Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL'INGEGNERIA

CICLO XXVIII

COORDINATORE Prof. Stefano Trillo

Data-Driven Fault Diagnosis and Fault Tolerant Control of Wind Turbines

Settore Scientifico Disciplinare ING-INF/04

Dottorando	**Tutore**
Dott. Farsoni Saverio	Dott. Ing. Silvio Simani

Anni 2013/2016

Acknowledgments

Firstly, I am grateful to my supervisor, Silvio Simani, for the constant assistance and the valuable suggestions that encouraged my personal and academic growth. Thanks to Marcello Bonfè and all the staff of the Laboratory on Intelligent Robotics and Automation, Department of Engineering, University of Ferrara, for the help and friendship established with a beginner such as me. Furthermore, I wish to express my gratitude to Paolo Castaldi, Elena Zattoni, Elisabetta Penati, from the University of Bologna, for the productive discussions and collaborations.

Special thanks to my family, my girlfriend and my friends for always supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The worldwide increased level of wind generated energy in power grids induces further requirements in terms of reliability and sustainability of wind turbines. Wind farms should have the capability to generated the desired value of electrical power continuously, depending on the current wind speed level and on the grid demand.

As a consequence, the possible faults affecting the system have to be properly identified and treated, before they endanger the correct functioning of the turbines or become critical faults. Wind turbines in the megawatt size are extremely expensive systems, therefore their availability and reliability must be high, in order to assure the maximization of the generated power while minimizing the Operation and Maintenance (O & M) services. Alongside the fixed costs of the produced energy, mainly due to the installation and the foundation of the wind turbine, the O & M costs could increase the total energy cost up to about the 30%, particularly considering the offshore installation (Odgaard and Patton, 2012).

These considerations motivate the introduction of fault diagnosis system coupled with fault tolerant controllers. Currently, most of the turbines feature a simply conservative approach against faults that consists in the shutdown of the system to wait for maintenance service. Hence, effective strategies coping with faults have to be studied and developed, for improving the turbine performance, particularly in faulty working conditions. Their benefits would concern the prevention of critical failures that jeopardize wind turbine components, thus avoiding unplanned replacement of functional parts, as well as the reduction of the O & M costs and the increment of the energy production. The advent of computerized control, communication networks and information techniques brings interesting challenges concerning the development of novel real-time monitoring and fault tolerant control design strategies for industrial processes.

Indeed, in the recent years, many contributions have been proposed related to the topics of fault diagnosis of wind turbines, see e.g. (Chen et al., 2011),(Gong and Qiao, 2013). Some of them highlight the difficulties to achieve the diagnosis of particular faults, e.g. those affecting the drive-train, at wind turbine level. However these fault are better dealt with at wind farm level, when the wind turbine is considered in comparison to other wind turbine of the wind farm (Odgaard and Stoustrup, 2013). Moreover, fault tolerant control of wind turbines has been investigated e.g. in (Odgaard and Stoustrup, 2015), (Parker et al., 2011) and international competitions on these issues arose (Odgaard and Stoustrup, 2012), (Odgaaard and Shafiei, 2015).

Hence, the *sustainable* control of wind turbine systems has been proven to be a challenging task and motivates the research activities carried out through this thesis.

In the following, after an overview about the nomenclature mostly endorsed in the related literature, a general introduction on fault diagnosis and fault tolerant control strategies is presented. Finally, the contents of this thesis is summarized.

1.1 Nomenclature

Since 1991 the Symposium on Fault Detection Supervision and Safety for Technical Processes (SAFEPROCESS), organized by the International Federation of Automatic Control (IFAC), has represented one of the most important international gathering of academia and industry experts, focused on the topics of: diagnosis of dynamic systems, fault tolerant control, process supervision, system reliability and safety, etc. Because of the past inconsistency of the related literature in terms of nomenclature, the SAFEPROCESS committee suggested a set of definitions that represent a good introduction to the issue of fault diagnosis and fault tolerant control. The main definitions, taken from (Isermann and Balle, 1997) and listed in the following with a brief description, regard the system states and signals, the functions and tasks, the system properties, and the fault characteristics.

- **System states and signals**:
  - **Fault**: an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable, usual, or standard conditions.
  - **Failure**: a interruption of a system's ability to perform a required function under specified operating conditions.
  - **Malfunction**: an intermittent irregularity in the fulfillment of a system's desired function.
  - **Error**: a deviation between a measured or computed value of an output variable and its true or theoretically correct one.
  - **Disturbance**: an unknown and uncontrolled input acting on a system.
  - **Residual**: a fault indicator, based on a deviation between measurements and model-equation-based computations.
  - **Symptom**: a change of an observable quantity from normal behavior.
- **Functions and tasks**:

  - **Fault detection**: determination of the presence of faults and the time of occurrence.
  - **Fault isolation**: determination of the kind, location and time of detection of a fault. Follows fault detection.
  - **Fault identification**: determination of the size and time-variant behavior of a fault. Follows fault isolation.
  - **Fault diagnosis**: Determination of the kind, size, location and time of detection of a fault. Follows fault detection. Includes fault detection and identification.
  - **FDI**: acronym of Fault Detection and Isolation.
  - **FDD**: acronym of Fault Detection and Identification (Diagnosis)

- **Models**:
  - **Quantitative model**: use of static and dynamic relations among system variables and parameters in order to describe the system behavior in quantitative mathematical terms.
  - **Qualitative model**: use of static and dynamic relations among system variables in order to describe the system behavior in qualitative terms, such as causalities.
  - **Analytical model**: the mathematical relations among the system variables are based on physical laws deriving by the knowledge of the system behavior.
  - **Data-driven model**: the mathematical relations among the system variables are inferred on the basis of a set data coming from the system itself.

- **System properties**:
  - **Reliability**: ability of a system to perform a required function under stated conditions, within a given scope, during a given period of time.
  - **Safety**: ability of a system not to cause danger to persons or equipment or the environment.
  - **Availability**: probability that a system or equipment will operate satisfactorily and effectively at any point of time.

- **Faults, time dependency**:
  - **Abrupt fault**: fault modeled as stepwise function. It represents bias in the monitored signal.
  - **Incipient fault**: fault modeled by using ramp signals. It represents drift of the monitored signal.

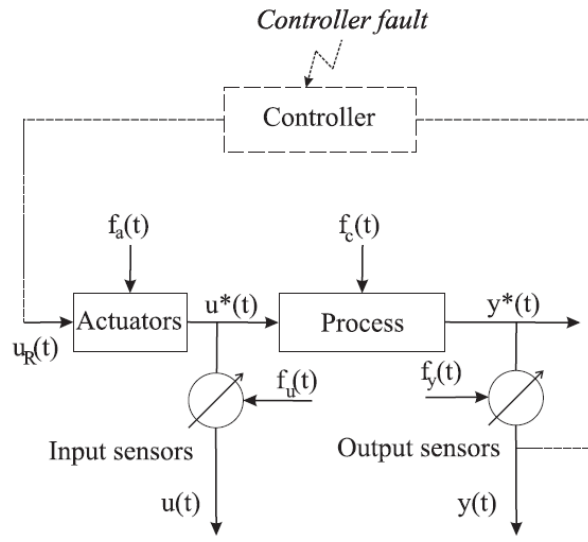

Figure 1.1: The possible fault locations inside the system components.

- **Intermittent fault**: combination of pulses with different amplitudes and lengths.

- **Faults, typology**:
  - **Additive fault**: Influences a variable by an addition of the fault itself. They may represent, e.g., offsets of sensors.
  - **Multiplicative fault**: are represented by the product of a variable with the fault itself. They can appear as parameter changes within a process.

1.2 Fault Diagnosis Methods

There exist several ways in which faults can affect the system. According to Fig. 1.1 a closed loop system can be viewed as the union of interconnected elements, namely the main process, the actuators, the controller, the input sensors and the output sensors. Each of these components can be associated to a fault, so that process faults, actuator faults, controller faults, and sensor faults can be considered.

In several application domains, a typical approach to fault diagnosis involves the so-called *hardware redundancy* (Isermann, 1997). It consists in the usage of multiple sensors, actuators or components to measure, or control, a particular signal. The related diagnosis is based on the comparison among the different redundant hardware information, hence, a voting technique is adopted to decide if and where a fault occurs. Although the hardware redundancy can be very effective, it involves extra cost concerning the equipment and the maintenance operations, that could be in conflict with the sustainability requirements.

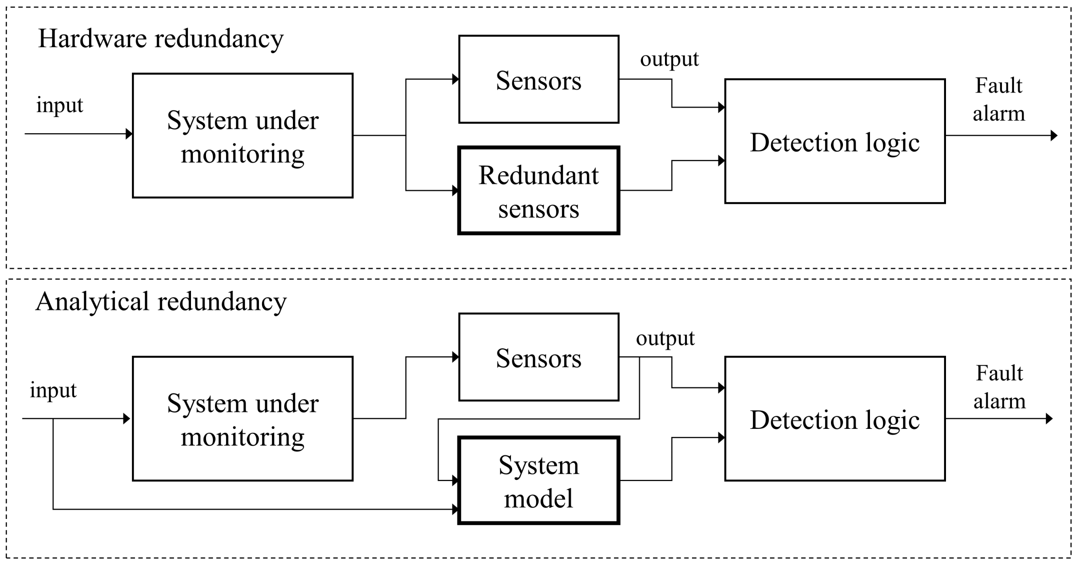

Figure 1.2: The hardware and the analytical redundancy schemes.

On the other hand, the so-called *analytical redundancy* (Patton et al., 1989), does not need additional hardware components, but it exploits a model of the system under investigation for estimating the value of a particular variable on the basis of the pre-existent sensors, so that a residual signal can be generated and a diagnosting logic can be inferred. In particular, the residual signal, already defined in Section 1.1, should be close to zero in normal operating condition and significantly different from zero when a fault occurs. The analytical redundancy is often referred to as *model-based* approach, because of its dependence on the system model. A model-based module can be implemented via software on a process control computer, without additional costs on the equipment, as already remarked. Some drawbacks of the model-based approach concern the accuracy of the generated estimations, as well as the disturbances and the noise affecting the process, that can lead to false alarm or missed fault.

The block diagram of Fig. 1.2 illustrates the typical implementation of the hardware redundancy compared with the analytical redundancy.

In more details, the detection of a fault by means of the model-based approach relies on the generation and the evaluation of the residual, as depicted in the block diagram of Fig. 1.3.

The residual is firstly generated as the error between the measured and the estimated variable. The latter comes from the model elaboration of the input-output measurements. The residual, at this stage, should be independent from system input and output, as it is ideally zero for every input-output conditions in fault-free case. Then, the residual has to be evaluated, in terms of fault likelihood, and a decision rule has to be applied to determine the fault occurrence. At this stage the residual can be simply compared to a fixed threshold (also called geometrical

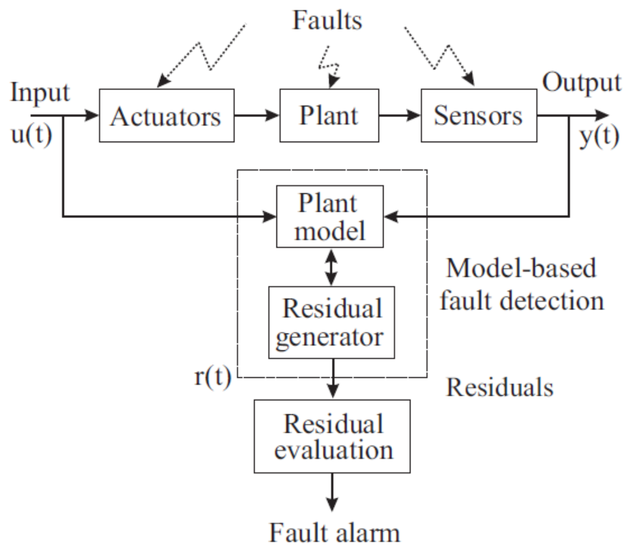

Figure 1.3: The detection logic based on the residual generation and evaluation.

methods), or pre-processed through a proper function, e.g. a moving average or more complex statistical methods, ahead of the threshold test.

A particular class of the model-based methods, called *signal-based* technique, occurs when only output measurements are available. it includes the case of vibration analysis, (e.g. related to rotating machinery) that are performed by means of band-pass filters or spectral analysis.

Among the basic model-based FDI techniques (Simani et al., 2003) can be mentioned the output-observer, the parity equation, the parameter estimation. They require an adequate knowledge of the state-space or input-output behavior of the system under investigation, that is expressed in terms of analytical relationships.

However, most of the real industrial processes have a strongly nonlinear behavior, that cannot be modeled using a single model for all the operating conditions. Indeed, the parameters of the system may vary with time and the unavoidable disturbances and noises have unknown characteristics. The result is a discrepancy between the plant behavior and its mathematical description, even in fault free condition, that can lead to the impossibility of generating a residual, as the fault may be hidden by the modeling errors. The Unknown Input Observer (UIO), the eigenstructure assignment and the parity relation methods are FDI strategies that take into account this key issue (Simani et al., 2003).

Therefore, the proper mathematical description of the system under investigation, required by classical model-based FDI strategy, is very difficult to derive in practice, sometime even impossible. Because of these assumptions, data-driven modeling approaches offer a natural tool to handle a poor knowledge of the system, together with disturbances and noises. Indeed, their implementations exploit input-output data directly acquired from the system, for deriving the relations among those variables. In particular, fuzzy logic theory (Babuška, 2012) allows the representation of the process under investigation by means of a collection of local affine fuzzy models, among which the transition are handled by identified fuzzy parameters. Furthermore, also neural networks can handle complex nonlinear behaviors (Liu, 2012), as they have the capability of learning the system functioning on the basis of the information provided by the training data. These considerations support the studies on data-driven strategies applied to the wind turbine context, because their behavior is characterized by very complex and noisy nonlinear dynamics, as described in Chapter 2.

1.3 Fault Tolerant Control Methods

Control solutions that are able to cope with possible malfunctions and fault situation are usually referred to as *Fault Tolerant* Control system (FTC), as they possess the capability to automatically manage the faults affecting the components (Patton, 2015). Fault tolerant control methods can be classified into two main categories, namely Passive Fault Tolerant Control (PFTC) and Active Fault Tolerant Control (AFTC) schemes. PFTC techniques do not require neither fault diagnosis nor controller reconfiguration, as they are designed to be robust against a set of possible faults. However, although they do not involve the problem related to fault diagnosis, as false alarms or missed fault, they have limited fault tolerant capabilities. They make use of robust control technique to ensure that the considered closed loop system remains insensitive to faults, using fixed controller parameter, without needing information regarding the fault occurrence. This is accomplished by designing a controller that is optimized for fault-free situations, while satisfying some graceful degradation requirements in the faulty case.

On the other hand, AFTC reacts to the faults actively by reconfiguring the control actions so that the performance of the fault-free system can be maintained, even in presence of faults.

In particular, AFTC is mainly based on a fault diagnosis block, that provides the real-time information about the faulty, or fault-free, status of the system under monitoring (Mahmoud et al., 2003). The controller exploits a further control loop aimed at the compensation of the faulty signals (fault *accommodation*). The main advantage of this approach is that the controller can be designed considering only the nominal operating conditions. The structure of the proposed AFTC scheme is addressed in Chapter 4.

1.4 Outline of the Thesis

The contents of the thesis are briefly summarized in the following, in order to provide an overview of the work.

- **Chapter 2**: it introduces the systems under investigation (i.e. the wind turbines) describing their characteristics, their categories, and their components, together with some global statistics that highlight the importance of the discussed topics. Then, two realistic benchmark systems are detailed presented, that represent high-fidelity simulators of a single wind turbine and of a wind farm, respectively. They rely on the analytical description of the system behaviors, and provide also the models of typical fault cases. The complexity of the nonlinear and uncertain dynamics of the wind turbines and of the related faults, supports the choice of data-driven approaches.

- **Chapter 3**: fuzzy logic and neural networks represent an effective tool for obtaining, from acquired data, a description of the system behavior, without the requirement of a precise analytical knowledge. Therefore, their functioning, their properties and their modeling capabilities are investigated into this chapter.

- **Chapter 4**: this chapter shows how the proposed data-driven techniques lead to the design of fault detection, isolation, and diagnosis schemes that provide the main component of the fault tolerant controller. Moreover, the Fault Mode and Effect Analysis (FMEA) is presented, as it represents an optimization tool able to reduce the complexity of the fault estimators.

- **Chapter 5**: this chapter addresses the performances achieved by developed fault diagnosis and fault tolerant control systems, implemented on both the single wind turbine and the wind farm benchmark models. It reports the result obtained in comparison with other schemes proposed in literature, and briefly described. Then, the validation is also performed by means of a Monte Carlo analysis that takes into account the possible disturbance and uncertainties affecting the wind turbines. Finally, the HIL test is carried out, in order to assess the controller performances in a more realistic framework.

- **Chapter 6**: it summarizes and comments the obtained results and proposes some future investigations about the discussed topic.

- **Appendix A**: this appendix relies on an alternative FTC system, based on adaptive nonlinear filters designed via the geometric approach. This method cannot be considered a pure data-driven scheme as it deals with the analytical representation of the system uncertainties, although it is obtained via a data-driven identification procedure.

Chapter 2

System and Fault Modeling

This chapter considers the system under analysis, i.e. the wind turbines. In the first section the general outlines are reported, with a brief characterization of the main wind turbine components and typologies. Some concepts rely on the control of wind turbines are also presented. Then, a detailed description of the implementation of the benchmark models used in the simulations is reported. They are well known models (Odgaard et al., 2013), (Odgaard and Stoustrup, 2013) which offer a realistic simulation framework for a single turbine and for a small wind park of nine turbines, respectively. Furthermore, several common fault scenarios can be handled.

2.1 System Description

The recent care about climatic variations, the needs of increasing the usage of renewable energy and the problems associated to the availability of fossil fuels yield to a growing attention to the generation of electrical energy from the kinetic wind energy. This kind of source is especially available in the temperate regions, where the most industrialized countries are located. During the last decades, several kind of wind turbines have been proposed and developed: horizontal/vertical-axis, with a different number of blades, upwind, downwind, etc. The upwind three-blades horizontal-axis turbine has achieved the recognition for the most suitable and efficient typology, so that it had a wider development than the other kind of turbines, in terms of power, size and diffusion (ABB, 2011).

Some interesting data can be taken from the statistics of the National Renewable Energy Laboratory (NREL) (GWEC, 2015): the installed wind power capacity grew at a rate of 30% in 5 years from 2002 to 2007, and in 2013 it reaches the value of 369.6 GW, as can be seen in Fig. 2.1.

During the biennium 2010/2011 the 50% of the new plants was installed outside the traditional European and North American market, particularly in China where the wind power capacity currently oversteps the value of 100 MW installed. Furthermore, many European Countries have achieved relatively high levels of wind power penetration, with respect to the other sources. As an example Denmark has

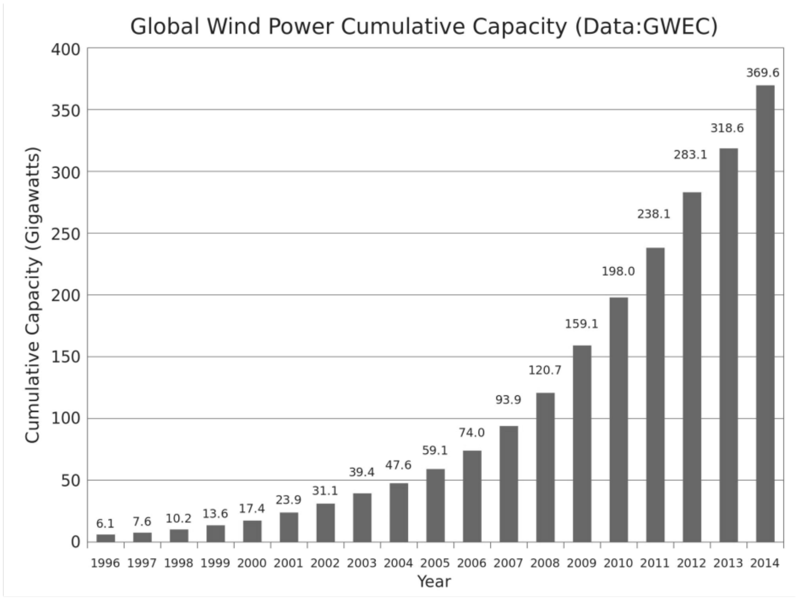

Figure 2.1: The global wind energy installed capacity.

the 21% of the stationary electricity production from wind power, Portugal 18% and Spain 16%.

It is worth observing that also the size of wind turbines strongly increase over the last decades, responding to the necessity of capturing more energy more efficiently. Fig. 2.2 shows the evolution of wind turbines along time.

The physical phenomenon of wind is generated by the movement of an air mass from an area where the atmospheric pressure is high towards an adjacent low-pressure area. The speed of the air flow is proportional to the difference of pressure between the areas. Another prominent factor that influences the wind intensity is the local profile of the land or the sea, indeed the wind gets stronger on large, flat surface as oceans, or on the top of the rises and in the valley oriented parallel to wind direction; while it reduces speed on irregular areas as towns or forests. An evaluation of the World's wind power potential, oriented to wind turbine plant realization, has been conducted in (Archer and Jacobson, 2005). Moreover, the direction and the intensity of the wind change rapidly with respect to their average value. This phenomenon, known as wind turbulence, not only can causes the wear and tear of the turbine blade, but also involves important uncertainty and variability factors that represent one of the main disadvantages in the generation of electrical energy.

The task accomplished by wind turbines is the transformation of the kinetic energy of the wind into electrical energy, without any usage of fuel oil, by means of the conversion into mechanical rotation energy. Indeed, the proper blade design involves a different profiles for each of the two blade surfaces, so that when the wind flows on the surfaces, a depression area is created at the upper surface relative to the lower surface. Because of this pressure difference, a force called *lift* generates the movement of the bound blade and its consequent rotation around the hub axis.

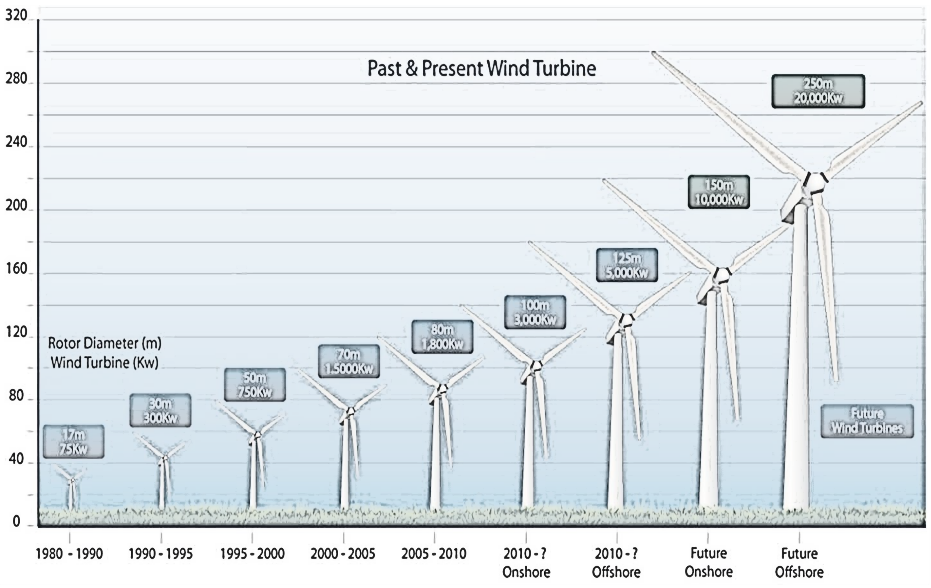

Figure 2.2: The evolution of the size and the nominal power of wind turbines along time.

The starting point for the evaluation of the mechanical power that can be ideally generated by the wind is the computation of the kinetic energy E_k of an air mass m, moving at a certain speed v_w:

$$E_k(t) = \frac{1}{2} m(t) v_w(t)^2$$

The derivative of the kinetic energy in time is the available specific power P_{av} of the air mass:

$$P_{av}(t) = \frac{dE_k(t)}{dt} = \frac{1}{2} q(t) v_w(t)^2$$

Where the capacity $q(t) = \frac{dm(t)}{dt} = \rho A v(t)$ can be calculated as the product of the air density ρ, the speed v_w and the cross-sectional area A of the stream tube of the air under consideration, thus obtaining:

$$P_{av}(t) = \frac{1}{2} \rho A v_w(t)^3 \tag{2.1}$$

With reference to a wind turbine, A represents the rotor swept area and v_w is the rotor effective wind speed, as described in (Bianchi et al., 2006).

The actual power of the rotor cannot be equal to the theoretical available wind power, and the difference is expressed by means of the so-called *power coefficient* $C_P(\beta, \lambda)$, which is a function of the pitch angle of the blades β and the tip speed ratio λ, defined as:

$$\lambda = \frac{\omega_r R}{v_w} \tag{2.2}$$

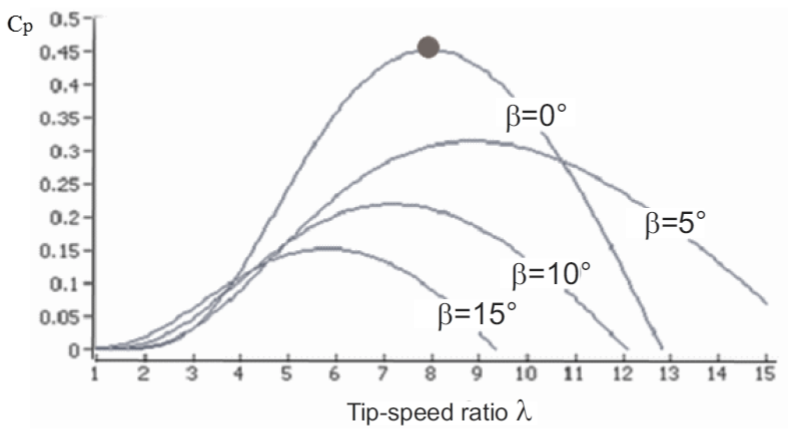

Figure 2.3: The C_p map as a function of the pitch angle and the tip speed ratio.

