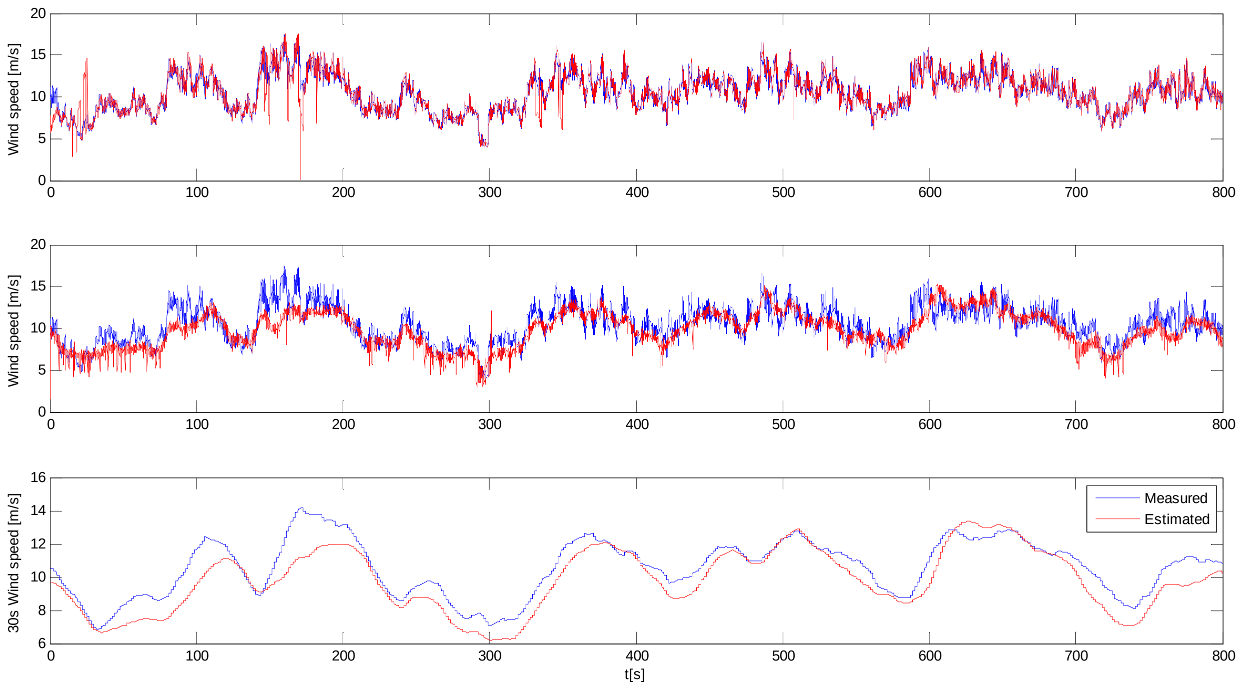


Fig. 6. Comparison of measured and estimated wind speed for the second example of turbulent wind

I. CONCLUSION

Current wind speed fluctuates too much to be used in control algorithms. Therefore, the 30-seconds average value is used. Estimation of the current turbulent wind speed shows that the estimated speed doesn't completely match actual wind speed. Estimation of a 30-second average wind speed value slightly better follows the actual average value. Certain estimator deviation is mostly caused by the rotor acceleration calculation which is approximated by the central difference (4), and besides that derivation is especially sensitive to noise. Potential improvement could be achieved by using higher order difference [10] and by using filters to reduce the sensitivity to noise.

The benefit of this estimation algorithm is reflected in the anemometer failure detection. On the wind turbine, there are two anemometers that parallel measure wind speed. If the measurements are different and there is no wind speed estimation, control system declares failure and wind turbine stops. If the wind speed estimation is present, it is compared with both anemometer measurements, and if one of them matches with the estimator, that measurement is declared correctly. Proper measurement is further used in the control algorithms and the turbine stays in normal operation. In this way, the system availability is increased up to 0.5-1%.

Estimation of effective wind speed can be also used in the design of wind turbine structural loads estimator [6].

REFERENCES

[1] Van der Hooft, E. L; Schaak, P; Van Engelen, T. G: Wind turbine control algorithms. ECN-C—03-111, 2003.

[2] Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E.A.: Wind Energy Handbook. John Wiley & Sons, Ltd, 2001.

[3] Novak, P., Ekelund, T., Jovik, I., Schmidtbauer, M.: Modeling and control of variable-speed wind-turbine drive-system dynamics. Control system magazine, August 1995,15(4):28-38.

[4] Suarez, J., Azagra, E., Urroz, Y., Mascarell, O.H.: Application of wind speed estimation for power production increase. The European energy wind association – EWEA 2012, Copenhagen, Denmark, 2012.

[5] Van der Hooft, E.L., Van Engelen, T.G.: Estimated Wind Sped Feed Forward Control for Windturbine Operation Optimization. ECN-RX-04-126, 2004.

[6] Jelavić, M.: Wind turbine control for structural dynamic loads reduction. PhD Thesis, Faculty of electrical engineering, Zagreb, Croatia, 2009.

[7] Marijan, S.: Sustainability of embedded control systems for rail vehicles and power generation units. PhD Thesis, Faculty of electrical engineering, Zagreb, Croatia, 2011.

[8] Crnkovic, I.: Component-based Software Engineering for Embedded Systems. Conference on Software Engineering, ICSE'05, May 2005, St. Louis, USA, 712-713.

[9] Ivanuš, I.: Numerička matematika. Element, Zagreb, 2002.

[10] Perić, N., Petrović, I.: Automatizacija postrojenja i procesa, predavanja. Faculty of electrical engineering, Zagreb, Croatia, 2005.

[11] Yates, R.: Fixed-Point Arithmetic: An Introduction. Digital Signal Labs, August 2007.