Where R is the rotor radius, ω_r is the rotor angular speed. Then, the ideal rotor power becomes:

$$P_r(t) = P_{av}(t)C_P(\lambda(t), \beta(t)) = \frac{\rho \pi R^2 C_P(\lambda(t), \beta(t)) v_w^3(t)}{2} \tag{2.3}$$

Alternatively, the aerodynamics of the turbine can be described as in (Gasch and Twele, 2011), considering the rotor torque τ_r, that is transferred from wind to the rotor shaft, and the torque coefficient C_q:

$$\tau_r(t) = \frac{P_{av}(t)}{\omega_r(t)} \lambda C_q(\lambda(t), \beta(t)) = \frac{\rho \pi R^3 C_q(\lambda(t), \beta(t)) v_w^2(t)}{2} \tag{2.4}$$

It is worth noting that the relationships between λ, β and the power coefficient C_P are not analytically known, but they are represented by means of a two-dimensional map (look-up table), shown in Figure 2.3. A similar map can be applied to the torque coefficient C_q.

2.1.1 Wind Turbine Categories

A first categorization of wind turbines is based on the orientation of the rotor axis: it can be either vertical or horizontal, with respect to the installation surface. The vertical axis wind turbines constitutes only the 1% of the total amount of existing plants, while the remaining 99% belongs to the horizontal axis category (ABB, 2011). The wide diffusion of the latter category is mainly due to its higher efficiency. Indeed, the horizontal axis turbines can exploit the effects of a more strong wind, as their rotor is placed on the top of a high tower.

The vertical axis turbines can be divided into three main categories, depending on the construction technology: Savonius type, Darrieus type, and hybrid Savonius-Darrieus type. The Savonius turbines are low-speed turbines suitable for the installation in location characterized by low wind speed, because the blade surfaces are completely exposed to the wind, and cannot be reduced in case of excessive wind speed, that could damage the system without a robust protection structure. Therefore, they are used only in low-power application. The Darrieus turbines are faster and more efficient than the Savonius turbine, and their particular design allows to respond to the variation of the wind speed direction. They are suitable for low wind speed, but they do not need a robust structure to withstand extreme winds, so that they can be used also for large power application.

Regarding the horizontal axis wind turbines, they can be upwind, in case the wind hits the blades ahead of the tower, or downwind otherwise. The main advantage of the upwind configuration is that the tower does not interfere with the rotor, and the whole wind power can be captured by the blades. Their rotor can support one blade with counterweight, two blades or three blades. They need a mechanical system that align the rotor to the wind main direction, such as a yaw system or a tall vane. The efficiency increases with the blade number, but also the cost of production, however, the three blade horizontal axis is the currently most widespread model so that the systems considered in the following belong to this category.

2.1.2 Main Components of Wind Turbines

The structure of a wind turbine, with its main components and their connections, is depicted in Figure 2.4. A brief description of each component will be given in the following. More details are reported in (Darling, 2008).

- **Blades**: they interact with the wind producing the movement of the rotor shaft. The profile of the blade is designed in order to obtain a good value of the aerodynamic lift with respect to the aerodynamic resistance and, at the same time, to oppose the proper stiffness to the applied variable mechanical loads that determine the wear and tear effect along time. The construction materials should be light, such as plastic materials reinforced with glass, aluminum or carbon, depending on the blade size.

- **Hub**: it constrains the blades to the rotor shaft, transmitting the extracted wind power. It can contain the pitch actuator, that forces the blade to have a certain orientation relative to the wind main direction, for control purposes.

- **Brakes**: they can be of various types (mechanical, electrical, hydraulic) and they are used as parking brakes to avoid the rotor movement when the turbine has to be kept in not operating conditions.

- **Gear box**: one (or more) stage gear box is used to adapt the mechanical power of the rotor shaft to the generator shaft, by increasing the rotational speed and by decreasing the torque, in order to permit an efficient conversion of energy. Often the gear box ratio can be greater than 1:100. The design of the gear box involves epicycloidal or parallel axis gears.

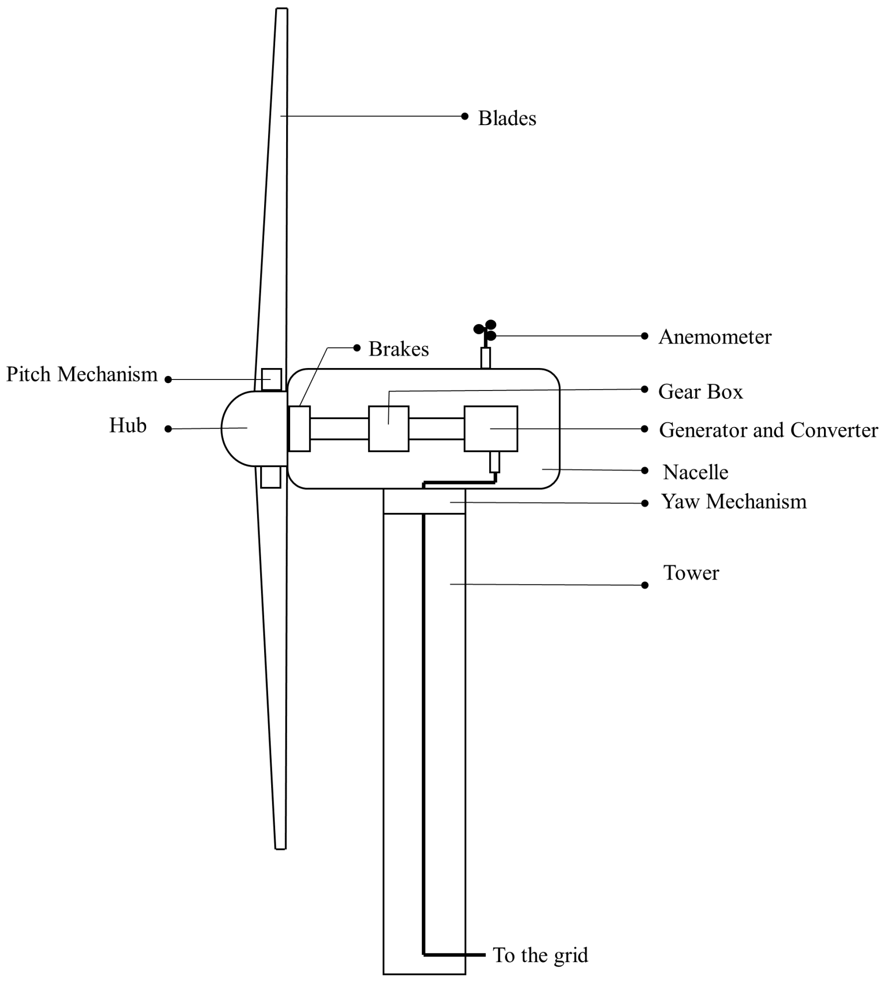

Figure 2.4: The main components of the considered wind turbine, seen from the outside.

- **Generator**: it converts the mechanical energy of the connected shaft to electrical energy. It can be an asynchronous or a synchronous machine, the former consists in an induction three-phase motor that works as generator providing energy to the grid, as its speed is higher than the synchronous speed and the applied torque is motive. The usual structure of an asynchronous generator involves a squirrel cage rotor, that implies a low difference between the synchronous speed and the actual speed in the nominal working region, so that it can be considered a constant speed machine. The configuration called doubly-fed, with a power converter located between the rotor and the grid allows the functioning of the system at a variable rotor speed. Otherwise, the synchronous generator provides a voltage which frequency is proportional to its rotational speed. In the configuration called *full-converter*, similarly to the doubly-fed generators, a power converter has to be interpose between the generator and the grid, in order to permit a variable speed functioning. The output power of a modern turbine can be up to five megawatts.
- **Nacelle**: it is the shell containing the shafts, the brakes, the gear box, the generator and the control equipment. It is located at the top of the tower.
- **Anemometer**: it is the sensor that provides the current wind speed. Its measurements are exploited in the control system.
- **Yaw mechanism**: several electrical motors orientate the heading orientation of the turbine parallel to the main wind direction.
- **Tower**: it supports the nacelle, the hub and the blades. The height of the tower determines the height of the hub that is a prominent value in the generation of power, since the wind speed increases with the distance from the earth surface.

2.1.3 The Overall Wind Turbine Analytical Description

The way in which the components interact leads to complex nonlinear dynamics, that are difficult to describe analytically. Currently, the most completed and used analytical model of horizontal-axis wind turbines is provided by the Fatigue, Aerodynamics, Structures, and Turbulence (FAST) package, developed by NREL (Manjock, 2005).

It represents a high-fidelity 24 degrees of freedom wind turbine model, that become a reference for academia as well as for industry researchers.

According to FAST, the overall model consists of four submodels for the mechanical structure, the aerodynamics, the dynamics of the pitch system and the generator converter system. The compact result is summarized in the following nonlinear state-space representation:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{g}(\mathbf{x}, v_w) \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{cases} \tag{2.5}$$

Where:

- $\mathbf{x}$ is the state vector that includes the main system variables as the pitch angle, the angular position and velocity of the rotor and of the generator, the linear displacement of the tower and of the blades, together with their velocity.

- $\mathbf{u}$ is the input vector, consisting of the reference pitch angle and the generator torque.

- v_w is the wind speed, considered as a disturbance term.

- $\mathbf{y}$ is the output vector that commonly involves some state variables selected by the matrix $\mathbf{C}$.

- $\mathbf{A}$ and $\mathbf{B}$ are constant matrices that define the linear part of the model. They are made up by terms related to the mass, the damping, and the stiffness of the system components.

- $\mathbf{g}$ is the function that takes into account the nonlinearities introduced by the thrust force acting on the rotor and by the aerodynamic rotor torque, by means of the power coefficient above mentioned.

The detailed derivation of this state-space expression can be found in (Simani, 2015), but it is beyond the scope of this thesis. Indeed, although the FAST model represents an appropriate simulation tool for testing the developed control algorithms, the reduced order model offered by the benchmark system described in Section 2.2 provides the suitable solution for design purposes, as it captures only the dynamic effects directly influenced by the controller, with a noteworthy reduction of the computational efforts required by the simulations.

2.1.4 Wind Turbine Control Issues

The control of wind turbines aims at the tracking of reference power curve, whose typical outline is depicted in Fig. 2.5 for a variable-speed turbine of several kW of nominal power (Simani, 2015).

It is a function of the instantaneous wind speed. Indeed, the turbine starts working only when the wind speed is high enough to ensure a power production greater than the system losses. The cut-in wind speed is usually below to 5 m/s. Similarly, in order to prevent serious damages, the turbine is powered down and stopped when the wind speed is too high, usually over the value of the cut-off speed (about 30 m/s). The operating range between the cut-in and the cut-off speed can be subdivided into two main region, namely the partial load region and the full load region. The first tries to maximized the generated power for a given value of the wind speed, then, in the full load region the power is kept constant to its nominal value to avoid exceeding safe electrical and mechanical load limits.

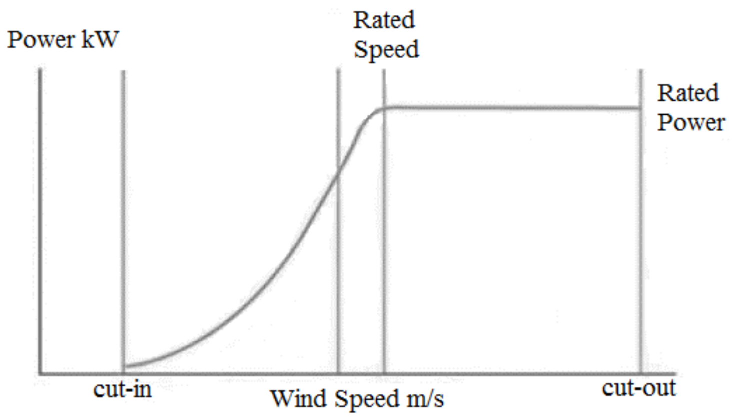

Figure 2.5: The standard reference power curve for a wind turbine, used as target for the control system.

According to the block diagram of Fig. 2.6 a switch reconfigures the control system to the current operating objectives between the partial and the full load regions. In this way two different controllers can be designed for those regions. The first control system should maintain the pitch angle at this optimal value (0 deg) while controlling the generator torque, in order to capture the maximal power from the wind. The full load controller acts on the pitch angle of the blade so that the power is maintained constant, while the generator torque is manipulated to remove the steady-state errors on the output power.

The transition between the two regions are regulated by a *bumpless* transfer mechanism. Indeed, if the switch between the two controller is not softened, a bump in the control signal may induce oscillations between the controllers, that may cause instability. The switching condition is based on the generator speed instead of the wind speed, since the large inertia of the rotor introduces a damping in the speed signal, with respect to the possible unpredictable changing of the wind speed.

More advanced controllers are comprehensive of a structural stress damper, aimed at reducing the drive-train oscillations and the structural stress that affects the turbine tower.

The controller can be implemented following different schemes (Simani, 2015), from the most basic PID regulators to the more recent solution relying on data-driven, adaptive control (Simani and Castaldi, 2013), time-varying approaches (Johnson et al., 2006). Other examples involve the feedback linearization (Kumar and Stol, 2010), or gain scheduling quadratic control (Bossanyi, 2003).

With reference to fault tolerant controllers, their design rely on output feedback and they should take into account the measurement noise. The controllers should manage the parameter-varying nature of the wind turbine during its operating conditions, due to the aerodynamics nonlinearities. A common solution is given by the Linear Parameter-Varying (LPV) modeling. Both AFTC and PFTC strategies can be proposed, the main difference is that AFTC is based on a fault diagnosis algorithm, which provides to the controller the on-line information about the faults,

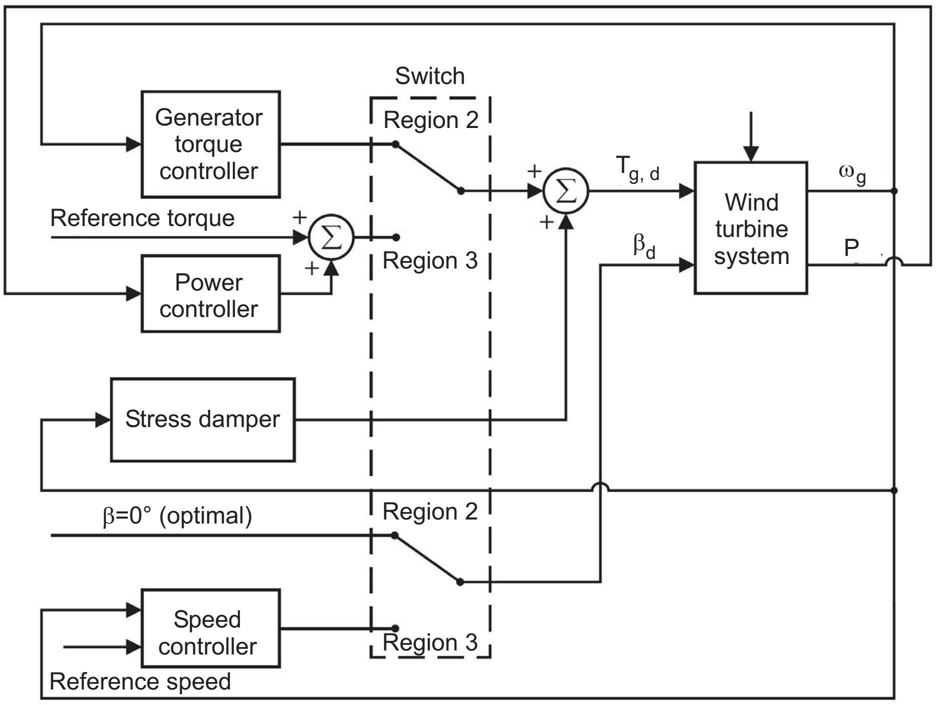

Figure 2.6: The standard configuration of the wind turbine controller.

often considered as unknown inputs. The knowledge of the fault permits the reconfiguration of the controller. The fault diagnosis block should compensate, alongside the faulty signals, the disturbance and the modeling errors.

On the other hand, the PFTC does not need a fault diagnosis module, but it should be robust against all the presumed faults, as already mentioned in Section 1.3. Moreover, for both approaches, the wind speed represents a necessary information, that can come either from acquired measurement or from a wind speed reconstructor. Fig. 2.7 shows the comparison between AFTC and PFTC considering the wind turbine system.

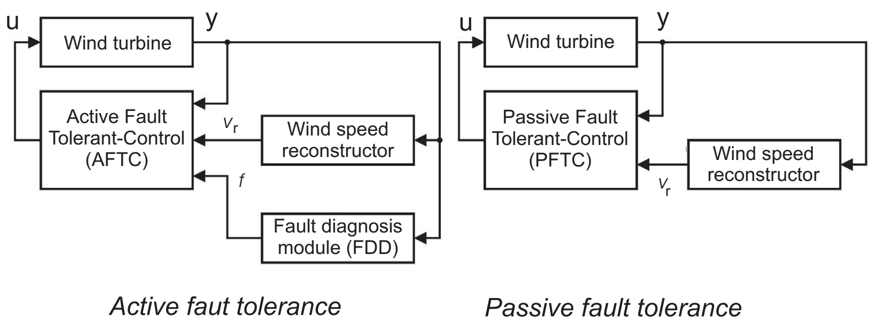

Figure 2.7: AFTC and PFTC comparison.

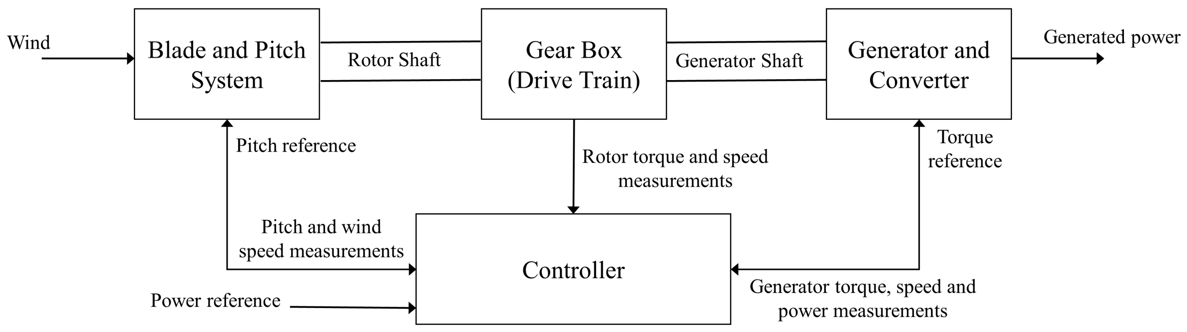

Figure 2.8: The block diagram of the turbine subsystems.

2.2 The Wind Turbine Benchmark System

The single wind turbine benchmark model considered in this thesis is described in detail in (Odgaard et al., 2013). It has been implemented in Matlab/Simulink environment and proposes a realistic simulator for a wind turbine system, as well as some common fault scenarios. It presents a specific kind of wind turbine: a three-blade horizontal-axis variable-speed pitch-controlled turbine with a full converter generator. The considered components have been reduced, with respect to the representation of Figure 2.4. The tower supports the nacelle, that contains the control systems and the equipment to convert energy from the rotating blades fixed in the hub to electrical energy for the grid. On the nacelle an anemometer provides the measurements of the current wind speed at the hub height. The internal components of the nacelle consist of a gear box that connects the rotor main shaft to the generator, adapting its torque and speed values; the generator, that converts the energy into the electrical form; a converter and a transformer that connect the turbine to the grid; finally the controller adjusts the pitch angle and the generator torque in order to follow the power reference.

The block diagram of Figure 2.8 shows how the main components are connected each other and the input/output variables which indicate the relationship among the blocks.

2.2.1 The Turbine Model

The wind turbine model consists of four submodels: the wind model, the blade and pitch model, the drive train model and the generator model.

The wind is considered a stochastic process, with the additive contributions of the effects of wind shear and tower shadows. A complete description of the wind model is reported in (Dolan and Lehn, 2006). Therefore the wind speed v_w is the sum of four components:

$$v_w(t) = v_m(t) + v_s(t) + v_{ws}(t) + v_{ts}(t) \tag{2.6}$$

Where v_m is the mean speed, v_s is the stochastic component: a gaussian white noise, v_{ws} is the wind shear effect that takes into account the speed variation due to the distance from the earth surface, and v_{ts} is the tower shadow effect, that includes the loss of speed due to the passing of the blade in front of the tower. Actually, four wind speeds are involved in the benchmark model, one for the wind v_{hub} acting on the hub, and one for each of the three blades v_{w1}, v_{w2}, v_{w3}. It is worth noting that the wind shear and the tower shadow terms are considered only in the blade wind speeds.

For a specific blade i, the wind shear term can be written in analytical form as:

$$v_{ws_i}(t) = \frac{2v_m(t)}{3R^2}(\frac{R^3\alpha}{3H}\chi + \frac{R^4}{4}\alpha\frac{\alpha-1}{2H^2}\chi^2) + \frac{2v_m(t)}{3R^2}(\frac{R^5}{5}\frac{(\alpha^2-\alpha)(\alpha-2)}{6H^3}\chi^3) \quad (2.7)$$

Where: $\chi = \cos(\theta_{r_i})$, θ_{r_i} is the angular position of the i-th blade, R is the radius and α, H are two aerodynamic parameters.

The term relative to the tower effect is represented by the equation:

$$v_{ts_i}(t) = \frac{m\bar{\theta}_{r_i}(t)}{3r^2}(\Psi - \nu) \quad (2.8)$$

With:

$$\Psi = 2a^2\frac{R^2 - r_0^2}{(R^2 + r_0^2)\sin^2(\bar{\theta}_{r_i}(t)) + k^2}$$

$$\nu = 2a^2k^2\frac{(r_0^2 - R^2)r_0^2\sin(\bar{\theta}_{r_i}(t) + k^2)}{R^2\sin^2(\bar{\theta}_{r_i}(t)) + k^2}$$

$$m = 1 + \frac{\alpha(\alpha-1)r_0^2}{8H^2}$$

$$\bar{\theta}_{r_i}(t) = \theta_{r_i}(t) + \frac{2\pi(i-1)}{3} - \text{floor}(\frac{\theta_{r_i}(t) + \frac{2\pi(i-1)}{3}}{2\pi})2\pi$$

In which r_0 is the radius of the blade hub, k and a are two aerodynamic parameters.

The so-called blade and pitch model is fed by the wind speed and it is based on the aerodynamic law 2.4. The final value of the aerodynamic torque at the rotor shaft is equally provided by the three blade, hence:

$$\tau_r(t) = \sum_{i=1}^{3}\frac{\rho\pi R^3 C_q(\lambda(t), \beta(t))v_w^2(t)}{6} \quad (2.9)$$

The actual value of the blade pitch angle β differs from the reference signal β_r provided by the controller by means of a second order transfer function:

$$\frac{\beta(s)}{\beta_r(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (2.10)$$

Where ζ is the damping coefficient, ω_n is the natural pulsation. It represents a commonly adopted solution in the modeling of hydraulic piston servo systems. In nominal conditions the parameters of the transfer function have the same value for each of the three blades.

The drive train model describes the power flow through the gear box from the rotor to the generator. The gear box involves a torque decrement and a speed increment on the generator shaft. The following two-mass first order differential equations represent the drive train model:

$$J_r\dot{\omega}_r = \tau_r(t) - K_{dt}\theta_\Delta(t) - (B_{dt} + B_r)\omega_r(t) + \frac{B_{dt}}{N_g}\omega_g(t) \tag{2.11}$$

$$J_g\dot{\omega}_g = \frac{\eta_{dt}K_{dt}}{N_g}\theta_\Delta(t) + \frac{\eta_{dt}B_{dt}}{N_g}\omega_r(t) - (\frac{\eta_{dt}B_{dt}}{N_g^2} + B_g)\omega_g(t) - \tau_g(t) \tag{2.12}$$

$$\dot{\theta}_\Delta(t) = \omega_r(t) - \frac{1}{N_g}\omega_g(t) \tag{2.13}$$

Where: J_r,J_g are the moments of inertia of the shafts, K_{dt} is the torsion stiffness, B_{dt} is the torsion damping factor, B_g is the generator shaft viscous friction, N_g is the gear ratio, η_{dt} is the efficiency coefficient, and θ_Δ is the torsion angle.

The dynamics of the generator and the converter is described by means of a first order transfer function:

$$\frac{\tau_g(s)}{\tau_{gr}(s)} = \frac{\alpha_g}{s + \alpha_g} \tag{2.14}$$

The torque reference signal τ_{gr} comes from the controller, α_g is a generator parameter.

Finally, the generated power is computed as the product of the torque and the speed of the generator, decreased by its efficiency η_g:

$$P_g(t) = \eta_g\omega_g(t)\tau_g(t) \tag{2.15}$$

2.2.2 The Controller Model

The controller has the main task to regulate the power generated by the turbine on the basis of the wind speed. Figure 2.9 depicts the reference power curve, that can be divided into four different regions. In the first region ($v_w < 3\,\text{m/s}$) the wind speed is too low to generate power and the turbine is in idle state. The second region ($3\,m/s < v_w < 12,5\,m/s$) can be defined as *partial load*, or power optimization region as the controller forces the turbine power to its maximal value for that wind speed range. When the wind speed reaches the value of $12,5\,m/s$ the maximal power corresponds to the nominal turbine power of 4,8 MW, so in the third region ($12,5\,m/s < v_w < 25\,m/s$) the controller keeps the generated power constant. In the fourth region ($v_w > 25\,m/s$) the wind speed is too high and the turbine is stopped to prevent damage. More detail on the common industrial wind turbine controllers can be found in (Johnson et al., 2006).

In the partial load region, the optimum power value is obtained by keeping the blade pitch angles constantly equal to zero: $\beta = 0\,\text{deg}$ and by controlling the generator torque reference τ_{gr} to its optimal value in order to get the optimal tip speed ratio λ_{opt}, with reference to Eq. 2.2. This value is found by means of the

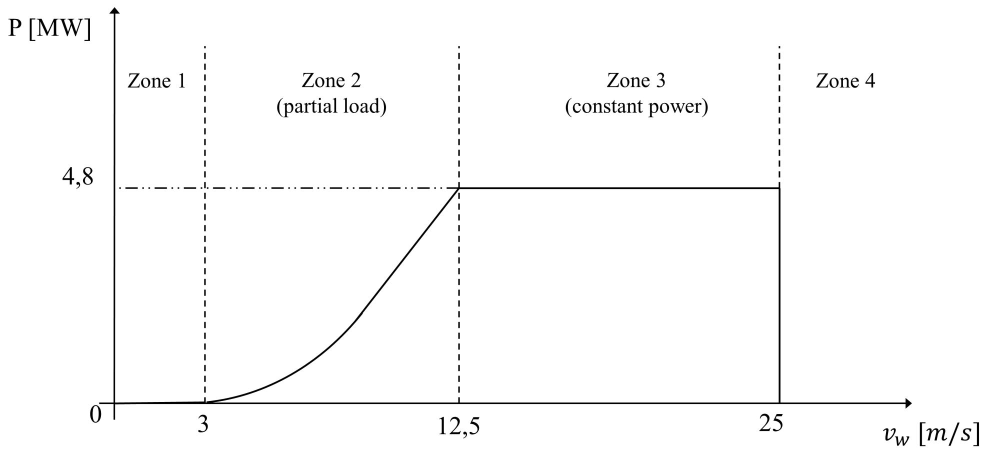

Figure 2.9: The reference power curve representing the target of the controller, with four different functioning regions, depending on the current wind speed.

C_P map, as the tip speed ratio can act on the maximal power coefficient $C_{P_{max}}$, after fixing the pitch angle to zero. The discrete reference torque signal becomes:

$$\tau_{gr}[k] = \frac{1}{2}\rho A R^3 \frac{C_{P_{max}}}{\lambda_{opt}^3}\left(\frac{\omega_g[k]}{N_g}\right)^2 \tag{2.16}$$

Where A is the area swept by the blades. In the constant power region the controller is switched to a PI regulator that acts on the pitch angles of the blades forcing the generator speed to its nominal value ω_{nom}, that implies the generation of the nominal power. In this region the discrete controller law is:

$$\beta_r[k] = \beta_r[k-1] + K_p e[k] + (K_i T_s - K_p)e[k-1] \tag{2.17}$$

Where K_p, K_i are the PI coefficients, $e[k]$ is the error on the generator speed.

Finally the switching logic of the controller is regulated by means of a bumpless mechanism: indeed, the condition for the switch from Region 2 to Region 3 is: $P_g[k] > P_r[k]$ AND $\omega_g[k] > \omega_{nom}$, while from Region 3 to Region 2 is: $\omega_g[k] < \omega_{nom} - \omega_\Delta$, with ω_Δ an hysteresis offset (Odgaard et al., 2013).

2.2.3 The Measurement Model

The measurements available to the controller come directly from several sensors or, in one case, they are obtained via estimation. In particular, for each of the three blades, a redundant couple of sensors measures the current pitch angle. Then, a couple of sensors measures the speed of the rotor and another one the speed of the generator, while a single sensor is available for the wind speed at hub height and another one for the generator torque. The wind torque measurements are estimated exploiting the hub anemometer. Table 2.1 reports a summary of the measured variables. The model of the measurements consists in the sum of the actual value with a white Gaussian noise.

Table 2.1: The measurements that are available to the controller.

Variable	Description
$\beta_{1,m1}$	Blade 1 pitch angle measurement, from the first sensor
$\beta_{1,m2}$	Blade 1 pitch angle measurement, from the second sensor
$\beta_{1,m1}$	Blade 2 pitch angle measurement, from the first sensor
$\beta_{1,m2}$	Blade 2 pitch angle measurement, from the second sensor
$\beta_{1,m1}$	Blade 3 pitch angle measurement, from the first sensor
$\beta_{1,m2}$	Blade 3 pitch angle measurement, from the second sensor
$\omega_{r,m1}$	Rotor shaft speed measurement, from the first encoder
$\omega_{r,m2}$	Rotor shaft speed measurement, from the second encoder
$\omega_{g,m1}$	Generator shaft speed measurement, from the first encoder
$\omega_{g,m2}$	Generator shaft speed measurement, from the second encoder
$\tau_{g,m}$	Generator shaft torque measurement
$P_{g,m}$	Generated power measurement
$v_{w,m}$	Wind Speed measurement at hub height, from the anemometer
$\tau_{r,m}$	Aerodynamic rotor torque, estimated from $v_{w,m}$

2.2.4 The Fault Scenarios

In the benchmark model three kinds of actual faults can be simulated: namely sensor, actuator and system faults. They are modeled as additive or multiplicative faults and they involve different degrees of severity, so that they can yield to the turbine shutdown in case of serious fault, or they can be accommodate by the controller if the risk for the system safety is low.

Regarding the considered sensor faults, they affect the measurements of the pitch angles and the measurements of the rotor speed, in form of a fixed value or a scaling error. They represent a common fault scenario of wind turbines, but their severity is low and they should be easy to identify and accommodate. In particular, an electrical or mechanical faults in the pitch sensors, if not handled, results in the generation of a wrong pitch reference system by the controller with the consequence of a loss in the generated power. The speed of the rotor is measured by means of two redundant encoders, an offset faulty signal can affect these measurements when the encoder does not detect the updated marker, while a gain factor faulty signal represents the reading of excessive markers each loop, due to dirt on the rotating part.

The considered actuator faults are modeled either as a fixed value or a changed dynamics of the transfer function. They affect the converter torque actuator as well as the pitch actuator. In the former case, the fault is located in the electronics of the converter, while in the latter case the fault is on the hydraulic system: it models the pressure drop in the hydraulic supply system (e.g. due to a leakage in hose or a blocked pump) or the excessive air content in the oil that causes the variation of the compressibily factor. The severity of these fault is of medium/high level.

Finally, the considered system fault concerns the drive train, in form of a slow variation of the friction coefficient in time due to wear and tear (months or year, but for benchmarking reason in the model it has been accelerated up to some seconds). It results in a combined faulty signal affecting the rotor speed and the generator speed. It can be listed as an high severe fault, as it can yield to the breakdown of the drive train, but in a long time. More details on these kind of fault and on its diagnosis at gear-box level are reported in (Hameed et al., 2009).

Table 2.2 shows the considered faults, with a brief description.

Table 2.2: The faults scenarios provided by the turbine benchmark model.

Fault	Description	Type
1	Fixed value of the blade 1 pitch sensor 1	Sensor fault
2	Scaling error of the blade 2 pitch sensor 2	Sensor fault
3	Fixed value of the blade 1 pitch sensor 1	Sensor fault
4	Fixed value of the rotor speed sensor 1	Sensor fault
5	Combined scaling error of the rotor speed sensor 2 and the generator speed sensor 2	Sensor fault
6	Pitch system changed response for the pitch actuator of the blade 2 due to air content in oil	Actuator fault
7	Pitch system changed response for the pitch actuator of the 3 due to low pressure	Actuator fault
8	Fixed value of the converter torque control signal	Actuator fault
9	Changed dynamics of the drive-train	System fault

2.2.5 Model Parameters

The parameters adopted in the benchmark model are summarized in Table 2.3. It is worth noting that some of these parameter values will be change in order to assess the FDI scheme.

Table 2.3: The value of the benchmark model parameters

Parameter	Value
R	$57,5\mathrm{m}$
ρ	$1{,}225\,\mathrm{Kg\,m^{-3}}$
ζ	0.6
ω_n	$11{,}11\,\mathrm{rad\,s^{-1}}$
B_{dt}	$775{,}49\,\mathrm{N\,m\,s\,rad^{-1}}$
B_r	$7{,}11\,\mathrm{N\,m\,s\,rad^{-1}}$
B_g	$45{,}6\,\mathrm{N\,m\,s\,rad^{-1}}$
N_g	95
K_{dt}	$2{,}7 \times 10^9\,\mathrm{N\,m\,rad^{-1}}$
η_{dt}	$0{,}97$
J_g	$390\,\mathrm{Kg\,m^2}$
η_g	$0{,}98$
α_g	$50\,\mathrm{rad\,s^{-1}}$
K_i	1
K_p	4
ω_{nom}	$162\,\mathrm{rad\,s^{-1}}$
P_{ref}	$4{,}8\,\mathrm{MW}$

2.2.6 The Complete Model

With these assumptions, the complete model of the system under analysis (Simani et al., 2015c) can be represented by means of a non-linear continuous-time function $\mathbf{f}_{wt}$, that describes the evolution of the turbine state vector $\mathbf{x}_{wt}$ excited by the input vector $\mathbf{u}$:

$$\begin{cases} \dot{\mathbf{x}}_{wt}(t) = \mathbf{f}_{wt}(\mathbf{x}_{wt}, \mathbf{u}(t)) \\ \mathbf{y}(t) = \mathbf{x}_{wt}(t) \end{cases} \tag{2.18}$$

Where the state of the system is considered equal to the monitored system output i.e. the rotor speed, the generator speed and the generated power: $\mathbf{x}_{wt}(t) = \mathbf{y}(t) = [\omega_{g,m1}, \omega_{g,m2}, \omega_{r,m1}, \omega_{r,m2}, P_{g,m}]$.

The input vector $\mathbf{u}(t) = [\beta_{1,m1}, \beta_{1,m2}, \beta_{2,m1}, \beta_{2,m2}, \beta_{3,m1}, \beta_{3,m2}, \tau_{g,m}]$ contains the measurements of the pitch angles from the three sensor couples as well as the measured torque. These vectors are sampled for obtaining a number of N input-output data $\mathbf{u}[k], \mathbf{y}[k]$ with $K = 1, ..., N$, in order to implement the data-driven estimators.

2.3 The Wind Farm Benchmark System

The wind farm benchmark model considered in this thesis has been proposed in (Odgaard and Stoustrup, 2013) by the same authors of the wind turbine benchmark model. It consists of nine wind turbines arranged in a squared grid of three rows and three columns. The distance between two adjacent turbines is seven times the rotor diameter R. Two measuring masts (anemometers) are placed in front of the first line of turbines, at a distance of ten times R, providing the measurements of the undisturbed wind speed. The considered turbines are 4,8 MW three-blades horizontal axis turbines, represented by a simpler model, with respect to the previously described turbine. Each of them is provided with a controller, but also a wind farm controller is included in the benchmark model. Three common fault scenarios can be simulated. The complete wind farm model consists of three main submodels: the wind and wake model, the plant model, and the controller model, interacting as in Figure 2.10.

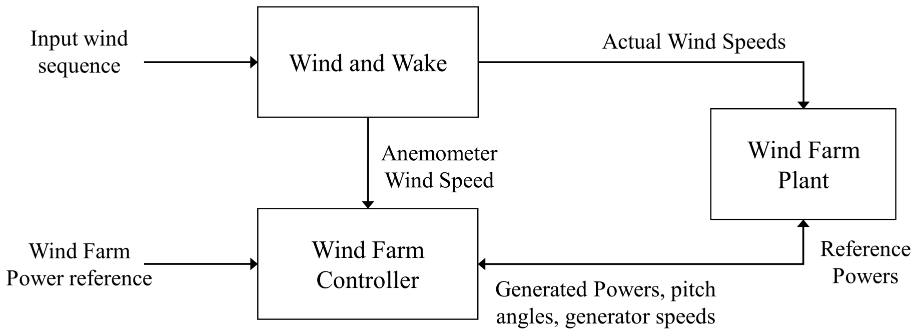

Figure 2.10: The subsystems of the wind farm benchmark model.

2.3.1 The Wind and Wake Model

The wind and wake model provides the wind speed for each of the nine turbines, contained in the vector $\mathbf{v}_w$, as well as for a measuring mast $v_{w,m}$. They are determined starting from a certain wind sequence (two different wind sequences are included in the simulator) and their elaboration takes into account the delay and the interaction among the turbines depending on wind direction. In particular the wake is described as reported in (Jensen, 1983) by means of a static deficit coefficient of 0.9. Finally the turbulence is modeled by an additive Gaussian white noise.

2.3.2 The Plant Model

The plant model represents the nine wind turbines with the same submodel for each of them. It receives as input the $\mathbf{v}_w$ vector and the $\mathbf{P}_r$ vector containing the nine reference signals from the controller. The outputs are the vectors $\mathbf{P}_g$, $\boldsymbol{\beta}$, $\boldsymbol{\omega}_g$ that contain respectively the generate powers, the pitch angles, the generator speeds, for each of the nine turbines. Inside the turbine submodel, the current wind speed is elaborated by means of a look-up table in order to compute the available power $P_w(t)$. Then, the generated power is computed as:

$$P_g(t) = P_c(t) + \gamma_p \sin(2\pi\sigma_p t) \tag{2.19}$$

Where the first term $P_c(t)$ is equal to the current lower value between the filtered available power $\hat{P}_w$ and the filtered reference power $\hat{P}_r$:

$$P_c(t) = \begin{cases} \hat{P}_w(t), & \text{if } \hat{P}_w(t) < \hat{P}_r(t), \\ \hat{P}_r(t), & \text{if } \hat{P}_w(t) > \hat{P}_r(t). \end{cases}$$

The second term of the Equation 2.19 represents the oscillations caused by the drive train, whose amplitude is γ_p and whose frequency is σ_p.

The filtered signals $\hat{P}_w(t)$ and $\hat{P}_r(t)$ differs from the input variables by means of a first order transfer function:

$$\hat{P}_w(s) = \frac{\tau_w(v_w)}{s + \alpha_w(v_w)} P_w(s)$$

$$\hat{P}_r(s) = \frac{\tau_p}{s + \tau_p} P_r(s)$$

Where the parameter τ_p is a fixed value, while the parameters τ_w and α_w depends on the wind speed and are computed by means of a look-up table.

Regarding the pitch angle output, the model is similar to that of the turbine benchmark model of Eq. 2.10, but the transfer function between the reference pitch signal and the actual pitch angle has been reduced to a first order transfer function:

$$\beta(s) = \frac{\tau_\beta}{s + \tau_\beta} P_r(s) \tag{2.20}$$

Then, The generator speed of each turbine is modeled as:

$$\omega_g(t) = f_\omega(P_c(t))\big(1 + \frac{\gamma_\omega}{\omega_{g,max}} \sin(2\pi\sigma_p t)\big) \tag{2.21}$$

Where f_ω is computed by means of a look-up table and the oscillation term, due to the drive train, has an amplitude equal to the ratio between the parameter γ_ω and the maximal generator speed $\omega_{g,max}$.

Finally, the wind farm controller forces each turbine to follow a reference power signal $P_r[k]$ that is one-ninth of the wind farm power reference. Moreover, in order to avoid fast variation of the control signal, the wind farm power reference is low-pass filtered obtaining $\hat{P}_{wf,r}$. The controller is discrete-time modeled and uses a sample frequency of 10 Hz (Odgaard and Stoustrup, 2013).

$$P_r[k] = \frac{1}{9}\hat{P}_{wf,r}[k] \tag{2.22}$$

2.3.3 The Fault Scenarios

Three common fault scenarios are implemented. They affect the output variables of the plant system. These kinds of fault are difficult to detect considering the single wind turbine, but they can be identified at wind farm level.

The first fault considered represents the debris build-up on the blade surface. Its effect is a change of the aerodynamic of the affected turbine and the consequent decreasing of the generated power that is modeled by a scaling factor of 0.97 applied to the generated power signal. An analysis of this kind of fault is reported in (Johnson et al., 2006).

The second fault is a misalignment of one blade caused by an imperfect installation. The effect is an offset between the actual and the measured pitch angle of the affected turbine. This fault can excite structural modes and creates undesired vibrations that can damage the system severely. The faulty signal involves an offset of 0.3 deg on the pitch angle.

The third fault represents the wear and tear in the drive train. it has been demonstrated (Odgaard and Stoustrup, 2012) that this kind of fault is difficult to detect at wind turbine level, and the current trend is to analyze the frequency spectra of different vibration measurement. In this benchmark model the fault affects the generated power increasing the amplitude of its oscillation of the 26% of the nominal value, and the generator speed increasing the amplitude of its oscillation of the 130%.

2.3.4 Model Parameters

Table 2.4 shows the parameters used in the wind farm benchmark model.

Table 2.4: The value of the wind farm benchmark model parameters

Parameter	Value
R	$57,5$m
τ_p	$1{,}2\,\mathrm{rad\,s^{-1}}$
γ_p	$1000\,\mathrm{W}$
σ_p	$10\,\mathrm{Hz}$
τ_β	$1{,}6\,\mathrm{rad\,s^{-1}}$
$\omega_{g,max}$	$158\,\mathrm{rad\,s^{-1}}$
γ_ω	0.4

Chapter 3

Data-Driven Modeling and Identification

In this chapter, the strongly nonlinear systems that characterize the behavior of the wind turbine and of the wind farm, summarized in Chapter 2 by Eq. 2.18, is modeled by means of data-driven strategies, in order to build up the fault diagnosis blocks as explained in Chapter 1. Two data-driven modeling approaches are introduced, relying on fuzzy logic and artificial neural networks.

3.1 Fuzzy Modeling and Identification

In this section, after a brief introduction on fuzzy logic, the design of the dynamic estimators by means of the Takagi-Sugeno (TS) models is described. Indeed, the unknown relationships between noisy measurements and faults are provided by fuzzy models, which consist in a number of rules connecting the inputs with the output of the system under investigation, on the basis of a knowledge of its dynamics in form of IF $\Longrightarrow$ THEN relations, processed by a fuzzy reasoning (Babuška, 2012).

3.1.1 Introduction to Fuzzy Logic as Numerical Approximation of Linguistic Proposition

A linguistic reasoning commonly is not characterized by a precise numerical meaning as the classic crisp logic would require. Fuzzy logic proposes an effective tool to approximate an uncertain reasoning.

Indeed, a binomial crisp reasoning deals with two concepts: *equal to* or *not equal to*, whilst fuzzy logic introduces the *degree of fulfillment* as the similarity between the concept under analysis and a defined prototype used as basis of comparison. Therefore, the degree of fulfillment can be expressed by any real number between the two crisp values 0 and 1. The belonging of an item x to a set A is no more a dichotomous yes/no issue, but it can be represented by the so-called membership

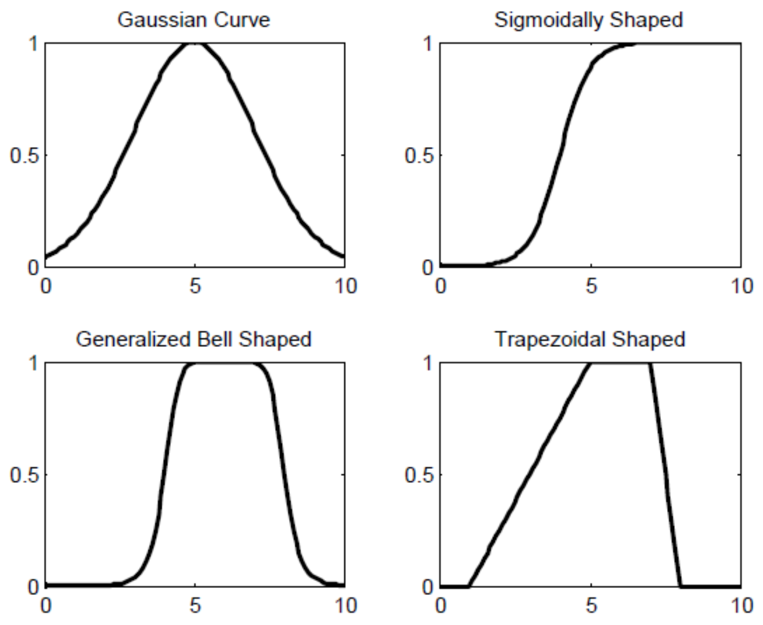

Figure 3.1: Four meaningful membership functions: Gaussian, Sigmoidal, Generalized Bell, and Trapezoidal.

function μ defined as:

$$\mu_A(x) : X \longrightarrow (0, 1)$$

Where X is the domain of x.

Some common membership function shapes are depicted in Figure 3.1, while Table 3.1 shows the number of required parameters for each function.

A Fuzzy Inference System (FIS) is the engine of the fuzzy reasoning, which provides the output as a consequence of the input condition. So, if properly designed, it can model the input-output relations of a system whose knowledge is expressed by linguistic propositions, instead of mathematical laws. Generally speaking, considering a Multi-Input Multi-Output (MIMO) system, with $\mathbf{x} \in R^{n_i}$ input vector and $\mathbf{y} \in R^{n_o}$ output vector, the FIS processing involves three main steps:

- **fuzzyfication**: the computation of the degree of fulfillment of the input;
- **reasoning**: the inference process, that provides the conditioned membership functions of the output;
- **defuzzification**: the computation of the output value.

In detail, the fuzzification process consists in the computation of the degree of fulfillment of all the n_i measured inputs x_i with $i = 1...n_i$ referred to the all the considered predefined n_s fuzzy sets in which the domain X has been subdivided. The result is achieved by means of the membership functions $\mu_j(x_i)$ with $j = 1...n_s$. It is worth noting that a fuzzy set can represent a linguistic term. As an example, the linguistic term *low* can be associated to a fuzzy set which the *speed* domain

Table 3.1: Some commonly used membership functions, with the mathematical expression and the number of required parameters

Name	Function	Number of Parameters
Gaussian	$f(x) = e^{\frac{-(x-c)^2}{2\sigma^2}}$	2
Sigmoidally shaped	$\frac{1}{1+e^{-a(x-c)}}$	2
Generalized bell-shaped	$f(x) = \frac{1}{1+\frac{x-c}{a}^2}$	3
Trapezoidal shaped	$f(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c \le x \le d \\ 0, & x > d \end{cases}$	4

consists of. So the proposition *the speed is low* can be fuzzified into a membership function that implies high values (about one) for low speed measurement and a value close to zero otherwise. Then, the fuzzy reasoning can be carried out, based on a number of rules R in the general form of:

$R : IF$ (fuzzy combination of inputs) $THEN$ (fuzzy combination of outputs)

Where the antecedents and the consequents indicate the combined membership functions of the inputs and the combined membership function of the outputs with reference to the fuzzy set (or *cluster* as denominated in the following). It is worth observing that, unlike the input, the output is still unknown but all the membership functions are defined for the input and for the output as well.

The combination of inputs (or outputs) membership functions can be achieved either by means of the union or the intersection among the members, depending on which one better characterizes the inputs relations for the relative rule. The so called t-norm represents the crisp logic AND operator. It can be performed though different methods, as reported in Eq. 3.1

$$\mu_A(x) = t(\mu(x_1), \mu(x_2)) = \begin{cases} \min(\mu(x_1), \mu(x_2)) & \text{intersection} \\ \mu(x_1)\mu(x_2) & \text{product} \\ \max(0, \mu(x_1) + \mu(x_2) - 1) & \text{Lucasiewicz t-norm} \end{cases} \tag{3.1}$$

Otherwise, the OR operator is realized by the s-norm of Eq. 3.2

$$\mu_A(x) = s(\mu(x_1), \mu(x_2)) = \begin{cases} \max(\mu(x_1), \mu(x_2)) & \text{union} \\ \mu(x_1) + \mu(x_2) - \mu(x_1)\mu(x_2) & \text{probabilistic sum} \\ \min(1, \mu(x_1) + \mu(x_2)) & \text{Lucasiewicz s-norm} \end{cases} \tag{3.2}$$

Afterwards, the fuzzy implication expressed by the THEN operator has to be defined, such that the relations between the antecedent and the consequent can generate the rule output membership function $\mu_R(y)$. The THEN operator is described by a t-norm, defined in the Cartesian product of the input and output domains $X \times Y$:

$$\mu_R(y) = t(\mu_A(x), \mu_B(y)) = \begin{cases} \min(\mu_A(x), \mu_B(y)) & \text{Mamdani implication} \\ \mu_A(x)\mu_B(y) & \text{product} \\ \max(1-\mu_A(x), \mu_B(y)) & \text{Kleene implication} \\ \min(1, 1-\mu_A(x)+\mu_B(y)) & \text{Lucasiewicz implication} \end{cases} \tag{3.3}$$

Normally, more than one rule R_1,R_2,etc. are required to model the behavior of a complex system. So the ELSE operator can be introduced:

$$\begin{aligned} &R_1 : IF \quad \mu_{A_1}(x) \quad THEN \quad \mu_{B_1}(y) \quad ELSE, \\ &R_2 : IF \quad \mu_{A_2}(x) \quad THEN \quad \mu_{B_2}(y) \quad ELSE... \end{aligned}$$

The s-norm is the suitable tool for representing the ELSE operator, as it describes the degree of alternative among the rules.

The last step of the fuzzy reasoning is the defuzzification process, that involves the generation of the estimated i-th system output $\hat{y}$, with $i = 1, ...n_o$, given the inferred membership function $\mu_R(y_i)$. The center of gravity of the membership function is a suitable approximation value:

$$\hat{y} = \frac{\int y_i \mu_R(y_i) dy_i}{\int \mu_R(y_i) dy_i} \tag{3.4}$$

This reasoning approach was proposed by Mamdani in (Mamdani, 1977). It is characterized by antecedents and consequents that are both fuzzy propositions (represented by membership function). It is usually adopted to model systems which are commonly described by linguistic rules exploiting linguistic terms and variables, as the membership functions can cope with the uncertainty of expressed by a linguistic preposition.

3.1.2 Takagi-Sugeno Fuzzy Rules

The approximation of nonlinear Multi-Input Single-Output (MISO) systems (but also extension to MIMO systems can be considered) is usually better achieved by the Takagi-Sugeno (TS) fuzzy reasoning, as reported in (Fantuzzi and Rovatti, 1996) and (Rovatti, 1996). According to TS approach, proposed in (Takagi and Sugeno, 1985), the consequents become crisp functions of the input, while the antecedents remain fuzzy propositions, therefore the fuzzy rule takes the form of:

$$R_i : IF \quad \text{(fuzzy combination of inputs)} \quad THEN \quad \text{output} = f_i(\text{inputs}) \tag{3.5}$$

The antecedent does not differ from the Mamdani rules, with a combined membership function $\lambda_i(\mathbf{x})$ that takes into account the logical connectives expressed by

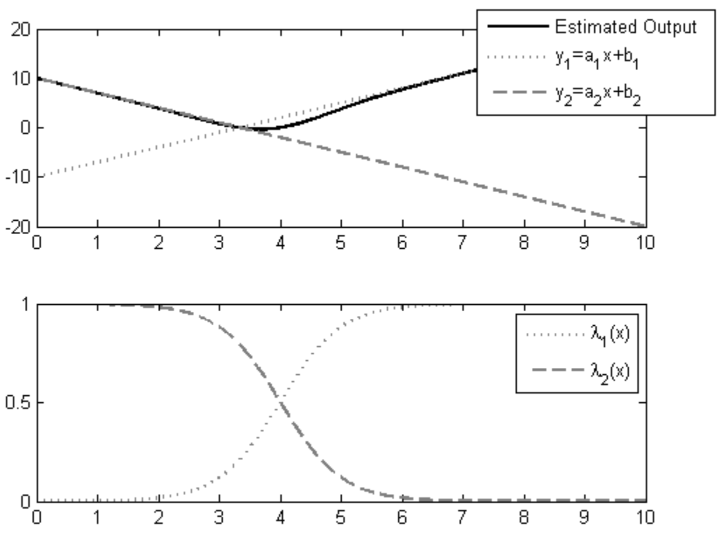

Figure 3.2: The output of a TS FIS, characterized by two rules, with consequent parameters $a_1 = -3, b_1 = 10, a_2 = 3, b_2 = -10$. The antecedent membership functions λ_1, λ_2 are sigmoidally shaped

linguistic propositions. The rule consequent function f_i have a defined structure: it is instance of parametrized function in the affine linear form:

$$y_i = \mathbf{a}_i^T \mathbf{x} + b_i \tag{3.6}$$

Where $\mathbf{a}_i$ is a parameter vector and b is a scalar offset, while y_i is the i-th rule output. The number of rules is supposed equal to the number of clusters n_C. Furthermore, the antecedent of each rule defines the degree of fulfillment for the corresponding consequent model, so that the rule global model can be seen as a fuzzy composition of linear local models.

Thus, the TS inference takes the form of the simple algebraic expression of Eq. 3.7:

$$\hat{y} = \frac{\sum_{i=1}^{n_C} \lambda_i(\mathbf{x}) y_i}{\sum_{i=1}^{n_C} \lambda_i(\mathbf{x})} \tag{3.7}$$

The estimated output $\hat{y}$ is the weighted average of linear functions of the measured input, where the weights are the combined degree of fulfillment of the system input. An example of TS models as nonlinear function approximator is depicted in Fig. 3.2 for a Single-Input Single-Output (SISO) system, with $n_C = 2$.

It is worth noting that the nonlinear system under investigation can have either a static or a dynamic behavior: in the latter case, the considered model input vector $\mathbf{x}$ can contains current as well as previous samples of the system input or output. Indeed, in order to introduce the time dependence into the model of Eq. 3.5 the consequents are considered as discrete-time linear AutoRegressive models with eXogenous input (ARX) of order o, in which the regressor vector takes the form of:

$$\mathbf{x}(k) = [y(k-1), ..., y(k-o), u(k), ..., u(k-o)]^T \tag{3.8}$$

Where u and y are the actual system input and output vectors, and k is the time step. The affine parameters of Eq. 3.6 can be grouped in:

$$\mathbf{a}_i = [\alpha_1^{(i)}, ..., \alpha_o^{(i)}, \delta_1^{(i)}, ..., \delta_o^{(i)}]^T \tag{3.9}$$

Where the $a^{(i)}$ coefficients are associated to the output samples, and the $\delta^{(i)}$ are associated to the input.

3.1.3 FIS Design from Data

An effective approach to the design of a FIS as approximator of a complex nonlinear system begins with the partitioning of the available data into subsets characterized by simpler (linear) behavior.

A cluster can be defined as a group of data that are more similar each other rather than to the members of another cluster. The similarity among data can be expressed in terms of their distance from a particular item, exploited as the cluster prototype. Fuzzy clustering provides an effective tool to obtain a partitioning of data in which the transitions among subsets are smooth, rather than abrupt. Indeed, fuzzy clustering allows an item to belong to several cluster simultaneously, with different degrees of fulfillment, whereas the classic crisp clustering relies on mutual exclusive subsets. Different clustering methods have been proposed in literature, see e.g. the review (Jain et al., 1999) or the more recent works (Jun et al., 2011) and (Graaff and Engelbrecht, 2012).

Typically, the available data consist in noisy measurements acquired from the system. They are grouped into the data matrix $\mathbf{Z}$, whose columns are the vectors z containing the measurements of a single observation of the system under analysis:

$$\mathbf{Z} = \begin{bmatrix} z_{11} & \cdots & z_{1N} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nN} \end{bmatrix} \tag{3.10}$$

Where n is the data dimension, N is the number of available observations.

Most fuzzy clustering algorithms are based on the optimization of the c-means goal function $J(\mathbf{Z}, \mathbf{U}, \mathbf{V})$ that depends on:

- $\mathbf{Z}$: the data matrix above defined;
- $\mathbf{U} = [\mu_{ik}]$: the so-called fuzzy partition matrix, that contains the values of the membership function for the couple i-th measurement/k-th cluster;
- $\mathbf{V} = [\mathbf{v}_1 ... \mathbf{v}_{n_C}]$, containing the cluster prototypes, that have to be determined and that represent the centers from which the distance of each measurement can be calculated.

The widespread c-means goal function adopted in this work was formulated in (Bezdek, 2013):

$$J(\mathbf{Z}, \mathbf{U}, \mathbf{V}) = \sum_{i=1}^{n_C} \sum_{k=1}^{N} (\mu_{ik})^m D_{ik\mathbf{A}}^2 \tag{3.11}$$

With $m > 1$ weighting exponent, and

$$D^2_{ik\mathbf{A}} = \|\mathbf{z}_k - \mathbf{v}_i\|^2_{\mathbf{A}} = (\mathbf{z}_k - \mathbf{v}_i)^T \mathbf{A} (\mathbf{z}_k - \mathbf{v}_i), \quad i = 1, ..., n_C, \quad k = 1, ..., N \tag{3.12}$$

that is a squared inner product distance norm. The matrix $\mathbf{A}$ determines the cluster shape.

The minimization algorithm exploits a series of Picard iterations consisting in the updating of the cluster prototypes and of the partition matrix, until the stop criterion is met. A brief summary of the algorithm steps is reported in the following.

1. **Initialization**: set up $\mathbf{Z}$ from acquired measurements and fix the value of n_C, m, $\mathbf{A}$ and of the termination threshold ϵ. Then, initialize randomly the partition matrix $\mathbf{U}$;

2. **Computation of cluster prototypes**: start the iteration computing the cluster prototypes as the weighted average of observations:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} \mu_{ik}^m \mathbf{z}_k}{\sum_{k=1}^{N} \mu_{ik}^m}, \quad i = 1, ..., n_C \tag{3.13}$$

3. **Computation of distances**: according to Eq. 3.12 calculate the distance of each observation vector from the cluster prototypes;

4. **Updating the partition matrix**: the degrees of fulfillment can be updated by means of the expression:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{n_C} \left(\frac{D_{ik\mathbf{A}}}{D_{jk\mathbf{A}}}\right)^{\frac{2}{m-1}}} \tag{3.14}$$

5. **Check the termination criterion**: if the norm of the difference between the current and the previous partition matrix is bigger than ϵ, then start another iteration from step 2.

An important point concerns the determination of the optimal number of clusters n_C, as the clustering algorithm operates on the assumption of a certain number of clusters, regardless of whether they are really present in the data or not. Once the partition matrix has been estimated, the antecedent degrees of fulfillment are easily derive by interpolation or curve fitting methods.

Then, the design of the FIS assumes the form of an identification problem addressed to the estimation of the consequent parameters $\mathbf{a}_i$ and b_i of Eq. 3.6 in a noisy environment. The identification scheme adopted in this work was proposed in (Simani et al., 1999) and successfully exploited in the approximation of nonlinear functions through the piecewise affine models (Fantuzzi et al., 2002). This approach consists in the minimization of the prediction errors of the individual TS local models understood as n_C independent problems. Their solutions rely on the so-called Frisch scheme (Beghelli et al., 1990).

Considering a discrete-time MISO system, the noise is supposed to affect the input $\mathbf{u}$ as well as the output y measurements in form of the additive signals $\tilde{\mathbf{u}}$, $\tilde{y}$ on the noise-free unmeasurable quantities $\mathbf{u}^*$, y^* :

$$\begin{aligned} \mathbf{u}(k) &= \mathbf{u}^*(k) + \tilde{\mathbf{u}}(k) \\ y(k) &= y^*(k) + \tilde{y}(k) \end{aligned} \tag{3.15}$$

Thus, considering the i-th TS consequent of the type of Eq. 3.7 and the associated dynamic local ARX model of order o with the regressors grouped into the vector $\mathbf{x}$ as in Eq. 3.8, the acquisition of N_i noisy measurement of input and output samples permits the construction of the i-th data matrix $\mathbf{X}^{(i)}$ defined as:

$$\mathbf{X}^{(i)} = \begin{bmatrix} y(k) & \mathbf{x}^T(k) & 1 \\ y(k+1) & \mathbf{x}^T(k+1) & 1 \\ \vdots & \vdots & \vdots \\ y(k+N_i-1) & \mathbf{x}^T(k+N_i-1) & 1 \end{bmatrix} \tag{3.16}$$

The i-th covariance matrix $\mathbf{\Sigma}^{(i)}$ from the acquired data can be computed as:

$$\mathbf{\Sigma}^{(i)} = \mathbf{X}^{(i)^T}\mathbf{X}^{(i)} \geq 0 \tag{3.17}$$

That is a positive-definite matrix consisting in the sum of two terms:

$$\mathbf{\Sigma}^{(i)} = \mathbf{\Sigma}^{(i)^*} + \bar{\tilde{\mathbf{\Sigma}}}^{(i)} \tag{3.18}$$

Where $\mathbf{\Sigma}^{(i)^*}$ concerns the noise free signals, while $\bar{\tilde{\mathbf{\Sigma}}}^{(i)}$ is the noise covariance matrix, which depends on the unknown noise variances $\bar{\tilde{\sigma}}_{\mathbf{u}}$,$\bar{\tilde{\sigma}}_y$ through the expression:

$$\bar{\tilde{\mathbf{\Sigma}}}^{(i)} = \mathbf{diag}\big[\bar{\tilde{\sigma}}_y\mathbf{I}, \bar{\tilde{\sigma}}_{\mathbf{u}}\mathbf{I}, 0\big] \tag{3.19}$$

The solution of the identification problem above mentioned requires the estimation of $\bar{\tilde{\sigma}}_u$ and $\bar{\tilde{\sigma}}_u$, that can be performed solving the equation:

$$\mathbf{\Sigma}^{(i)^*} = \mathbf{\Sigma}^{(i)} - \tilde{\mathbf{\Sigma}}^{(i)} \tag{3.20}$$

with $\tilde{\mathbf{\Sigma}}^{(i)} = \mathbf{diag}\big[\tilde{\sigma}_y\mathbf{I}, \tilde{\sigma}_{\mathbf{u}}\mathbf{I}, 0\big]$, in the variables $\tilde{\sigma}_{\mathbf{u}}$,$\tilde{\sigma}_y$.

In case all the assumption regarding the Frisch scheme (Simani et al., 1999) are satisfied, there exists one common point belonging to all the surfaces $\mathbf{\Gamma}^{(i)} = 0$ determined as the root locus of Eq. 3.20, that represents the actual noise variance values $(\bar{\tilde{\sigma}}_{\mathbf{u}}, \bar{\tilde{\sigma}}_y)$. However, in real cases, the Frisch assumptions are commonly violated, so that a unique solution cannot be obtained. In these situations the identification aims at finding the nearest point of all the surfaces.

After the computation of the variances, the covariance noise matrix can be built as in Eq. 3.19, and the linear parameters in each cluster (therefore in each TS consequent) can be finally determined as a solution of the following expression:

$$\big(\mathbf{\Sigma}^{(i)} - \bar{\tilde{\mathbf{\Sigma}}}^{(i)}\big)\mathbf{a}_i = 0 \tag{3.21}$$

3.2 Neural Network Modeling and Identification

Alongside the fuzzy models, another data-driven approach, based on neural networks, has been proposed in order to implement the fault diagnosis block. In this section, after a brief introduction on the general structure, the properties, and the functioning of a neural network, the architecture of an open-loop Nonlinear AutoRegressive with eXogenous input (NARX) network is reported, as it represents, in combination with the backpropagation Levenberg-Marquardt training algorithm, the exploited solution for the implementation of the neural network fault estimators.

3.2.1 Introduction to Neural Network

The denomination *neural network* refers to its architectural analogy with the human brain. Indeed, a neural network consists of many interconnected elemental units, called *neurons*, each of which processes its input and produces its output in order to accomplish the final global task of the network. The way in which neurons are connected, and the elaboration that they execute, determine the typology of the network and its suitability to solve a particular problem.

The objective of a neural network can be summarized into the combination of the human brain properties with the processing rate and accuracy of electronics. Indeed, the brain is a highly complex, nonlinear and parallel information processing system, currently more effective than any computer in the field of pattern recognition, classification and decision making. However, transistors are thousand times faster than brain neurons and more precise in computation. The result is an artificial intelligence system, with data-driven learning and adaptation properties, capable of approximate any nonlinear multidimensional function.

In this work, a set of neural estimators is designed and trained in order to reproduce the behavior of the systems under investigation, thus accomplishing the modeling and identification task.

The structure of the i-th single neuron (Haykin et al., 2009), also called *perceptron*, is depicted in Fig. 3.3. It features a MISO system where the output y_i is computed as a function f of the weighted sum v_i of all the n_i neuron inputs $u_{i,1}...u_{i,n_i}$, with the associated weights $w_{i,1}...w_{i,n_i}$.

The function f, denominated *activation function*, represents the engine of the neuron. Although any differentiable function can be exploited as activation function, the most common choices are shown in Fig. 3.4 and their expression are reported in Table 3.2.

3.2.2 Neural Network Architectures

A structural categorization of neural networks concerns the way in which their elements are connected each others (Liu, 2012). In a *feedforward network*, also called *multilayer perceptron*, neurons are grouped into unidirectional layers, as shown in Fig. 3.5. The first of them, namely the *input* layer, is fed directly by

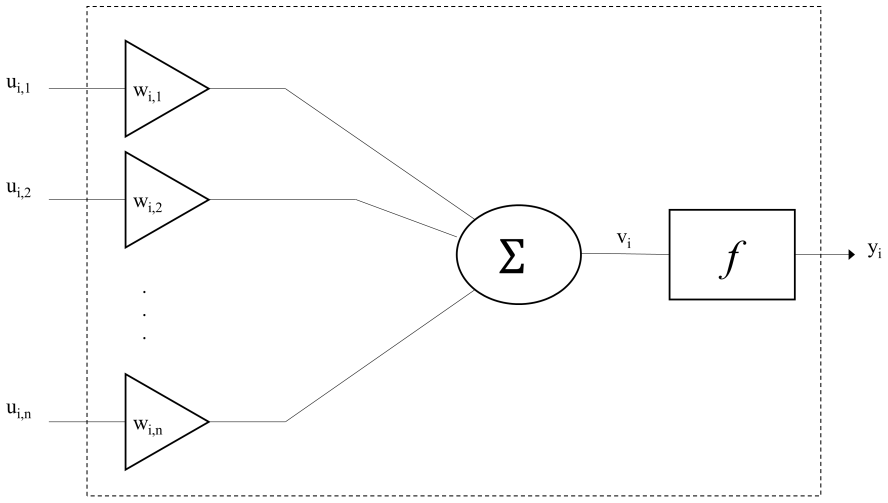

Figure 3.3: The model the i-th elemental neuron which forms the network.

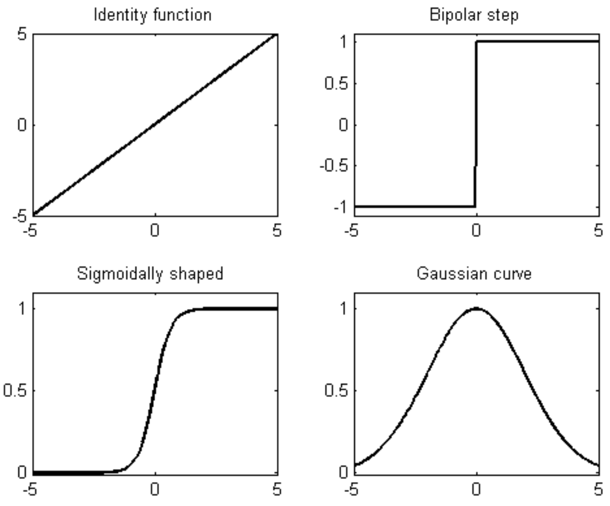

Figure 3.4: The shape of some common activation functions.

Table 3.2: Some commonly used activation function, with the mathematical expression and the number of required parameters

Name	Function	Number of Parameters
Identity	$f(x) = x$	0
Bipolar Step	$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$	0
Sigmoidally shaped	$\frac{1}{1+e^{-a(x-c)}}$	2
Gaussian	$f(x) = e^{\frac{-(x-c)^2}{2\sigma^2}}$	2

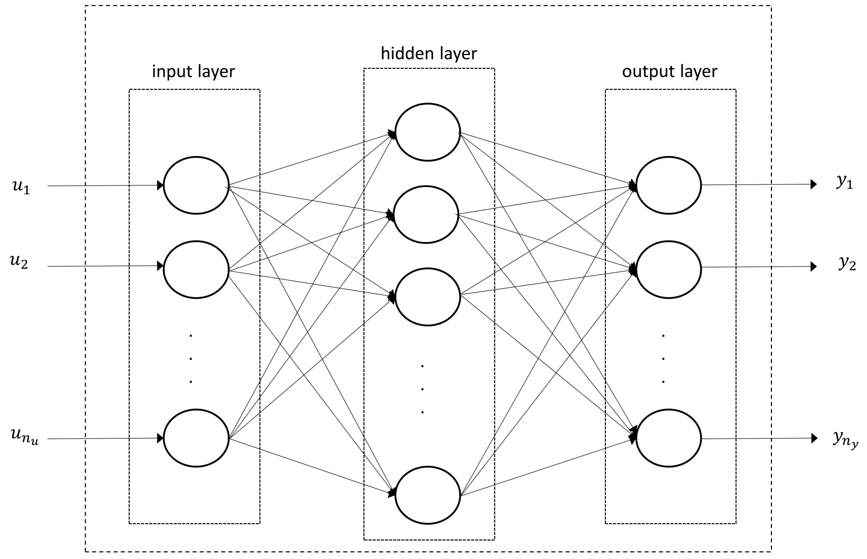

Figure 3.5: The architecture of a feedforward network, with the input layer, one hidden layer and the output layer.

the network inputs, then each successive *hidden* layer takes the inputs from the neurons of the previous layer and transmits the output to the neurons of the next layer, up to the last *output* layer, in which the final network outputs are produced. Therefore, neurons are connected from one layer to the next, but not within the same layer. The only constraint is the number of neurons in the output layer, that has to be equal to the number of actual network outputs. Often neurons of different layers can have different activation functions.

On the other side, *recurrent networks* (Medsker and Jain, 1999) are multilayer networks in which the output of some neurons is fedback to neurons belonging to previous layers, thus the information flow in forward as well as in backward directions allowing a dynamic memory inside the network.

A noteworthy intermediate solution is provided by the multilayer perceptron with a tapped delay line, which is a feedforward network whose inputs come from a delay line. This kind of network represents a suitable tool to model, or predict, the evolution of a dynamic system. in particular the open loop Nonlinear AutoRegressive with eXogenous inputs (NARX) network belongs to this latter category as its inputs are delayed samples of the system inputs and outputs. Indeed, if properly trained, a NARX network can estimate the current (or the next) system output on the basis of the acquired past measurements of system inputs and outputs.

Generally speaking, considering a MIMO system, the elaborations of the open-loop NARX network follow the law:

$$\hat{\mathbf{y}}(k) = f_{net}\big(\mathbf{u}(k)...\mathbf{u}(k-d_u), \mathbf{y}(k-1)...\mathbf{y}(k-d_y)\big) \tag{3.22}$$

Where $\hat{\mathbf{y}}$ is the estimation of the system output, $\mathbf{u}$ and $\mathbf{y}$ are the measured system inputs and outputs, k is the time step, d_u and d_y are the number of delay of

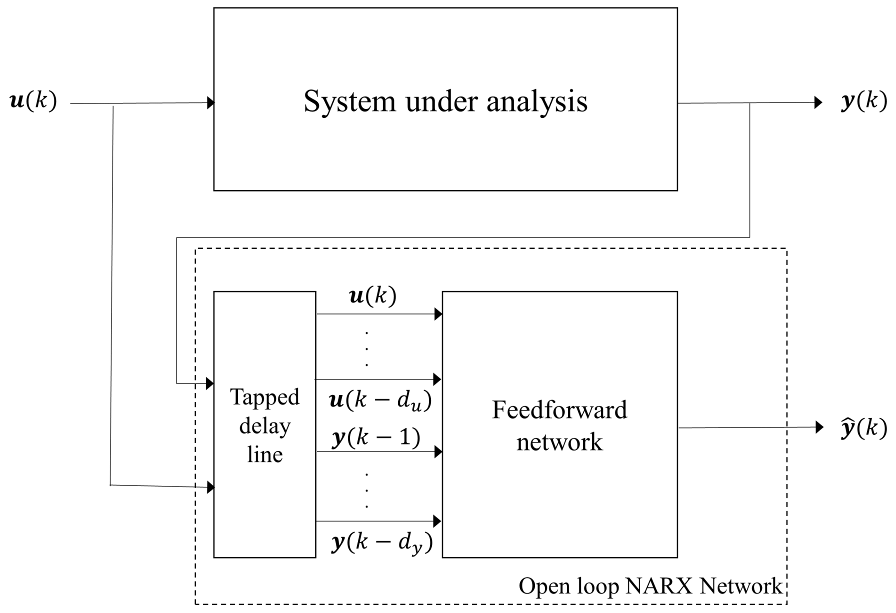

Figure 3.6: The open loop NARX network, that reconstructs the system output on the basis of the delayed measurements.

inputs and outputs, respectively. f_{net} is the function realized by the network, that depends on the layer architecture, the number of neurons, their weights and their activation functions. The functioning of an open-loop NARX network used as estimator is depicted in Fig. 3.6.

It is worth noting that when only input measurements are available, a NARX network can become a recurrent network by closing the loop feeding back the network outputs to the inputs, as shown in Fig. 3.7.

The parameters on which the designer can act concerns the overall architecture (number of neurons, connections between layers), while the value of the weights inside each neuron are derived from the network training.

3.2.3 Training the Network

A neural network is a learning system requiring an initial training procedure that adjusts the weights to improve the network performance. When the network task is the estimation of a nonlinear function, the training is performed by presenting to the network a set of examples of proper behavior, consisting in the inputs and the desired outputs (targets) for the relative inputs. Training can be implemented in two different ways:

- **Incremental mode**: each couple input-target generates an updating of the network weights;

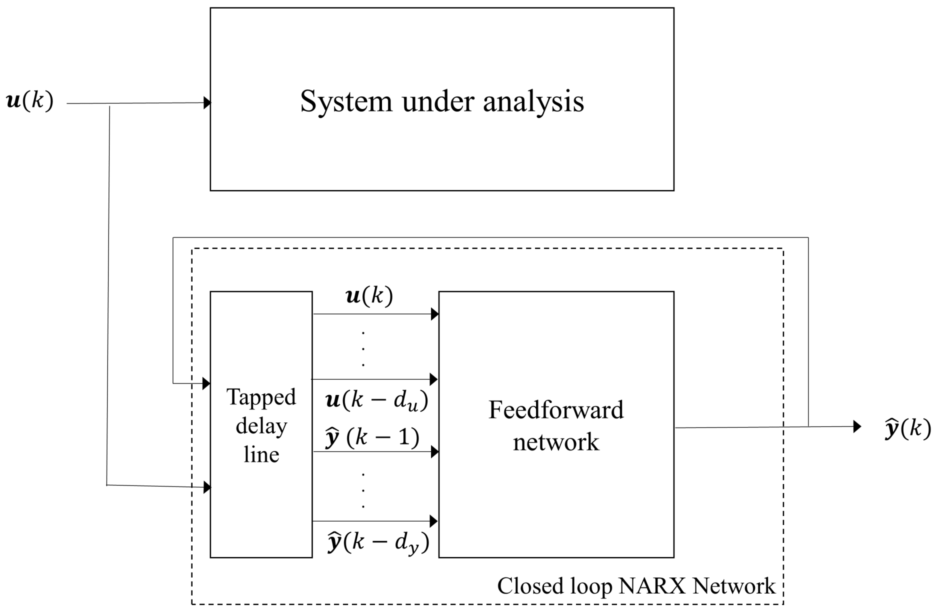

Figure 3.7: The closed loop NARX network, that reconstructs the system output on the basis of the delayed measurements and estimation.

- **Batch mode**: all inputs and targets are applied to the network before the weights are updated.

  Although this kind of training requires more memory storage capability, with respect to the incremental mode, it is characterized by a faster convergence and produces smaller errors, thus it will be considered in the following.

The training objective is the minimization of a performance function E, which depends on the weight vector $\mathbf{w}$.

Generally speaking, considering a number P of available example patterns consisting in the input-target pairs $(\mathbf{u}_p, \mathbf{t}_p)$, with $p = 1, ..., P$, defining $\hat{\mathbf{y}}_p$ the output generated by the network fed by $\mathbf{u}_p$, the p-th error vector can be expressed as:

$$\mathbf{e}_p = [\mathbf{t}_p - \hat{\mathbf{y}}_p] = [e_{p,1}, ..., e_{p,M}]^T \tag{3.23}$$

with $p = 1, ..., P$ and M number of outputs.

Furthermore, the global error vector $\bar{\mathbf{e}}$ collects each $\mathbf{e}_p$:

$$\bar{\mathbf{e}} = [e_{1,1}, ..., e_{1,M}, ..., e_{P,1}, ..., e_{P,M}]^T \tag{3.24}$$

Consequently, the performance function becomes:

$$E(\mathbf{w}) = \frac{1}{P}\sum_{p=1}^{P}(\mathbf{t}_i - \hat{\mathbf{y}}_i)^2 = \frac{1}{P}\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2 \tag{3.25}$$

Where the dependence of E by the N parameters grouped in the vector $\mathbf{w} = [w_1, ..., w_N]^T$ is implicit in the generated output $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\mathbf{w})$.

Any standard numerical optimization algorithm can be used to update the parameters, in order to minimize E. Among these, the most commons are iterative, and make use of characteristic matrices, such as the gradient $\mathbf{g}$ (or the Hessian $\mathbf{H}$) of the performance function, or the Jacobian $\mathbf{J}$ of the estimation error, defined as:

$$\mathbf{g} = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \left[\frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_N}\right]^T \tag{3.26}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \cdots & \frac{\partial^2 E}{\partial w_1 w_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_N w_1} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \tag{3.27}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial^{e}_{1,1}}{\partial w_1} & \cdots & \frac{\partial^{e}_{1,1}}{\partial w_N} \\ \frac{\partial^{e}_{1,2}}{\partial w_1} & \cdots & \frac{\partial^{e}_{1,2}}{\partial w_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^{e}_{P,M}}{\partial w_1} & \cdots & \frac{\partial^{e}_{P,M}}{\partial w_N} \end{bmatrix} \tag{3.28}$$

The successive iterations of these algorithms consist in the updating of the parameters and the calculation of the new value of the performance function, until a stop criterion is met. The updating rules of the most common optimization algorithms (i.e. the gradient descent, the Newton, the Gauss-Newton and the Levenberg-Marquardt algorithm) are reported in Table 3.3.

Table 3.3: The parameter updating rules of the most common optimization algorithm, aimed at the minimization of the performance function E. k is the iteration index, α is the learning rate and μ the combination coefficient.

Algorithm	Updating rule
Gradient Descent	$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \mathbf{g}_k$
Newton	$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$
Gauss-Newton	$\mathbf{w}_{k+1} = \mathbf{w}_k - \left(\mathbf{J}_k^T\mathbf{J}_k\right)^{-1}\mathbf{J}_k\bar{\mathbf{e}}_k$
Levenberg-Marquardt	$\mathbf{w}_{k+1} = \mathbf{w}_k - \left(\mathbf{J}_k^T\mathbf{J}_k + \mu\mathbf{I}\right)^{-1}\mathbf{J}_k\bar{\mathbf{e}}_k$

It can be demonstrated that the gradient descent algorithm, for a sufficiently small learning rate α value, is asymptotically convergent: around the solution $\mathbf{g}$, the gradient is close to zero and the weights do not meaningfully change. Otherwise, the Newton and the Gauss-Newton algorithms provide a faster convergence, but they both involve the computation of the inverse of a matrix which may not be invertible, causing instability in the procedure. Moreover, the Hessian matrix entails a burdensome computational effort, as it contains the second order derivative terms.

The *Levenberg-Marquardt* algorithm, originally proposed in (Marquardt, 1963), introduces an approximation of the Hessian matrix as $\mathbf{H} \approx \mathbf{J}^T\mathbf{J}+\mu\mathbf{I}$, where the first term of the sum is the Jacobian approximation (also exploited in Gauss-Newton) and the second term, driven by the combination coefficient $\mu > 0$, ensures the invertibility of the resulting matrix.

Therefore, the Levenberg-Marquardt algorithm provides both a fast and a stable convergence and it represents a suitable tool to train a neural network. Indeed, as explained in the next chapter, the neural network fault estimator blocks have been trained exploiting this method.

The training of a neural network based on the Levenberg-Marquardt algorithm, as explained in (Hagan and Menhaj, 1994), uses a technique denominated *back-propagation* training, in order to compute the Jacobian matrix for the updating rule.

Its name refers to the backward processing that starts from the output layer of the network towards the first layer, after a previous forward computation of neuron outputs.

Indeed, considering a multilayer network, the j-th neuron is fed by n_j inputs $u_{j,i}$ with $i = 1, ..., n_j$ and produces the output y_j by means of the activation function f and the neuron weights $w_{j,i}$ associated to the inputs:

$$y_j = f(\text{net}_j) \tag{3.29}$$

$$\text{net}_j = \sum_{i=0}^{n_j} w_{j,i} u_{j,i} \tag{3.30}$$

Then, the complex relations existing between a given output y_j of a neuron and a given output of the network $\hat{y}_m$ is expressed by means of an unknown nonlinear function F:

$$\hat{y}_m = F_{m,j}(y_j) \tag{3.31}$$

Where the dependence of F by other neurons outputs are implicit. The point of view of the single j-th neuron shown in Fig. 3.8, where the remaining part of the network can be seen as a black box represented by the function F.

The construction of the Jacobian matrix of Eq. 3.28 relies on the computation of terms in form of:

$$\frac{\partial e_{p,m}}{\partial w_{j,i}} = \frac{\partial (t_{p,m} - \hat{y}_{p,m})}{\partial w_{j,i}} = -\frac{\partial \hat{y}_{p,m}}{\partial w_{j,i}} \tag{3.32}$$

As the targets do not depend on the neuron weights.

Three useful derivative terms are derived in the following, as they act an important role in the Jacobian computation:

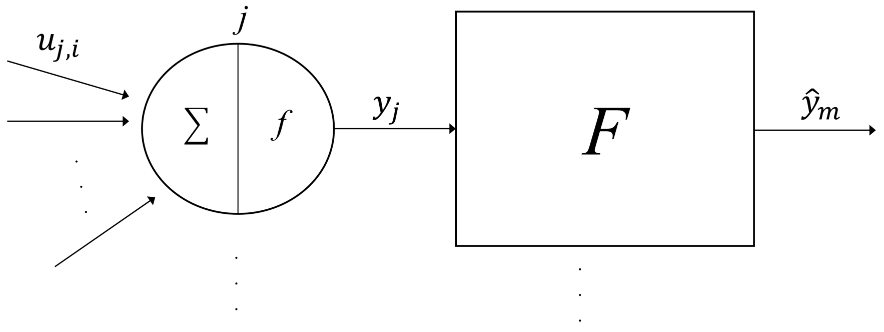

Figure 3.8: The point of view of a single neuron of the network: the elaboration of its input $u_{j,i}$ results in the neuron output y_j, then the remaining part of the network can be seen as a nonlinear function that maps y_j into a network output $\hat{y}_m$.

$$\frac{\partial \text{net}_j}{\partial w_{j,i}} = u_{j,i} \tag{3.33}$$

$$\frac{\partial y_j}{\partial \text{net}_j} = \frac{\partial f(\text{net}_j)}{\partial \text{net}_j} = f'_j \tag{3.34}$$

$$\frac{\partial \hat{y}_m}{\partial y_j} = \frac{\partial F_{m,j}(y_j)}{\partial \text{net}_j} = F'_{m,j} \tag{3.35}$$

Hence, Eq. 3.32 can be rewritten as the product of these terms:

$$\frac{\partial e_{p,m}}{\partial w_{j,i}} = -\frac{\partial \hat{y}_{p,m}}{\partial w_{j,i}} = -\frac{\partial \hat{y}_m}{\partial y_j}\frac{\partial y_j}{\partial \text{net}_j}\frac{\partial \text{net}_j}{\partial w_{j,i}} = -F'_{m,j} f'_j u_{j,i} \tag{3.36}$$

It is worth noting that each $u_{j,i}$ has been computed during the forward phase, as well as the derivative of the activation function f'_j, whilst the term $F'_{m,j}$ is computed recursively, starting from the output layer where its value is initialized to:

$$F'_{m,j} = \begin{cases} 1 & \text{if } m = j \\ 0 & \text{otherwise} \end{cases} \tag{3.37}$$

Once the Jacobian matrix has been computed, the iterations of Levenberg-Marquardt can start.

It is worth observing that a proper modification to the algorithm can be applied, when it is exploited to train a network. Indeed, many common implementations of the algorithm decrement the combination coefficient by a μ_{dec} factor after each iteration that provides a smaller value of the performance function. Otherwise, the weights are restored to the previous value and the combination coefficient is incremented by a μ_{inc} factor. The algorithm stops when the performance function oversteps a prefixed threshold. Finally, the overall training procedure is summarized in the block diagram of Fig. 3.9

3.2.4 Other Training Algorithms

In the related literature, many other training algorithms have been proposed, in (Hagan et al., 1996) a detailed overview of the most common training procedures is reported.

Although the backpropagation is an endorsed technique that is currently adopted in most of the training strategies, the rule according to which weights are updated may vary on the basis of the requirements of the application.

Table 3.4 reports some further examples of updating rule, the symbol Δ means the difference between two consecutive iterations, while α,Z,A,B,m are design parameters. With reference to the resilient algorithm the multiplication is understood as element-wise.

Table 3.4: Several optimization algorithms commonly used for training a network

Algorithm	$\Delta w_k = w_{k+1} - w_k$
Conjugate gradient with Powel restart	$-g_k + Z \cdot \Delta w_{k-1}$
Conjugate gradient with Fletcher updates	$-g_k + \frac{\\|g_k\\|^2}{\\|g_{k-1}\\|^2} \cdot \Delta w_{k-1}$
Conjugate gradient with Polak updates	$-g_k + \frac{\Delta g_{k-1}^T g_k}{\\|g_{k-1}\\|^2} \cdot \Delta w_{k-1}$
one-step secant	$-g_k + A\Delta w_{k-1} + B\Delta g_{k-1}$
Resilient	$A\Delta g \cdot \text{sgn}(g_{k-1})$
Gradient descent with momentum	$m\Delta w_{k-1} + \alpha(1-m)g_k$
Gradient descent with adaptive momentum	$m\Delta w_{k-1} + \alpha m g_k$

3.2.5 Problems Related to Neural Networks

The generalization skill of a neural network, understood as the capability to provide an approximately correct answer when presented with inputs that has not been used for training, strongly depends on the example patterns exploited for the training. Normally, a new input leads to an accurate output if it is close to an input already processed during the training, therefore the training data sets should be representative of all the operating conditions.

However, another problem may arise, although the training patterns cover the whole range of cases that may occur. It is called *overfitting* and involves a poor generalization capability due to memorization of the training examples.

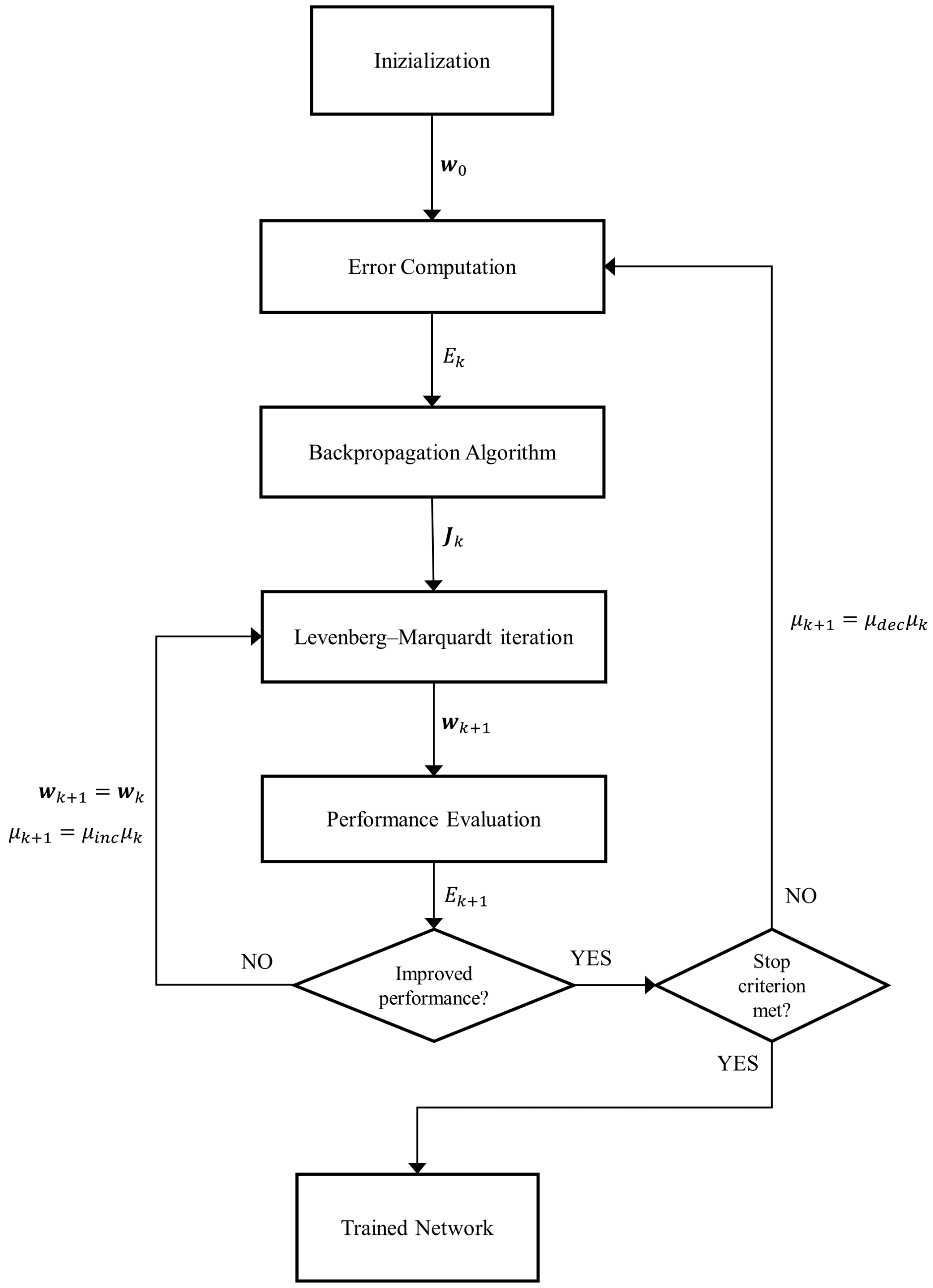

Figure 3.9: The block diagram of the training procedure, exploiting the backpropagation algorithm in combination with the Levenberg-Marquardt iterations.

Overfitting is often caused by an oversized network. Thus, in order to improve the network generalization, an appropriate number of neurons and layers should be adopted. Small networks do not have the power to overfit data, but they cannot realize complex nonlinear functions. The technique called *early stopping* prevents overfitting during the training processing. A validation data set is removed from the training example patterns and used to monitor the generalization performances. When, during the training, the error on validation data set start increasing, the training is stopped, as the excessive number of patterns presented results in the network overfitting.

Chapter 4

Fault Diagnosis and Fault Tolerant Control Design

This chapter introduces the Fault Diagnosis and the Fault Tolerant schemes adopted in the simulations of Chapter 5. Firstly, the Fault Mode and Effect Analysis is described, as it leads to an effective design of the fault estimators. Then, the Fault Diagnosis scheme is presented highlighting its features of the Fault Detection and Isolation. Finally, the implementation of the Fault Tolerant Controller is shown. It relies on the on-line accommodation of the system faults, modeled as either actuator (input) faults or sensor (output) faults.

4.1 Fault Mode and Effect Analysis

Following the guidelines reported in (Stamatis, 2003), a Failure Mode and Effect Analysis (FMEA) has been performed on the wind turbine system, as well as on the wind farm. The FMEA is a sensitivity analysis aimed at estimating the most sensitive measurements with respect to the simulated fault conditions.

In practice, the monitored fault signals have been injected into the benchmark simulators, assuming that only a single fault may occur in the considered plant. Then, the relative mean square errors (RMSE) between the fault-free and faulty measured signals are computed, so that, for each fault, the most sensitive signal can be selected. The results of the FMEA are shown in Table 4.1 for the wind turbine.

Table 4.1: The most sensitive measurements with respect to the faults, for the wind turbine benchmark

Fault	1	2	3	4	5	6	7	8	9
Measurement	$\beta_{1,m1}$	$\beta_{2,m2}$	$\beta_{3,m1}$	$\omega_{r,m1}$	$\omega_{r,m1}$	$\beta_{2,m1}$	$\beta_{3,m2}$	$\tau_{g,m}$	$\omega_{g,m1}$
RMSE	11,29	0,98	2,48	1,44	1,45	0,80	0,73	0,84	0,77

Afterwards the FMEA can be conducted on the basis of a selection algorithm that is achieved by introducing the normalized sensitivity function N_x, defined as in the follow:

$$N_x = \frac{S_x}{S_x^*}$$

Where:

$$S_x = \frac{\left\|x_f[k] - x_n[k]\right\|_2}{\left\|x_n[k]\right\|_2} \tag{4.1}$$

$$S_x^* = \max \frac{\left\|x_f[k] - x_n[k]\right\|_2}{\left\|x_n[k]\right\|_2} \tag{4.2}$$

Its value represents the effect of the considered fault case with respect to a certain measure signal $x[k]$. The subscripts f and n indicate the faulty and the fault-free case, respectively. Therefore the measurements most affected by the considered fault imply a value of N_x equal to 1. Otherwise, a small value of N_x, i.e. close to zero, denotes a signal x not affected by the fault. The signals characterized by high value of N_x can be selected as the most sensitive measurements and they will be considered in the design of the FDI blocks. The results of the FMEA sensitivity are reported in Table 4.2 and in Table 4.3, for the wind turbine and for the wind farm, respectively.

Table 4.2: The selected signals for each fault included in the wind turbine benchmark, divided by inputs and outputs

Fault	Most Sensitive Inputs	Most Sensitive Outputs
1	$\beta_{1,m1}$, $\beta_{1,m2}$	$\omega_{g,m2}$
2	$\beta_{1,m2}$, $\beta_{2,m2}$	$\omega_{g,m2}$
3	$\beta_{1,m2}$, $\beta_{3,m1}$	$\omega_{g,m2}$
4	$\beta_{1,m2}$	$\omega_{g,m2}$, $\omega_{r,m1}$
5	$\beta_{1,m2}$	$\omega_{g,m2}$, $\omega_{r,m2}$
6	$\beta_{1,m2}$, $\beta_{2,m1}$	$\omega_{g,m2}$
7	$\beta_{1,m2}$, $\beta_{3,m2}$	$\omega_{g,m2}$
8	$\beta_{1,m2}$, $\tau_{g,m}$	$\omega_{g,m2}$
9	$\beta_{1,m2}$	$\omega_{g,m1}$, $\omega_{g,m2}$

Table 4.3: The selected signals for each fault included in the wind farm benchmark

Fault	Most Sensitive Measurements
1	β_2, $P_{g,2}$, β_7, $P_{g,7}$, $v_{w,m}$
2	β_1, $\omega_{g,1}$, β_5, $\omega_{g,5}$, $v_{w,m}$
3	β_6, $P_{g,6}$, β_8, $\omega_{g,8}$, $v_{w,m}$

As a result, the fault diagnosis blocks that have to be designed, can implement the reduced fault models instead of the overall system model of Eq. 2.18 with a

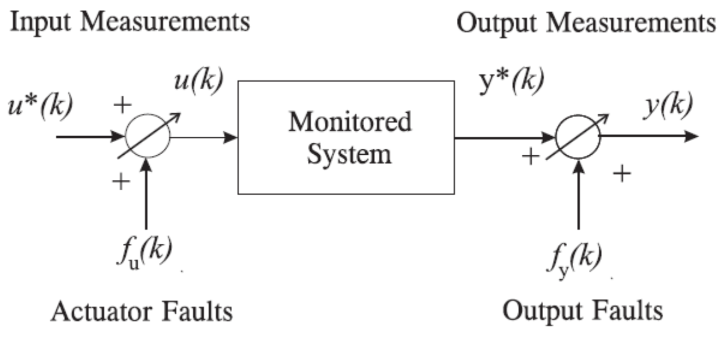

Figure 4.1: The faults affecting the system under analysis, i.e. the wind turbine or the wind farm, as additive signals on the input (actuator) and output measurements.

noteworthy simplification of the inner structure, thus providing a decrease in the computational effort.

4.2 Fault Diagnosis

In the following the discrete-time monitored systems, i.e. the wind turbine and the wind farm, is assumed to be affected by additive faults on the input (actuator) and output (sensor) measurements, as represented in Fig. 4.1, in forms of:

$$\mathbf{u}(k) = \mathbf{u}^*(k) + \mathbf{f}_u(k) \tag{4.3}$$
$$\mathbf{y}(k) = \mathbf{y}^*(k) + \mathbf{f}_y(k) \tag{4.4}$$

Where $\mathbf{u}^*(k), \mathbf{y}^*(k)$ are the actual unmeasurable variables, $\mathbf{u}(k), \mathbf{y}(k)$ represent the sensor acquisitions, affected by both the measurement noise and the faults. $\mathbf{f}_u(k), \mathbf{f}_y(k)$ are additive signals, that assume values different from zero only in presence of faults.

Among the different approaches to generate the residual signals, recalled in Chapter 1, the solution adopted in this work exploits fuzzy and neural network models, which provide an on-line estimation of the faulty signals. Hence, as shown in Fig. 4.2 residuals $\mathbf{r}$ are generated by means of the direct comparison of the measured $\mathbf{y}(k)$ and the estimated outputs $\hat{\mathbf{y}}$:

$$\mathbf{r}(k) = \hat{\mathbf{y}}(k) - \mathbf{y}(k) \tag{4.5}$$

The fault diagnosis process involves, as first step, the fault detection task. It is performed here by using a proper thresholding logic operating on the residuals after their elaboration into a proper evaluation function:

$$\mathbf{r}_e(k) = f(\mathbf{r}(k)) \tag{4.6}$$

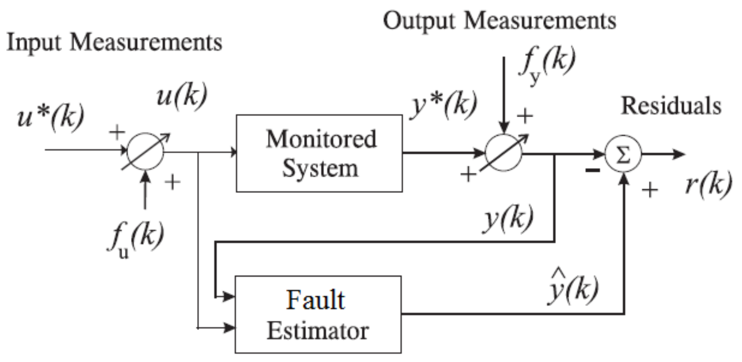

Figure 4.2: The residual generation achieved by the difference between the acquired measurements and the estimated outputs.

Where the proposed function f can be the identity function, in case of fuzzy estimator, or the moving average or variance in case of neural networks, as explained in Chapter 5. Then, the occurrence of the i-th fault can be detected according to:

$$\begin{cases} \bar{r}_{e_i} - \delta\sigma_{r_i} \leq r_{e_i} \leq \bar{r}_{e_i} + \delta\sigma_{r_i} & \text{fault-free} \\ r_{e_i} < \bar{r}_{e_i} - \delta\sigma_{r_i} \text{ or } r_{e_i} > \bar{r}_{e_i} + \delta\sigma_{r_i} & \text{faulty} \end{cases} \tag{4.7}$$

Where the i-th item r_{e_i} of the residual vector $\mathbf{r}_e$ is considered a random variable, whose unknown mean $\bar{r}_{e_i}$ and variance $\sigma^2_{r_i}$ can be estimated in fault-free condition, after the acquisition of N samples, as follows:

$$\bar{r}_{e_i} = \frac{1}{N}\sum_{k=1}^{N} r_{e_i}(k) \tag{4.8}$$

$$\sigma^2_{r_i} = \frac{1}{N}\sum_{k=1}^{N}(r_{e_i}(k) - \bar{r}_{e_i})^2 \tag{4.9}$$

The tolerance parameter $\delta \geq 2$ has to be properly tuned in order to separate the fault-free from the faulty condition. The δ value determines the trade-off between the false alarm rate and the fault detection probability. A common choice of δ relies on the three-sigma rule, otherwise extensive simulations can be performed to optimize the δ value.

Consequently to the fault detection, the fault isolation task is achieved by means of two observer schemes. Faults are here subdivided into two main groups: the faults concerning the inputs $\mathbf{f}_u$ and the faults concerning the output $\mathbf{f}_y$. Following the generalized observer scheme of Fig. 4.3, in order to uniquely isolate one of the input faults, under the assumption that no output faults occur in the meanwhile, a bank of MISO estimators is used, whose number is equal to the number of input faults to be isolate. Then, the i-th fault estimator is driven by all but the i-th input, so that, when the i-th fault occurs, its residual signal is the only one that does not detect the fault. In particular, when the i-th fuzzy or neural

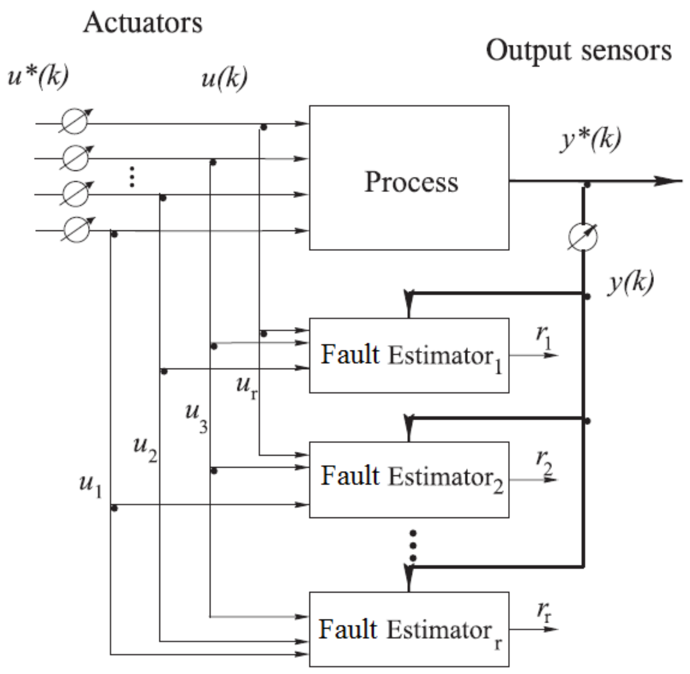

Figure 4.3: The general observer scheme: the fault estimators are driven by all but one input, so that the relative residual is insensitive only to the fault affecting that input.

network estimator, insensitive to the i-th fault has to be designed, all but the i-th input, in addition to the output, are considered in the regressor vector, for the fuzzy models, and in the training of the neural networks.

It is worth noting that multiple faults occurring at the same time cannot be correctly isolated, using this configuration.

On the other hand, in order to isolate one or multiple output faults, under the assumption of no input faults occurring, another bank of estimator is used. In this case, the i-th estimator provides directly the i-th residual as it is driven by all the system inputs and only the i-th output. This configuration is better known as dedicated observer scheme.

The isolation capabilities of the adopted observer banks can be summarized by means of the so-called *fault signature*, depicted in Table 4.4, where each entry that is characterized by a value equal to 1 means that the considered residual is sensitive to the fault (zero otherwise), under the hypothesis above mentioned.

The FMEA, that has to be executed before the design of the observers, suggests how to select the inputs-output configuration for the fault estimator blocks. Then, the design of the fuzzy or neural networks model can be performed. Finally, the threshold test logic of Eq. 4.7 allows the achievement of the fault diagnosis task. A summary of the complete design flow is shown in Fig. 4.5.

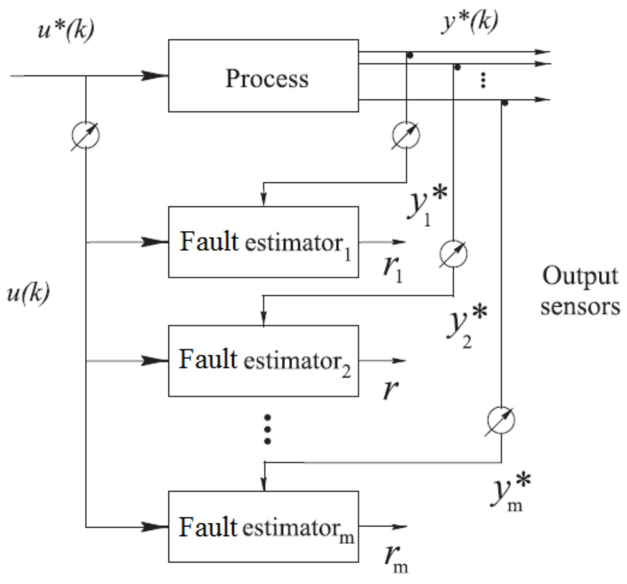

Figure 4.4: The dedicated observer scheme: all the fault estimators are driven by all inputs and each of them generates the residual relative to only one output faulty signal.

Table 4.4: The fault signatures for input and output observer schemes

	u_1	u_2	...	u_r	y_1	y_2	...	y_r
r_{u_1}	0	1	...	1	0	0	...	0
r_{u_2}	1	0	...	1	0	0	...	0
$\vdots$			$\ddots$				...	$\vdots$
r_{u_r}	1	1	...	0	0	0	...	0
r_{y_1}	0	0	...	0	1	0	...	0
r_{y_2}	0	0	...	0	0	1	...	0
$\vdots$			...				$\ddots$	$\vdots$
r_{y_m}	0	0	...	0	0	0	...	1

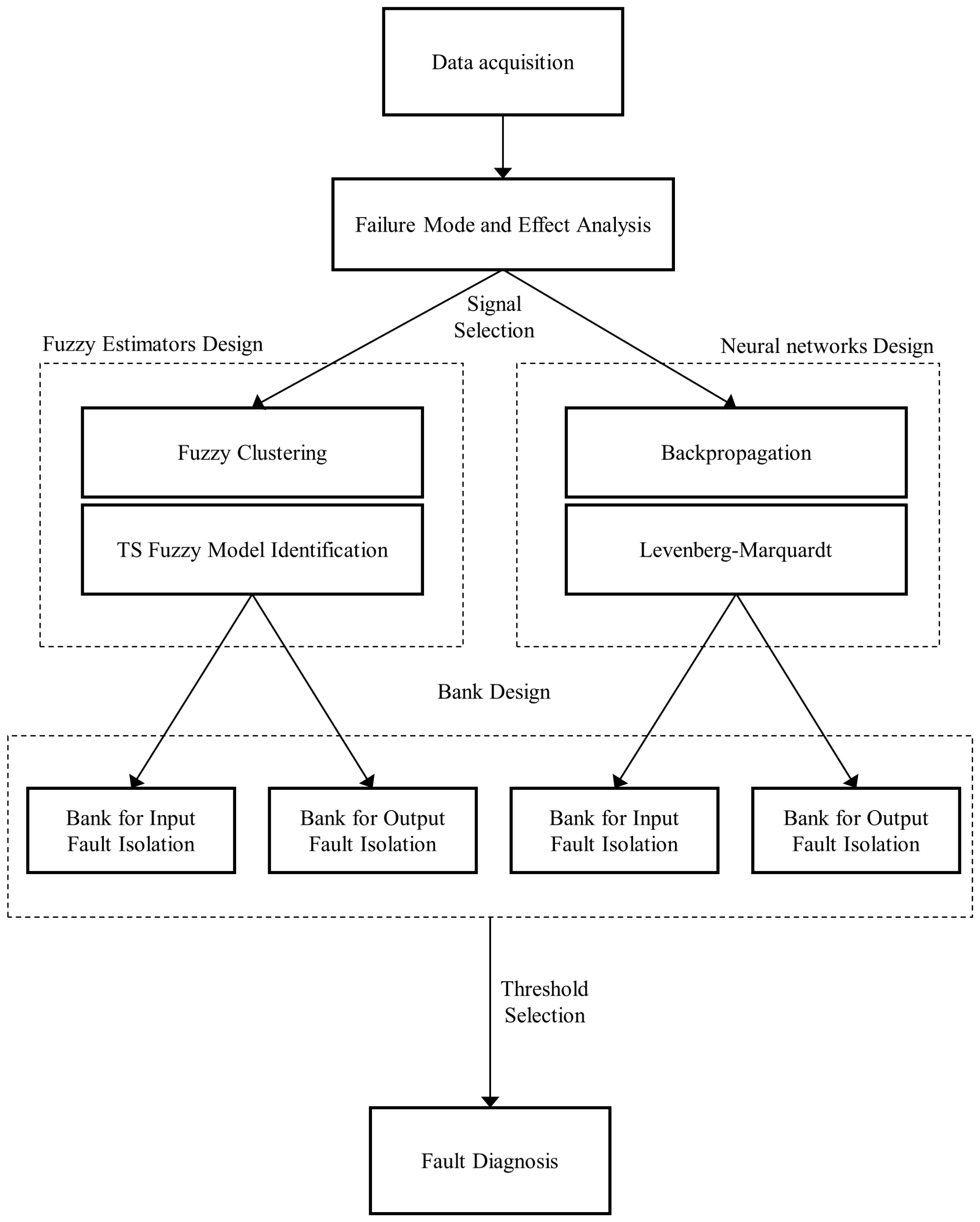
Data acquisition
Failure Mode and Effect Analysis
Signal Selection
Fuzzy Estimators Design
Neural networks Design
Fuzzy Clustering
TS Fuzzy Model Identification
Backpropagation
Levenberg-Marquardt
Bank Design
Bank for Input Fault Isolation
Bank for Output Fault Isolation
Bank for Input Fault Isolation
Bank for Output Fault Isolation
Threshold Selection
Fault Diagnosis

Figure 4.5: The block diagram of the overall FDI design procedure.

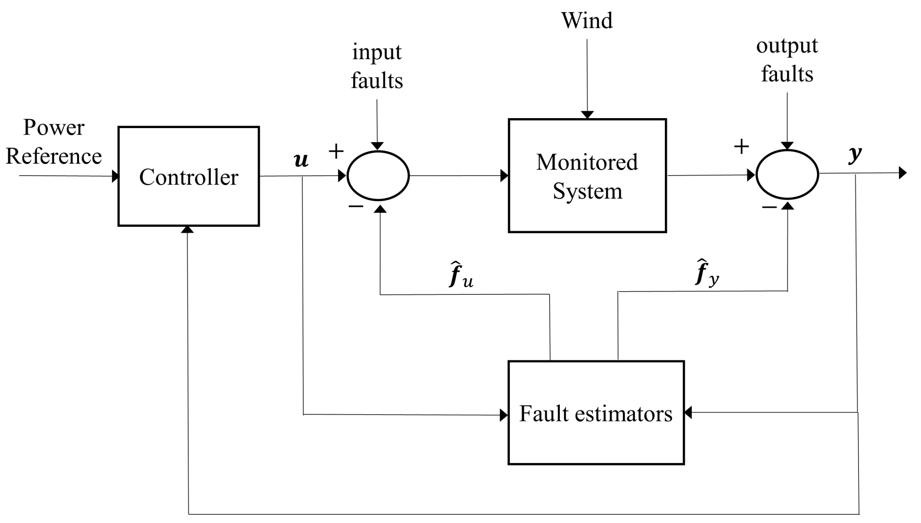

Figure 4.6: The FTC strategy adopted in this work: the FDI block provide the on-line fault estimations, that are used to compensate the faulty input-output signals, so that the controller can force the system to track the desired reference.

4.3 Fault Tolerant Control

The structure of the FTC system proposed in this work, and applied to both the wind turbine and the wind farm benchmarks described in Chapter 2, is mainly based on the Fault Diagnosis module that provides the on-line fault estimation by using fuzzy or neural network models. The result of a proper fault identification permits the compensation of the faulty measurement signals, before their access to the controller, so that the proper reference signal can be send to the turbine system, without the modification of the pre-existent controller. Fig. 4.6 shows the overall FTC strategy.

Therefore, the fault estimations $\hat{\mathbf{f}}_u$,$\hat{\mathbf{f}}_y$ are exploited for the compensation of both the input and the output measurements used by the system controller. In particular, the actuator signal coming from the controller is compensated by $\hat{\mathbf{f}}_u$, while $\hat{\mathbf{f}}_y$ corrects the output measurement acquired from the monitored system. After the fault accommodation the controller can track the nominal power reference signals. It is worth noting that, thanks to this fault estimation feedback, the controller could be easily designed considering the fault-free system condition.

Further investigations regarding the stability analysis of the overall FTC module are highlighted in Chapter 5, where it is shown that the variables of the models remain bounded in a set, which assure control performance, even in presence of faults. Moreover these faults do not modify the system structure, hence the global stability is guaranteed. However, whilst the fault effect is eliminated in steady-state condition, during the transient the compensation can be not properly handled, and the stability properties should be considered.

Chapter 5

Simulations, Experiments and Results

This chapter shows the simulations related to the considered benchmark systems, in which the proposed solution for the fault diagnosis and the fault tolerant control have been implemented. Firstly, the focus is placed on the single wind turbine benchmark, both the fuzzy and the neural network fault estimators are analyzed and validated by means of a Monte Carlo analysis. Then, their performances are compared to those of other fault diagnosis methods, commonly adopted in the related literature. Furthermore, the tracking capability of the fault tolerant controller, based on the fault accommodation strategy, is evaluated.

Afterwards, the wind farm benchmark system is considered. Similarly to the analysis carried out for the single turbine system, the developed fault diagnosis and fault tolerant control modules are tested, validated and evaluated with respect to other commonly adopted solutions.

Finally, in order to assess the proposed systems in a more realistic framework, the Hardware in the Loop (HIL) test has been performed, by means of an industrial computer interacting with the on-board electronics.

5.1 Wind Turbine Simulations

In the following, with reference to the wind turbine benchmark model of Section 2.2, all the simulations are driven by the same wind mean speed sequence (see Eq. 2.6), reported in Fig. 5.1. It comes from real acquisition of wind speed data from a wind farm. It represents a good coverage of typical operating condition, as it ranges from 5 to 20 m/s, with a few spikes at 25 m/s. The other wind speed components are represented by uniform random variables.

The simulations last for 4400 s, during which only one fault may occur. The discrete-time benchmark model runs at a sampling frequency of 100 Hz, so that $N = 440000$ samples per simulation are acquired. With reference to the different scenarios described in Sec. 2.2.4, Table 5.1 reports the shape and the time of the fault signals affecting the system. They are modeled as input (actuator) or output

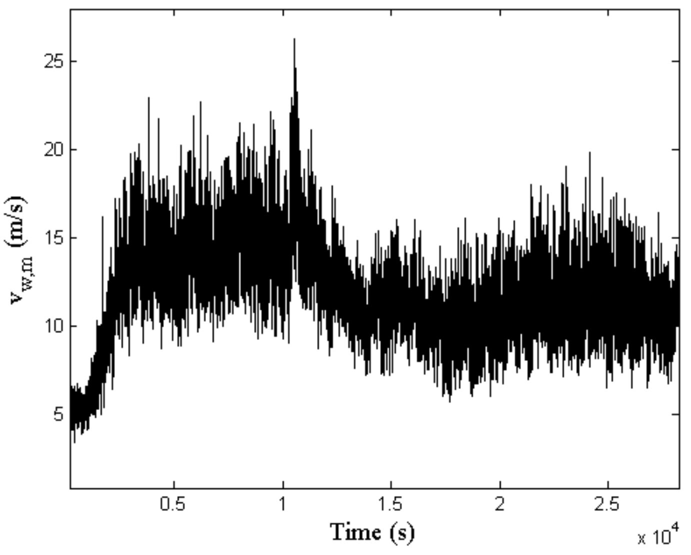

Figure 5.1: The wind speed sequence driving the simulations.

(sensor) additive fault, based on the FMEA results of Section 4.1.

Table 5.1: The characteristics of the faults affected the system during the simulation

Fault	Type	Shape	Time (s)
1	actuator	step	2000 – 2100
2	actuator	step	2300 – 2400
3	actuator	step	2600 – 2700
4	actuator	step	1500 – 1600
5	actuator	step	1000 – 1100
6	sensor	step	2900 – 3000
7	sensor	trapezoidal	3500 – 3600
8	sensor	step	3800 – 3900
9	sensor	step	4100 – 4300

In order to highlight how faults affect the system, the comparison between the faulty and the fault free signal is represented in Fig. 5.2, regarding the most affected signals of the FMEA test.

5.1.1 Fault Diagnosis via Fuzzy Identified Models

The issue on the Fault Diagnosis of the wind turbine benchmark model via fuzzy models is discussed in (Simani et al., 2015a) and (Simani et al., 2015c). As addressed in Section 3.1, the fuzzy c-means clustering exploits a number $n_C = 4$ of clusters and $o = 3$ delay on input and output regressors. The algorithm generates the membership function points, that are fitted through Gaussian membership

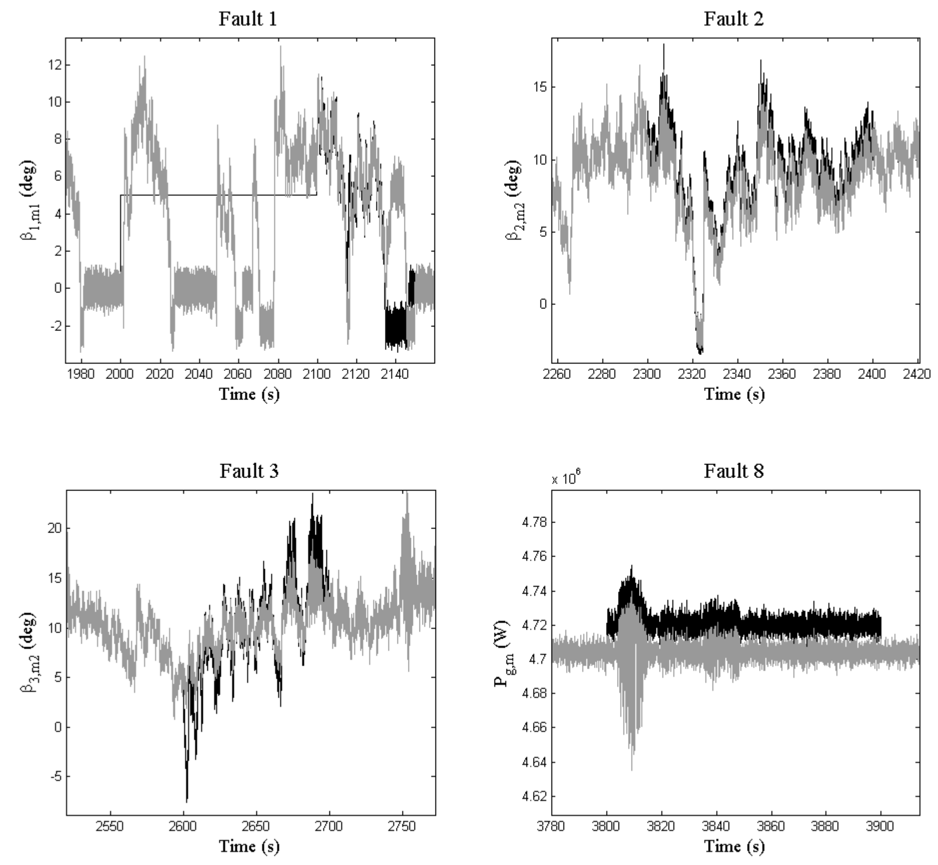

Figure 5.2: The faulty signals (black line) compared with the fault-free signals (grey line), in case of fault 1,2,3, and 8.

functions.

Afterwards, the regressand α and δ of Eq.3.9 is identified for each cluster, following the procedure explained in Section 3.1. As a result, the TS models can be implemented and the nine fault estimators are built and organized into the two observer schemes of Section 4.2, in order to accomplish the fault diagnosis and identification task.

The modeling capabilities of the fuzzy TS models are evaluated in terms of Root Mean Squared Error (RMSE), where the error is calculated as the difference between the measured and the estimated signals, for each output provided by the fuzzy estimators, in nominal fault-free condition. Table 5.2 shows the achieved modeling performances of the nine designed fault estimators.

Table 5.2: The capability of the estimators to reconstruct the system output in fault-free-condition, in terms of RMSE.

Fault Estimator	1	2	3	4	5	6	7	8	9
RMSE	0.016	0.023	0.021	0.020	0.019	0.021	0.017	0.021	0.019

Furthermore, these error signals are directly exploited as residual and they are compared with the thresholds of Eq. 4.7, optimally selected in order to achieve the optimization of the fault diagnosis performance indices, e.g. the missed fault and the false alarm rate, defined in the following. Table 5.3 reports the adopted δ value for the threshold logic of each fault estimator.

Table 5.3: The design parameter δ for each residual generator, that determines the threshold logic.

Residual	1	2	3	4	5	6	7	8	9
δ	3.8	4.3	4.2	4.5	3.7	4.4	4.3	3.5	3.9

The meaningful simulation results, proposed in the following, consider two actuator faults f_u and two sensor fault f_y, namely faults 1,4 and faults 8,9 of the scenarios described in Section 2.2.4.

These faults change the monitored input and output signal $\mathbf{u}$, $\mathbf{y}$ affecting the residual r_1,r_4 and r_8,r_9 generated by the fuzzy fault estimators. These residual are depicted in Fig. 5.3, that clearly show the achievement of the fault detection task, as they are significantly above the threshold bounds only when the relative fault is active.

5.1.2 Fault Diagnosis via Neural Networks

Nine open-loop NARX neural network (see Sec. 3.2) have been designed to estimate the nonlinear behavior between the acquired measurements and the proposed faults. The selected architecture of the networks involves two layers, namely the

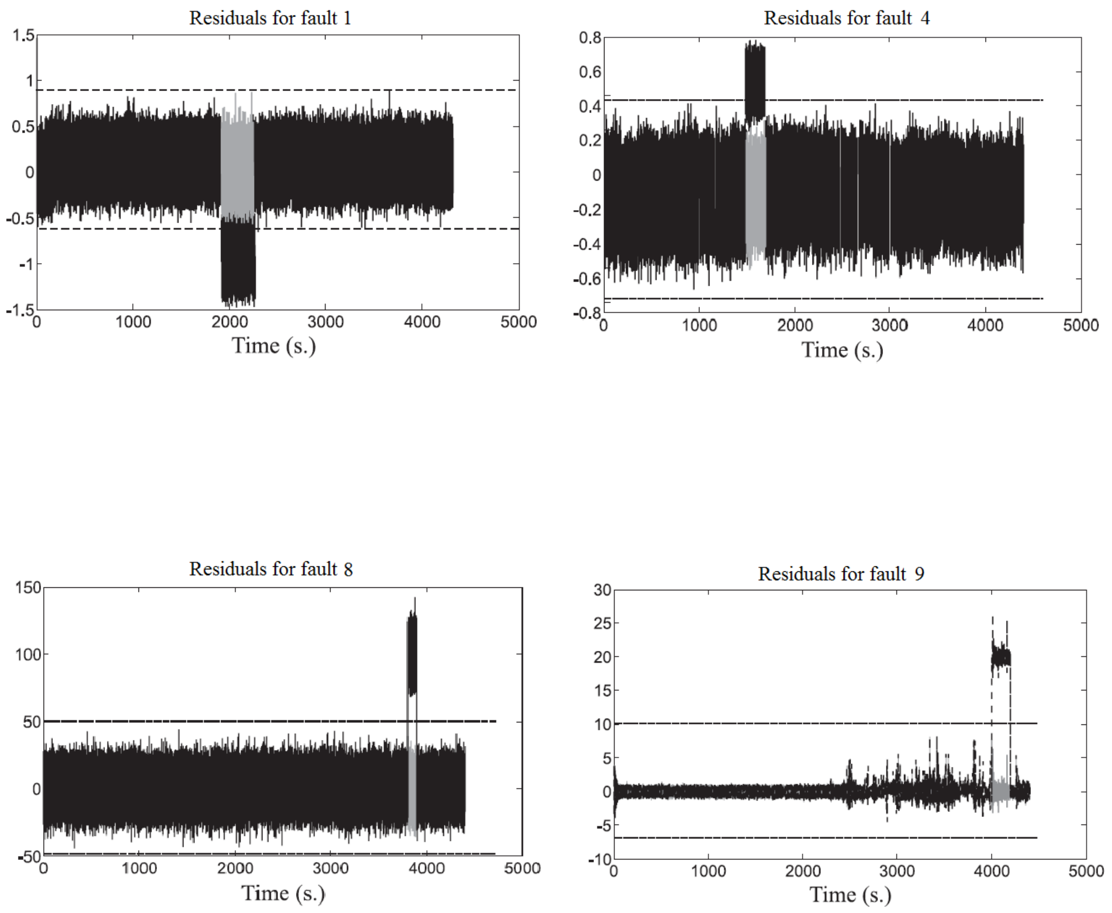

Figure 5.3: The residuals generated in faulty conditions by fuzzy estimators (black continuous line) compared the fault-free residuals (grey line) and the fixed thresholds (dotted line). The considered residuals regard fault 1,4,8 and 9

hidden layer and the output layer. The number of neurons in the hidden layer has been fixed to $n_h = 16$. Finally, a number of $d_u = d_y = 4$ has been chosen for the input-output delays. Similarly to the fuzzy models, the neural networks modeling capabilities have been tested in terms of RMSE and the results are reported in Table 5.4.

Table 5.4: The capability of the neural networks to reconstruct the system output in fault-free-condition, in terms of RMSE.

Fault Estimator	1	2	3	4	5	6	7	8	9
RMSE	0.009	0.009	0.009	0.012	0.011	0.011	0.009	0.009	0.014

The fault detection task is achieved by comparing the residual with a fixed optimized threshold, in this case, in contrast to the direct approach exploited by the fuzzy estimators, the residuals are filtered by an evaluation function ahead of the threshold comparison. This evaluation function can be either a mobile average (MA) or a mobile variance (MV), with a properly tuned window size, as reported in Table 5.5.

Table 5.5: The residual filter functions, for each fault estimator.

Residual	Evaluation function	Window samples
1	MV	20
2	MA	60
3	MV	20
4	MA/MV	45/55
5	MA	50
6	MV	60
7	MV	70
8	MA	50
9	MA	50

Figure 5.4 shows some meaningful residual signal for actuator faults, together with the relative thresholds, while Fig 5.5 shows the residual regarding the sensor faults. Further details on validation and comparative results are described in the following.

5.1.3 Validation and Comparative Analysis

The evaluation of the performances of the considered Fault Diagnosis strategies is based on the computation of the following indices:

- **False Alarm Rate** (FAR): the ratio between the number of wrongly detected faults and the number of simulated faults;

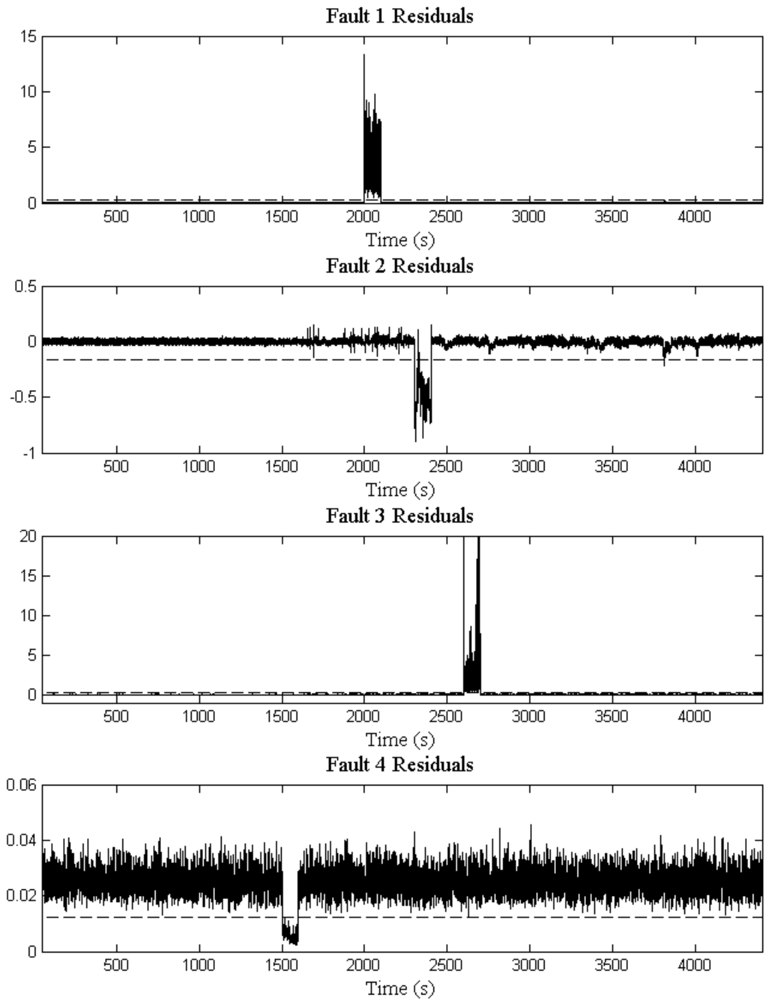

Figure 5.4: The residuals generated in faulty conditions by neural network estimators (continuous line) compared the fixed thresholds (dashed line). The considered residuals concerns the actuator faults 1-4.

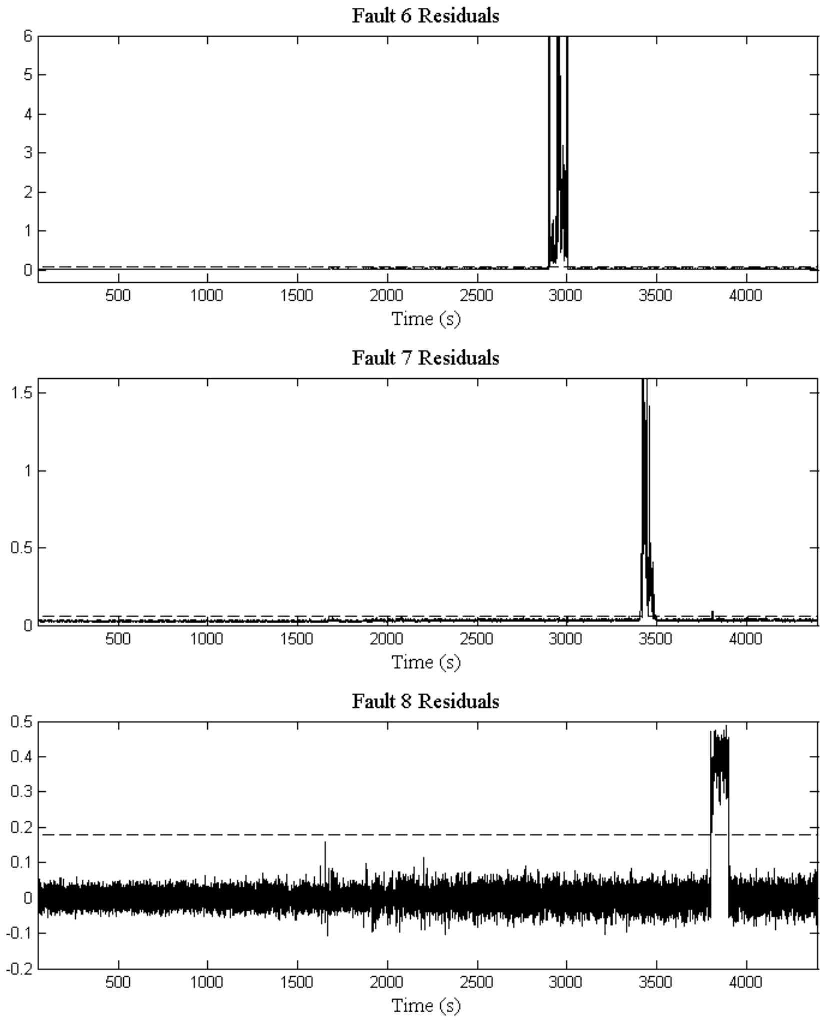

Figure 5.5: The residuals generated in faulty conditions by neural network estimators (continuous line) compared the fixed thresholds (dashed line). The considered residuals concerns the sensor faults 6-8.

- **Missed Fault Rate** (MFR): the ratio between the total number of missed faults and the number of simulated faults;
- **True FDI Rate** (TFR): the ratio between the number of correctly detected faults and the number of simulated faults (complementary to MFR);
- **Mean FDI Delay** (MFD): the delay time between the fault occurrence and the fault detection.

A proper Monte Carlo analysis has been performed in order to compute these indices and to test the robustness of the considered FDI schemes. Indeed, the Monte Carlo tool is useful at these stage, as the efficacy of the diagnosis depends on both the model approximation capabilities and the measurements errors.

In particular, a set of 1000 Monte Carlo runs has been executed, during which realistic wind turbine uncertainties have been considered by modeling some meaningful variables as Gaussian stochastic processes around the nominal values and with standard deviations corresponding to the realistic minimal and maximal error values of Table 5.6.

Table 5.6: The realistic variation of some model parameters, simulated during the Monte Carlo analysis.

Parameter	Nominal Value	Min. Error	Max. Error
ρ	1,225 Kg/m^3	$\pm 0.1\%$	$\pm 20\%$
J	7,794 $\times 10^6$ Kg/m^3	$\pm 0.1\%$	$\pm 30\%$
C_p	C_{p0}	$\pm 0.1\%$	$\pm 50\%$

In addition to the proposed fuzzy and neural network fault estimators, the performance indices of other fault diagnosis schemes are analyzed.

The first alternative approach considered here uses a Support Vector machine based on a Gaussian Kernel (GKSV) developed in (Laouti et al., 2011). The scheme defines a vector of features for each fault, which contains relevant signals obtained directly from measurements, filtered measurements or their combinations. These vectors are subsequently projected onto the kernel of the Support Vector Machine (SVM), which provides suitable residuals for all of the defined faults. Data with and without faults were used for learning the model for the FDI of the specific faults.

The second scheme consists in an Estimation-Based (EB) solution shown in (Zhang et al., 2011). In particular, a fault detection estimator is designed to detect a fault, and an additional bank of estimators is derived to isolate them. The method was designed on the basis of a system linear model and used fixed thresholds. Each estimator for fault isolation was computed on the basis of the particular fault scenario under consideration.

The third method relying on Up-Down Counters (UDC) was addressed in (Ozdemir et al., 2011). These tools, are commonly used in the aerospace framework, and they provide a different approach to the decision logic applied to the FDI residuals. Indeed, the decision to declare the fault occurrence involves discrete-time dynamics and is not simply a function of the current residual value.

The fourth approach Combines Observer and Kalman filter (COK) methods (Chen et al., 2011). It relies on an observer used as a residual generator for diagnosing the faults of the drive-train, in which the wind speed is considered a disturbance. This diagnosis observer was designed to decouple the disturbance and simultaneously achieve optimal residual generation in a statistical sense. For the other two subsystems of the wind turbine, a Kalman filter-based approach was applied. The residual evaluation task used a generalized likelihood ratio test, and cumulative variance indices were applied. For fault isolation purpose, a bank of residual generators was exploited. Sensor and system faults were thus isolated via a decision table.

Finally, the fifth method is a General Fault Model (GFM) scheme, which is a method of automatic design (Svärd et al., 2011). The FDI strategy consists of three main steps. In the first step, a large set of potential residual generators was designed. In the second step, the most suitable residual generators to be included in the final FDI system were selected. In the third step, tests for the selected set of residual generators were performed, which were based on comparisons of the estimated probability distributions of the residuals, evaluated with fault-free and faulty data.

The comparative analysis results are reported in Table 5.7.

The results show the efficacy of the proposed FDI solutions. In details, both fuzzy and neural network estimators seem to work better than other approaches, and they have a noteworthy performance level considering the mean delay time, which is significantly lower than 10 s for all the fault cases. Also false alarm and missed fault rate are often lower than those of other approaches, particularly neural networks features an almost null missed fault rate for all the considered faults. However, for both fuzzy and neural networks FDI design, optimization stages are required, for example for the selection of the optimal thresholds. Furthermore, GKSV involves delays bigger than 25 s, with false alarms and missed fault rate up to 35 %. EB has comparable performance with respect to GKSV in terms of false alarm, true detection and missed fault rate, but with a quicker detection. UDC often involves high false alarm rates, bigger than 12% for all the detectable faults. COK and GFM have similar performances, with delay time higher than 10 s, false alarm and missed fault bigger than 10 %. Fault 9 concerns the drive train. This fault is difficult to detect at wind turbine level, therefore it is investigated also in the context of the wind farm benchmark. However, the fuzzy estimators can detect it, with a minimum delay but with a lower True FDI Rate, with respect to the other fault cases.

5.1.4 Fault Tolerant Control

The fault diagnosis modules, working as nonlinear adaptive filters (Simani et al., 2014a), are exploited to create a further control loop, so that the accommodation of the diagnosed faults allows the system to work properly with the pre-existent

Table 5.7: The comparative analysis of different approaches to the fault diagnosis of the wind turbine benchmark model, among these the fuzzy and the neural network estimators.

Fault	Index	GKSV	EB	UDC	COK	GFM	Fuzzy	Neural
1	FAR	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	MFR	0.002	0.003	0.002	0.003	0.002	0.001	0.001
	TFR	0.978	0.977	0.987	0.977	0.982	0.999	0.999
	MFD (s)	0.03	0.03	0.04	10.32	0.05	0.02	0.01
2	FAR	0.234	0.224	0.123	0.003	0.235	0.001	0.228
	MFR	0.343	0.333	0.232	0.029	0.532	0.003	0.001
	TFR	0.657	0.667	0.768	0.971	0.468	0.997	0.999
	MFD (s)	47.24	44.65	69.03	19.32	13.74	0.08	0.08
3	FAR	0.004	0.141	0.123	0.056	0.135	0.003	0.001
	MFR	0.006	0.132	0.241	0.128	0.232	0.008	0.001
	TFR	0.974	0.868	0.769	0.872	0.768	0.992	0.999
	MFD (s)	0.05	0.54	0.05	19.32	0.74	0.02	0.01
4	FAR	0.006	0.005	0.123	0.056	0.236	0.004	0.001
	MFR	0.005	0.006	0.113	0.128	0.333	0.004	0.001
	TFR	0.975	0.994	0.887	0.872	0.667	0.996	0.999
	MFD (s)	0.15	0.33	0.04	19.32	17.64	0.02	0.69
5	FAR	0.178	0.004	0.234	0.256	0.236	0.002	-
	MFR	0.223	0.005	0.254	0.329	0.242	0.003	-
	TFR	0.777	0.995	0.746	0.671	0.758	0.997	-
	MFD (s)	25.95	0.07	0.04	31.32	9.49	0.03	-
6	FAR	0.897	0.173	0.334	0.156	0.096	0.042	0.001
	MFR	0.987	0.234	0.257	0.129	0.042	0.033	0.001
	TFR	0.013	0.766	0.743	0.871	0.958	0.967	0.999
	MFD (s)	95.95	11.37	12.94	34.02	9.49	3.03	0.01
7	FAR	0.899	0.044	0.134	0.134	0.123	0.047	0.676
	MFR	0.899	0.035	0.121	0.101	0.098	0.023	0.001
	TFR	0.101	0.965	0.879	0.899	0.902	0.977	0.999
	MFD (s)	99.95	26.17	13.93	35.01	29.79	5.07	6.87
8	FAR	0.004	0.045	0.144	0.109	0.099	0.003	0.466
	MFR	0.007	0.011	0.101	0.032	0.124	0.002	0.001
	TFR	0.993	0.989	0.899	0.968	0.876	0.998	0.999
	MFD (s)	0.07	0.08	0.09	0.06	8.94	0.05	0.20
9	FAR	-	-	-	-	-	0.134	-
	MFR	-	-	-	-	-	0.165	-
	TFR	-	-	-	-	-	0.835	-
	MFD (s)	-	-	-	-	-	0.30	-

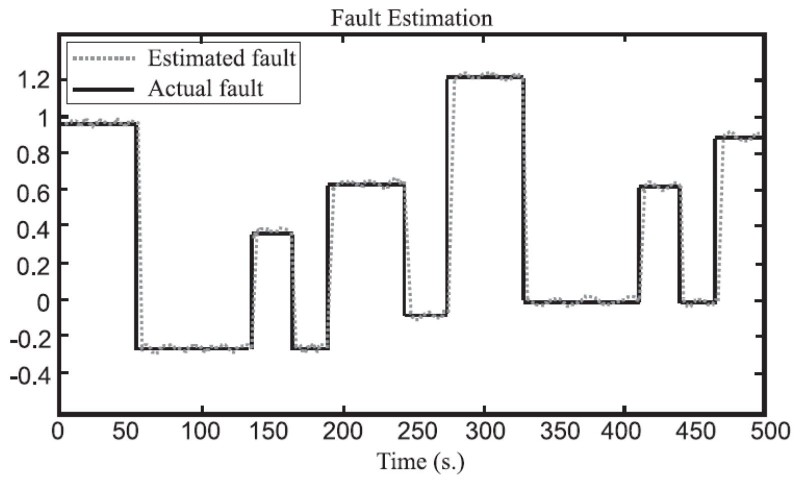

Figure 5.6: The pulse sequence representing the fault 1 injecting into the system in order to test the performance of the designed FTC system. The black line is the actual fault, while the grey line is the estimation of the fuzzy FDI module.

controller, designed in fault-free condition, as addressed in Section 4.3. The following test refers to the simulation of the actuator Fault 1, but the fault shape has been changed into a sequence of rectangular pulses, with different size and length. The considered FDI block, in this case featured by fuzzy estimators, provides an accurate tracking of the fault signal, with minimal detection delay, as depicted in Fig. 5.6. Under this condition, Fig. 5.7 highlights the effectiveness of the FTC, which improves the tracking capabilities, in presence of faults. Indeed, the figure shows the output signal compared with its desired reference value, with and without FTC. In particular the FTC has been applied at $t = 250s$ during a $500s$ simulation in which the fault 1 of Fig. 5.6 has been injected into the system.

The pulse sequence can represent a typical and realistic fault shape, however the fault estimator can be easily generalized to reconstruct different shaped signals, e.g. polynomial faults (Baldi et al., 2013).

Finally, the tracking capability of the proposed FTC strategy has been evaluated in terms of per-cent Normalized Sum of Squared Error (NSSE), generally defined as:

$$\text{NSSE\%} = 100\sqrt{\frac{\sum_{k=1}^{N}\Big(r(k) - y(k)\Big)^2}{\sum_{k=1}^{N} r^2(k)}} \tag{5.1}$$

Where r is the reference signal, y is the output, N the number of acquired sample per simulation. The NSSE has been computed after the execution of 500 Monte Carlo runs, characterized by the parameter uncertainties of Table 5.6, and the result is reported in Table 5.8, for the best, the average, and the worst case occurred over the 500 simulations of the fault 1, 2, and 3, modeled as pulse sequences.

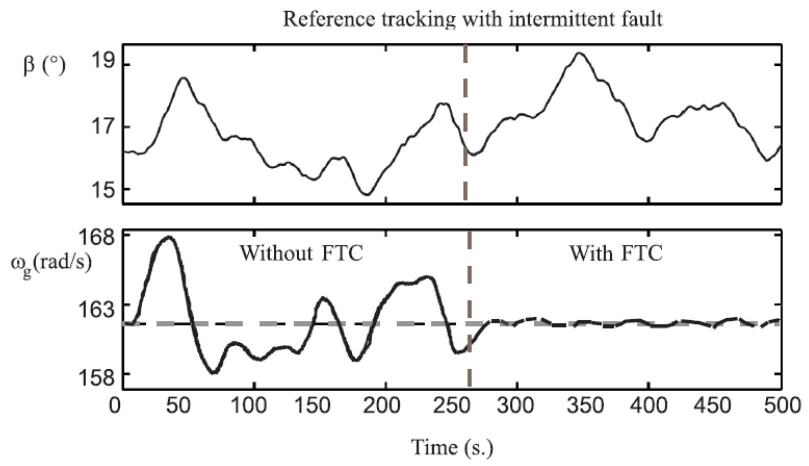

Figure 5.7: The comparison between the tracking of the desired generator speed, without FTC (0 s-250 s) and with FTC (250 s-500 s), together with the control signal β.

Table 5.8: The best, the average, and the worst value of NSSE% relative to the tracking error computed after the simulations of three fault cases.

Simulated fault	Best NSSE%	Average NSSE%	Worst NSSE%
1	8.04	11.23	15.05
2	9.01	12.23	14.74
3	7.96	10.35	13.83

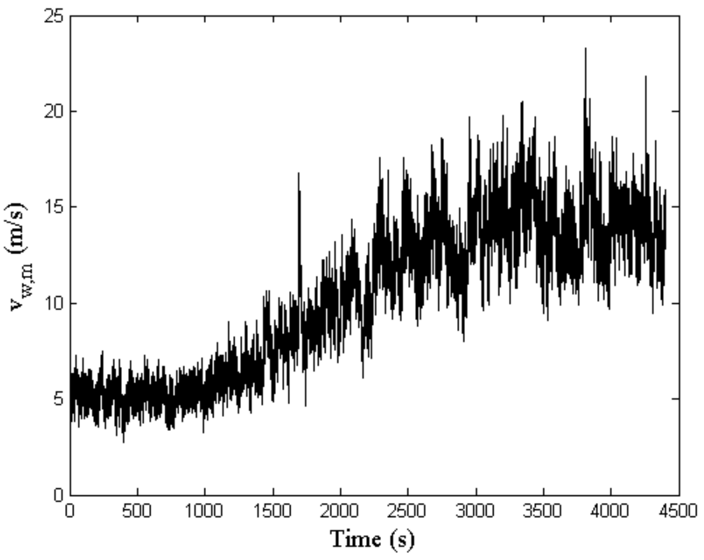

Figure 5.8: The mean wind speed sequence driving the simulations.

5.2 Wind Farm Simulations

The following simulations refers to the wind farm benchmark model associated with the fuzzy fault estimators (Simani et al., 2015b), and highlight how the proposed data-driven fault tolerant control strategy involves performance similar to those related to the single wind turbine system.

Similarly to the wind turbine benchmark, the mean wind sequence driving all the simulations covers the most common operative range from 5 m/s up to 15 m/s, with a peak value of about 23 m/s. The wind and wake submodel processes this sequence in order to generate the actual wind speed signal for all the turbines of the wind farm and for both the measurement musts associated with the two wind scenarios provided by the benchmark model (see Section 2.3), taking into account the disturbances and the interaction among turbines. Fig. 5.8 depicts the leading wind sequence.

The available data consist of 440000 samples of input-output measurements, with a sampling rate of 100 Hz.

The three proposed faults affect different wind turbines at different times, by influencing the measured variables, i.e. β_i, $\omega_{g,i}$, $P_{g,i}$. These faults are difficult to detect at wind turbine level (see e.g. the fault 9 of the wind turbine benchmark that is addressed also in the wind farm as fault 3). However, they can be more easily detected at wind farm level, by comparing the performance of different wind turbines. Table 5.9 shows the affected turbines per fault, together with the occurrence time.

The shape of the simulated faults is the same for all the three cases: they are modeled as an abrupt additive rectangular pulse.

Table 5.9: The turbine on which a fault acts, together with the time during which the fault is active.

Fault	Affected Turbine	Time (s)
1	7	1000-1100
	2	3000-3100
2	1	1300-1400
	5	3300-3400
3	6	1600-1700
	8	3600-3700

5.2.1 Fault Diagnosis

Here, as addressed in (Simani et al., 2014b), the fuzzy fault estimators are designed choosing a number of $n_C = 5$ clusters and $o = 2$ delays on the input and output regressors, related to the signals selected by the FMEA. The TS models exploit Gaussian membership functions. Afterwards, the fault estimators are organized into a dedicated observer scheme that fulfills the isolation of the three faults provided by the benchmark model.

Firstly, the model capabilities of the three estimator can be evaluated in terms of Predicted Per Cent Reconstruction (PPCRE), defined as the per cent difference between the measurements and the estimation, computed in fault free conditions. It expresses the percentage of data not correctly reconstructed by the estimator. As highlighted in Table 5.10, although this index increases when computed on data which is not used for estimation (i.e. validation and test data sets), however, the percentage is always smaller than 5%, thus featuring a good modeling capability.

Table 5.10: The Predicted Per Cent Reconstruction Error of the fuzzy fault estimators, relative to the wind farm fault scenarios.

Data Set	**PPCRE** (%)		
	Fault 1 Estimator	Fault 2 Estimator	Fault 3 Estimator
Estimation	0.90	0.87	0.92
Validation	2.80	1.80	2.10
Test	4.20	3.50	4.00

Then, the detection of the three faults is achieved by means of the residual generated by the fault estimators, after the proper tuning of the threshold parameter δ. In particular, Fig. 5.9 highlights the residual relative to the fault 1 estimator, whose value is bounded by the thresholds when the fault is not active, while it is significantly over the threshold when the fault occurs on the two different turbines. Similar results are achieved by the fault 2 estimator, whose residual is depicted in Fig. 5.10.

Finally, the performances of the fault estimators, in terms of False Alarm Rate

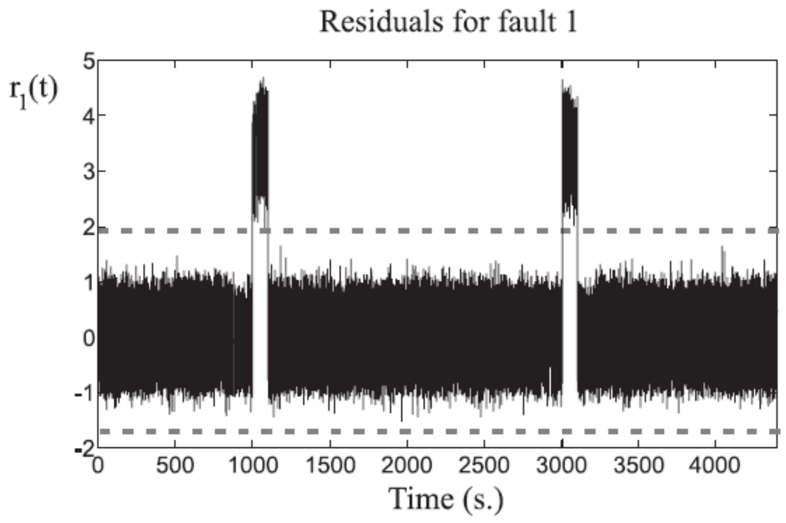

Figure 5.9: The residual generated by the fault 1 estimator (continuous line), with its threshold levels (dotted line).

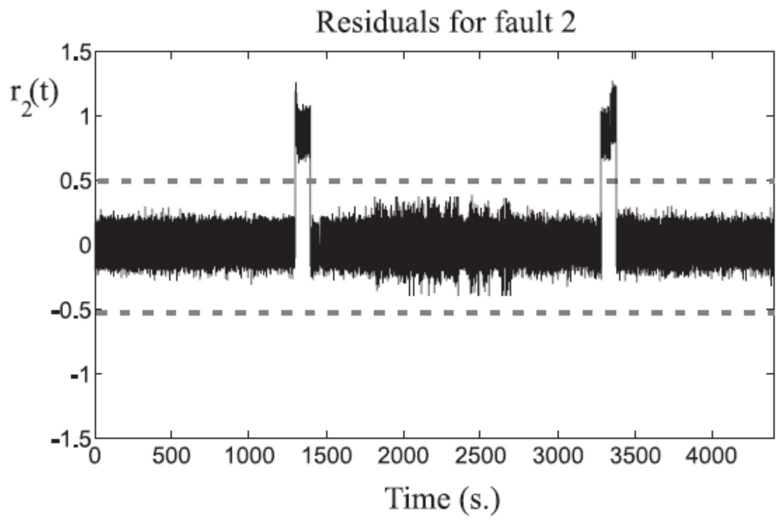

Figure 5.10: The residual generated by the fault 2 estimator (continuous line), with its threshold levels (dotted line).

(FAR), Missed Fault Rate (MFR), True FDI Rate (TFR), Mean FDI Delay (MFD) are computed exploiting a Monte Carlo analysis, in which realistic uncertainties on the model parameters are considered. In order to maintain the consistency of the performance indices, the same values of Table 5.6 are adopted to execute the analysis on the wind farm benchmark. The results are shown in Table 5.11, which reports also the optimized values of the threshold parameter δ.

Table 5.11: The performance indices of the wind farm fault estimators, computed by means of a Monte Carlo analysis.

Fault estimator	FAR	MFR	TFR	MFD (s)	δ
1	0.002	0.003	0.997	0.75	4.8
2	0.001	0.001	0.999	0.95	4.5
3	0.002	0.003	0.997	0.60	4.6

The achieved results lead to some considerations: the proper choice of the δ parameter can provide false alarm and missed fault rates of less than 0.3%, and true detection rates bigger than 99.7%, with minimal mean detection delay. Moreover, the Monte Carlo analysis validates the robustness and the stability of the proposed fault diagnosis scheme, with respect to error, noise and uncertainty.

5.2.2 Comparative Analysis

Based on the considered benchmark model, an international competition has been proposed at the IFAC World Congress 2014, where the developed fuzzy FDI strategy (Simani et al., 2014b) was awarded among the best contributions (Odgaaard and Shafiei, 2015).

Two noteworthy solutions proposed at the Congress are taken into account for the comparison with the fuzzy fault diagnosis module. The first of them is the fault detector developed by (Borchersen et al., 2014), that relies on a Cumulative Sum (CUSUM) method for the evaluation of the residual signals. The generation of the residuals depends on the estimation of the wind speed and direction. This set of information is used to group the different wind turbine of the wind farm with similar operating conditions, so that different turbines in the same group can be compared. This relatively simple approach can represent a common industrial implementation.

The second scheme considered in the comparative analysis is based on the Interval Parity Equation (IPE) (Blesa et al., 2014). Here, the fault detection scheme exploits parity equations and an unknown, but bounded, description of the noise and modeling errors. The detection evaluates if the acquired input-output measurements are within the interval prediction bounds. The isolation is performed by means of observer based schemes.

Similary to the comparative tests carried out in the context of the single wind turbine fault diagnosis, detection time and missed faults rate are used as performance indices. They are calculated in terms of average values, captured over a significant number of simulations.

Table 5.12 and Table 5.13 report the comparisons.

Table 5.12: The mean fault detection delay time of the considered detectors, relative to the three fault cases.

Method	Fault 1 (s)	Fault 2 (s)	Fault 3 (s)
CUSUM	2.2	-	1
IPE	0.8	6.3	1.4
Fuzzy	0.75	0.95	0.60

Table 5.13: The mean missed fault rate of the considered detectors.

Method	Fault 1 (%)	Fault 2 (%)	Fault 3 (%)
CUSUM	0	-	0
IPE	30	30	60
Fuzzy	0.1	0.3	0.1

Although, the CUSUM-based method does not detect the fault case 2, it has the best performance in terms of missed faults, with a small detection delay. The IPE method perform a quick detection of fault cases 1 and 3, but with a high number of missed fault. The fuzzy detector seems to provide the faster detection of all the fault cases (less than 1 s), with a small number of missed faults.

5.2.3 Fault Tolerant Control

As already remarked, the fault diagnosis module has been completed by means of the standard wind farm controller implemented in the benchmark model. The nonlinear filters featured by the fault estimators provide the on-line correction to apply to the faulty input-output signals, according to the block diagram of Fig. 4.6.

In order to test the tracking capability of the proposed FTC strategy, the fault case 1 has been injected into the system changing its shape into a sequence of rectangular pulses of different length and size. Fig. 5.11 shows the actual fault signal compared with the estimated one.

Finally, the effectiveness of the control is highlighted in Fig. 5.12, where the wind farm reference power is compared with the total generated power. More precise information on the tracking performances can be inferred considering the per cent Normalized sum of Squared Error (NSSE%), already defined in Eq. 5.1. Its values, reported in Table 5.14, are computed also in this case by exploiting the Monte Carlo tool.

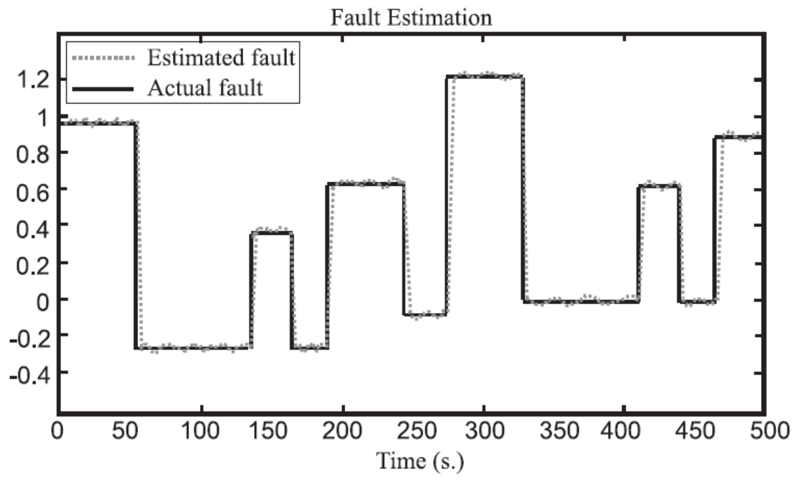

Figure 5.11: The pulse sequence representing the fault 1 injecting into the system in order to test the performance of the designed FTC system. The black line is the actual fault, while the grey line is the estimation of the fuzzy FDI module.

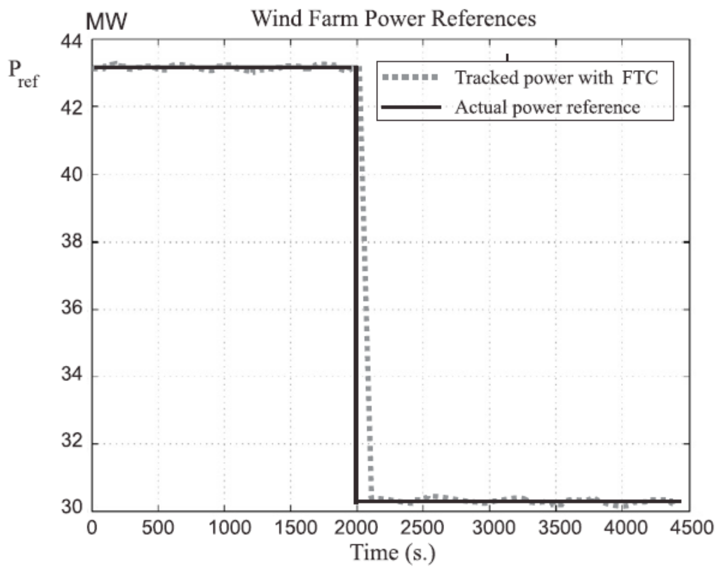

Figure 5.12: The tracking of the reference signal by means of the proposed FTC strategy, in presence of faults.

Table 5.14: The best, the average, and the worst value of NSSE% relative to the wind farm tracking error computed after the simulations of three fault cases.

Simulated fault	Best NSSE%	Average NSSE%	Worst NSSE%
1	11.14	12.63	14.05
2	12.31	13.63	15.74
3	10.46	11.55	12.73

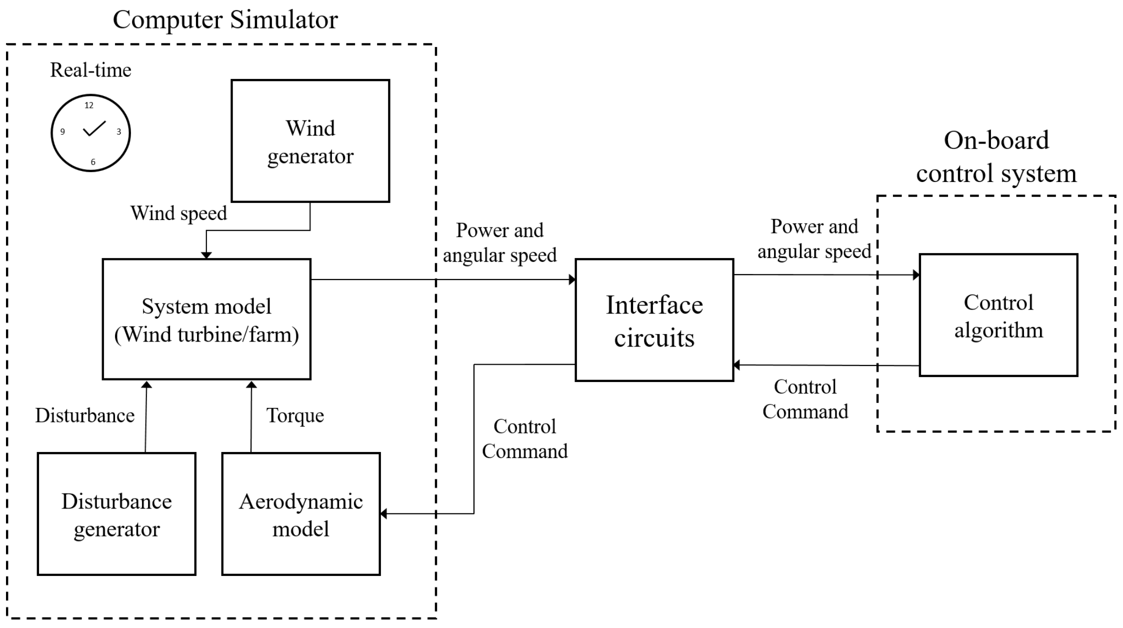

Figure 5.13: The block diagram of the hardware in the loop test rig.

5.3 Hardware in the Loop Test

The *Hardware In the Loop* (HIL) test-rig (Simani, 2012) has been implemented in order to assess the proposed fault diagnosis schemes in more realistic real-time working conditions. These experimental tests aim at validating the noteworthy results obtained in simulations, considering the almost real conditions that the systems under analysis (i.e. the wind turbine and the wind farm) may deal with, during their working situations.

The set-up of the test-rig, represented in Fig. 5.13, consists of three interconnected components:

- **Simulator**: the models of the system dynamics have been implemented in LabVIEW® environment, and consider factors such as disturbance, measurement noise and uncertainty, in addition to the system models described in Chapter 2. This software tool runs on an industrial CPU and allows the real-time monitoring of the simulated system parameters.

- **On board electronics**: The fault tolerant controller has been implemented in the AWC 500 system, which features standard wind turbines specifications. This element receives the signals relative to the generated power and the generator angular rates. Then, it processes the control algorithm, including the fault diagnosis module, and produces the generator torques and pitches command signals transmitted to the simulator.

- **Interface circuits**: they carry out the communication between the simulator and the on board electronics, receiving the output signals from the simulator and transmitting the signal generated by the control algorithm.

Table 5.15 and Table 5.16 refer to the fuzzy fault diagnosis and summarizes the results obtained using this real-time HIL set-up, respectively for the wind turbine and the wind farm simulated system.

It is worth observing the consistency of the almost real-time test with respect to the Monte Carlo analysis above mentioned in Section 5.1 and Section 5.2. Although the performances of the Monte Carlo analysis seem to be better than those obtained using the HIL platform, some issues have to be taken into account. Indeed, the numerical accuracy of the on-board electronics, which involves float calculations is more restrictive than the CPU of the simulator. Moreover, also the analog to digital and the digital to analog conversions can motivate possible deviations. Note that real situations do not require to transfer data from a computer to the on board electronics, so that this error is not actually introduced.

However, the obtained deviations are not critical and the developed control systems can be also considered in real wind turbine applications. 5.16

Table 5.15: The Fault Diagnosis performance indices relative to the wind turbine HIL test.

Simulated fault	FAR	MFR	TFR	MFD
1	0.005	0.005	0.995	0.07
2	0.004	0.004	0.996	0.49
3	0.004	0.004	0.996	0.08
4	0.005	0.005	0.995	0.07
5	0.003	0.004	0.997	0.06
6	0.004	0.005	0.996	0.76
7	0.005	0.004	0.995	0.64
8	0.005	0.004	0.995	0.06
9	0.004	0.005	0.996	0.18

Table 5.16: The Fault Tolerant Control performances, relative to the wind farm HIL test.

Simulated fault	Average NSSE%
1	13.74
2	14.37
3	15.01

Chapter 6

Conclusions and Further Investigations

The results achieved by the proposed system, reported in Chapter 5 lead to several considerations.

This thesis firstly proposes a procedure for the fault detection and isolation of both a single wind turbine and a wind farm benchmark model using data-driven approaches, relying on fuzzy logic and neural networks, and involving a design based on uncertain input-output measurements. It is assumed that the process under investigation is characterized by a complex nonlinear behavior, and its available measurements are normally not very reliable, due to the wind speed unpredictable and uncertain nature.

The residual generators considered here for diagnosis purposes have either the form of Takagi-Sugeno (TS) fuzzy models or recurrent neural networks.

In particular, the fuzzy models were derived using the combination of a clustering technique, called c-means clustering, and an identification procedure aimed at optimizing the coefficients of the TS local affine models, solving the noise-rejection problem.

On the other hand, the neural network residual generators are based on the open-loop Nonlinear AutoRegressive with eXogenous Input (NARX) architecture, as it provide an effective strategy to menage the relations between input-output past and current data. The networks are trained using target patterns coming from noisy measurement acquisitions and exploiting the Levenberg-Marquart backpropagation algorithm.

After the data-driven design of the estimators, they are tested in fault-free condition showing good modeling capabilities. Then, the fault detection and the fault isolation scheme are derived. The first exploits the evaluation of the generated residuals by means of a threshold logic which can be preceded by a filtering function. The threshold levels are optimized on the basis of the statistical properties of the residuals. The isolation scheme uses a bank of generalized or dedicated observers for highlighting the fault location.

Finally, the effectiveness of the proposed approaches is tested on the data acquired from the simulated benchmark systems. The detection and isolation of the faults affecting sensors, component and actuators of the process under diagnosis is properly achieved. The performances are evaluated in terms of false alarms, missed faults and mean detection delay and compared with those of other advanced fault diagnosis schemes. A Monte Carlo analysis validates the robustness and the stability of the fault diagnosis by taking into account the typical variation and uncertainty affecting several model parameters.

As a result of the proper accomplishment of the fault diagnosis task, a fault tolerant controller can be implemented on the basis of the generated fault estimations. Indeed, following an Active Fault Tolerant Control (AFTC) strategy the input-output faulty signals are compensated by means of a second control loop, in order to accommodate the inner control action. In this way, the main controller can be designed for fault-free nominal conditions.

Also the proposed fault tolerant control scheme is tested in the context of the wind turbine and wind farm benchmark models, obtaining interesting performances in terms of tracking errors in faulty conditions. The validation of the controller relies on a Monte Carlo analysis, which demonstrates the robustness against parameter uncertainties and disturbances.

In order to investigate the performances of the overall fault tolerant control scheme in a more realistic real-time framework, the Hardware In the Loop (HIL) test has been carried out. It consists of a simulator, running on an industrial computer, that interacts with the on-board electronics through the interface circuits. The simulation involves the system under monitoring, together with additional noise and disturbances, then, the on-board electronics implements the fault tolerant controller that receive the simulated measurements and generates the control action. The HIL test shows good tracking capability in faulty condition, characterized by performance indices similar to those obtained in the pure simulations.

It is worth noting that further related researches can be carried out, aimed at optimizing the residual generators, in particular the neural network schemes, as different architectures, as well as different training algorithm can be tested and compared, in order to optimize the modeling capability. Furthermore, also the AFTC can be improved by investigating different solutions for the main controller, as the simple pre-existent controller as been exploited in this thesis. Moreover, a noteworthy validation should be fulfilled by testing the proposed AFTC in different benchmark systems, first of all the Fatigue, Aerodynamics, Structures, and Turbulence (FAST) simulator.

However, the overall achieved result induces future studies concerning the application to real wind turbine installations, where the currently adopted control strategies are often too conservative, as they involve the shutdown of the faulty turbine to wait for maintenance services.

Concluding, wind energy is a fast growing industry and this growth implies a large demand for better modeling and control. Possible faults, malfunctions, uncertainties and disturbances make the control a challenging task to overcome. These considerations motivate the need for advanced modeling and further development of sustainable control strategy, with the main objective of reducing the wind energy cost. This clean and renewable energy source should match the global electricity needs, and it represents the proper candidate to solve the future World energy requirements, if the technological barriers will be overcome. The industrial application of sustainable control is still in its prototyping phase, and there exist many opportunities to significantly improve the efficiency and the lifetime of wind farms.

Appendix A

Active FTC via Nonlinear Geometric Approach

With reference to the wind farm benchmark model of Section 2.3, this appendix addresses the development of an Active Fault Tolerant Control (AFTC) system, involving a fault accommodation scheme. The proposed fault diagnosis module, required by the control scheme, is based on adaptive filters designed via the Nonlinear Geometric Approach (NLGA) (De Persis and Isidori, 2001). It generates the on-line fault estimation, that is exploited by a second control loop for compensating the faulty signals. Moreover, the nonlinear filters are decoupled from both the disturbance affecting the measurements and the interaction among wind turbines of the wind farm.

The direct application of the NLGA would be impossible, because of the poor knowledge of the analytical model of the system, particularly concerning the aerodynamics of the wind turbine, that are expressed by means of the look-up tables (e.g. for the power coefficient C_p, see Section 2.1). Therefore, in order to design a disturbance decoupled fault estimation module, the C_p map is approximated as second order polynomial functions. The same approach is used for deriving the wake model, that takes into account the effects of the interactions among turbines.

The first stage of the nonlinear filter design involves the estimation of the disturbance distribution functions, which allow the decoupling of the disturbance acting on the system. These disturbances are mainly due to two terms: the first contribution comes from the wind speed of the i-th turbine, $v_{w,i}$ that affects the system through its power coefficient $C_{p,i}$; the second contribution includes the effects of the interference of the i-th turbine with the j-th turbine, and it is represented by the signals $v_{wm,j}$ of the wake model.

The estimation of the disturbance distribution related to the first disturbance term, follows the procedure described in (Simani and Castaldi, 2014) and includes the estimation of the analytical functions relying on the C_p coefficient, as well as on the wind speed $v_{w,i}$. This method deals with the identification of the nonlinear relations of the uncertainty distribution from input-output data acquired from the turbine. Similarly, also the estimation of the wake disturbance distribution requires an analytical knowledge about the unknown inputs $v_{wm,j}$, that is derived exploiting the same technique.

Afterwards, the derivation of the nonlinear filter can be performed. Their structure is obtained by exploiting a disturbance decoupling scheme belonging to the NLGA framework. In more details, a coordinate transformation highlights a subsystem affected by the fault but decoupled by the disturbances, here represented by the vector $d = [v_{w,i}, v_{wm,j}]$.

The starting point is given by the overall nonlinear model, expressed as:

$$\begin{cases} \dot{x} = n(x) + g(x)c + l(x)f + p_d(x)d \\ y = h(x) \end{cases} \tag{A.1}$$

Where x is the state vector, $c(t)$ is the control input vector, $f(t)$ is the fault signal, $d(t)$ is the disturbance and y is the output vector. $n(x)$, $l(x)$, the columns of $g(x)$ and $p_d(x)$ are smooth vector field, $h(x)$ is a smooth map.

A coordinate change in the state and output space leads to new local state and output $(\bar{x}, \bar{y})$ transforming A.1 into:

$$\begin{cases} \dot{\bar{x}}_1 = n_1(\bar{x}_1, \bar{x}_2) + g_1(\bar{x}_1, \bar{x}_2)c + l_1(\bar{x}_1, \bar{x}_2, \bar{x}_3)f \\ \dot{\bar{x}}_2 = n_2(\bar{x}_1, \bar{x}_2, \bar{x}_3) + g_2(\bar{x}_1, \bar{x}_2, \bar{x}_3)c + l_2(\bar{x}_1, \bar{x}_2, \bar{x}_3)f + p_2(\bar{x}_1, \bar{x}_2, \bar{x}_3)d \\ \dot{\bar{x}}_3 = n_3(\bar{x}_1, \bar{x}_2, \bar{x}_3) + g_3(\bar{x}_1, \bar{x}_2, \bar{x}_3)c + l_3(\bar{x}_1, \bar{x}_2, \bar{x}_3)f + p_3(\bar{x}_1, \bar{x}_2, \bar{x}_3)d \\ \bar{y}_1 = h(\bar{x}_1) \\ \bar{y}_2 = \bar{x}_2 \end{cases} \tag{A.2}$$

This elaboration yields to an observable subsystem, that if exists, is affected by the faults but not by the disturbance. Therefore, the new reference frame can be represented by the $\bar{x}_1$ subsystem:

$$\begin{cases} \dot{\bar{x}}_1 = n_1(\bar{x}_1, \bar{y}_2) + g_1(\bar{x}_1, \bar{y}_2)c + l_1(\bar{x}_1, \bar{y}_2, \bar{x}_3)f \\ \bar{y}_1 = h(\bar{x}_1) \end{cases} \tag{A.3}$$

Where x_2 is denoted with y_2 as it is considered as independent input. NLGA can be designed if the following constraints are satisfied:

- $\bar{x}_1$ is independent from $\bar{x}_3$;
- the fault is a step function of the time;
- there exists a proper scalar component $\bar{x}_{1,s}$ of the state vector $\bar{x}_1$ such that the corresponding scalar component of the output vector is $\bar{y}_{1,s} = \bar{x}_{1,s}$ and the following relation holds: $\dot{\bar{y}}_{1,s} = M_1(t)f + M_2(t)$, with M_1, M_2 functions of input output measurements.

The proposed NLGA adaptive filter deals with a least-square algorithm with forgetting factor β, described by the adaptation law:

$$\begin{cases} \dot{P} = \beta P - \frac{1}{N^2} P^2 \breve{M}_1^2, & P(0) = P_0 > 0 \\ \dot{\hat{f}} = P \epsilon \breve{M}_1, & \hat{f}(0) = 0 \end{cases} \tag{A.4}$$

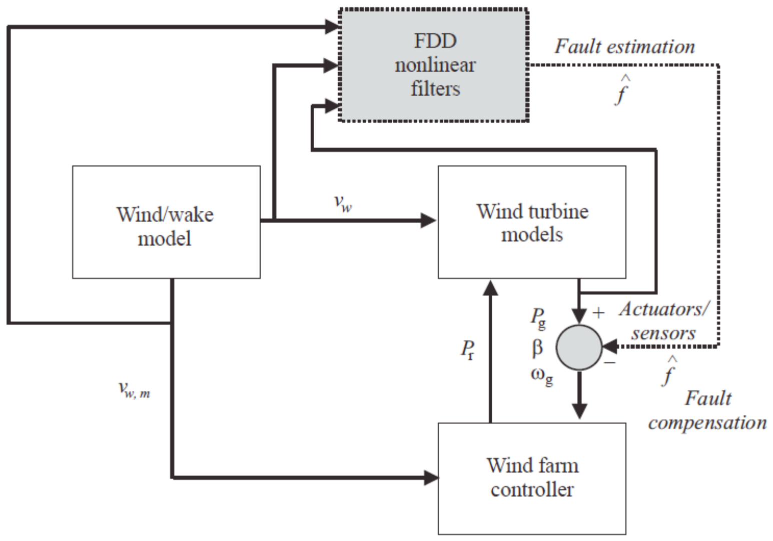

Figure A.1: The scheme of the integrated AFTC solution.

With $N^2 = 1 + \breve{M}_1^2$ normalization factor.

Then, the output estimation and the corresponding normalized estimation error are formulated as:

$$\begin{cases} \hat{\breve{y}}_{1,s} = \breve{M}_1 \hat{f} + \breve{M}_2 + \lambda \hat{\breve{y}}_{1,s} \\ \epsilon = \frac{1}{N^2}(\breve{y}_{1,s} - \hat{\breve{y}}_{1,s}) \end{cases} \tag{A.5}$$

$\lambda > 0$ is a parameter related to the bandwidth of the filter. The variables denoted by the superscript $\breve{\cdot}$ are obtained by means of a low pass filter, as follows:

$$\begin{cases} \dot{\breve{M}}_1 = -\lambda \breve{M}_1 + M_1, & \breve{M}_1(0) = 0 \\ \dot{\breve{M}}_2 = -\lambda \breve{M}_2 + M_2, & \breve{M}_2(0) = 0 \\ \dot{\breve{y}}_{1,s} = -\lambda \breve{y}_{1,s} + y_{1,s}, & \breve{y}_{1,s}(0) = 0 \end{cases} \tag{A.6}$$

Finally, Eq. A.4, A.5, and A.6 provide the considered adaptive filter.

Once the fault diagnosis module has been derived, the fault estimations are used for the compensation of the faulty signals related to the generated power, the pitch angle and the generator speed, namely P_g,β,ω_g, respectively. The compensated signals are exploited by the wind farm pre-existent controller, designed in fault-free nominal conditions, as shown by Fig. A.1.

As a consequence of these corrections, the controller provides the nominal tracking of the reference signal.

With reference to the input-affine model A.1, the state variables and the control input vector, applied to the i-th turbine are:

$$x = [x_1, x_2]^T = [\omega_{g,i}, P_{g,i}]^T \tag{A.7}$$

$$c = [P_{r,i}\beta_i] \tag{A.8}$$

Hereafter the fault case 2 is considered, in which the fault acts on the pitch angle. The result of the modeling leads to:

$$n(x) = \begin{bmatrix} -\frac{\rho A}{2J}0.0010R^3x_1^2 - \frac{1}{J} \\ -p_{gen}x_2 \end{bmatrix} \tag{A.9}$$

$$g(x) = \begin{bmatrix} 0 & \frac{\rho A}{2J}0.0003R^3x_2^2 - \frac{1}{J} \\ p_{gen} & 0 \end{bmatrix} \tag{A.10}$$

$$l(x) = \begin{bmatrix} 0 & \frac{\rho A}{2J}0.0003R^3x_1^2 - \frac{1}{J} \\ 0 & 0.0001 \end{bmatrix} \tag{A.11}$$

$$p_d(x) = \begin{bmatrix} \frac{\rho A}{2J}0.0010R^2x_1 & 0.0011 \\ 0.0002 & \frac{\rho A}{2J}0.0027x_2 \end{bmatrix} \tag{A.12}$$

Where the index i is dropped and the wind turbine parameters are defined as follows: A is area swept by the rotor, R is the rotor radius, ρ is the air density, J is the rotor inertia.

Finally, the design of the NLGA adaptive filter for the reconstruction of the fault 2 is based on the expression:

$$\dot{y}_{1,s} = M_1 f + M_2 \tag{A.13}$$

With:

$$\begin{cases} M_1 = 0.8x_1^2 - 0.036x_1 \\ M_2 = 1.02x_2^2 + 15.7x_2 - 0.3x_1^3 + 0.77x_1^2 \end{cases} \tag{A.14}$$

The derivation of the NLGA adaptive filters for the reconstruction of the fault cases 1 and 3 depends on a different selection of the vector A.11.

The fault diagnosis block provides an accurate estimation of the fault size, with minimal detection delay (see (Simani et al., 2015d) for more details). In order to summarize the tracking capabilities of the proposed strategy, the performance of the AFTC via NLGA has been evaluated in terms of Normalized Sum of Squared tracking Errors (NSSE), calculated by means of a Monte Carlo analysis consisting of more than 500 runs, which considers typical parameters variations.

Table A.1 highlights the comparison between the proposed approach and the FTC relying on fuzzy model described in Section 5.2.

Table A.1: Comparison of achieved performance in terms of average NSSE%.

AFTC methods	Fault 1	Fault 2	Fault 3
AFTC via NLGA	10.33%	11.56%	10.47%
AFTC via fuzzy TS model	14.07%	15.06%	15.34%

Despite the performance of the NLGA adaptive filters seem to be quite better respect to those of the identified TS fuzzy models, they strongly depend on the goodness of the analytical description, rather than the acquired data. Therefore, the application to real system may leads to a decreasing of the performance due to modeling errors that the data-driven approaches should be able to capture and manage more effectively.

Bibliography

ABB (2011). *Technical Application Papers No.13 - Wind power plants.*

Archer, C. L. and Jacobson, M. Z. (2005). Evaluation of global wind power. *Journal of Geophysical Research: Atmospheres (1984–2012)*, 110(D12).

Babuška, R. (2012). *Fuzzy modeling for control*, volume 12. Springer Science & Business Media.

Baldi, P., Castaldi, P., Mimmo, N., and Simani, S. (2013). Satellite attitude active FTC based on geometric approach and RBF neural network. In *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on*, pages 667–673. IEEE.

Beghelli, S., Guidorzi, R. P., and Soverini, U. (1990). The Frisch scheme in dynamic system identification. *Automatica*, 26(1):171–176.

Bezdek, J. C. (2013). *Pattern recognition with fuzzy objective function algorithms.* Springer Science & Business Media.

Bianchi, F. D., De Battista, H., and Mantz, R. J. (2006). *Wind turbine control systems: principles, modelling and gain scheduling design.* Springer Science & Business Media.

Blesa, J., Puig, V., Saludes, J., and Fernandez-Canti, R. M. (2014). Set membership parity space approach for fault detection in linear uncertain dynamic systems. *International Journal of Adaptive Control and Signal Processing.*

Borchersen, A., Larsen, J. A., and Stoustrup, J. (2014). Fault detection and load distribution for the wind farm challenge. In *Proceedings of the 19th IFAC World Congress 2014*, pages 4316–4321.

Bossanyi, E. (2003). Wind turbine control for load reduction. *Wind energy*, 6(3):229–244.

Chen, W., Ding, S. X., Sari, A., Naik, A., Khan, A. Q., and Yin, S. (2011). Observer-based FDI schemes for wind turbine benchmark. In *Proceedings of IFAC world congress*, volume 18, pages 7073–7078.

Darling, D. J. (2008). *The Encyclopedia of Alternative Energy and Sustainable Living. Wind Turbines.* Worlds of David Darling.

De Persis, C. and Isidori, A. (2001). A geometric approach to nonlinear fault detection and isolation. *Automatic Control, IEEE Transactions on*, 46(6):853–865.

Dolan, D. S. and Lehn, P. W. (2006). Simulation model of wind turbine 3p torque oscillations due to wind shear and tower shadow. In *Power Systems Conference and Exposition, 2006. PSCE'06. 2006 IEEE PES*, pages 2050–2057. IEEE.

Fantuzzi, C. and Rovatti, R. (1996). On the approximation capabilities of the homogeneous Takagi-Sugeno model. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 1067–1072. IEEE.

Fantuzzi, C., Simani, S., Beghelli, S., and Rovatti, R. (2002). Identification of piecewise affine models in noisy environment. *International Journal of Control*, 75(18):1472–1485.

Gasch, R. and Twele, J. (2011). *Wind power plants: fundamentals, design, construction and operation.* Springer Science & Business Media.

Gong, X. and Qiao, W. (2013). Bearing fault diagnosis for direct-drive wind turbines via current-demodulated signals. *Industrial Electronics, IEEE Transactions on*, 60(8):3419–3428.

Graaff, A. J. and Engelbrecht, A. P. (2012). Clustering data in stationary environments with a local network neighborhood artificial immune system. *International Journal of Machine Learning and Cybernetics*, 3(1):1–26.

GWEC (2015). Global Wind Energy Council. Global wind statistics 2014. Technical report.

Hagan, M. T., Demuth, H. B., Beale, M. H., and De Jesús, O. (1996). *Neural network design*, volume 20. PWS publishing company Boston.

Hagan, M. T. and Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993.

Hameed, Z., Hong, Y., Cho, Y., Ahn, S., and Song, C. (2009). Condition monitoring and fault detection of wind turbines and related algorithms: A review. *Renewable and Sustainable energy reviews*, 13(1):1–39.

Haykin, S. S., Haykin, S. S., Haykin, S. S., and Haykin, S. S. (2009). *Neural networks and learning machines*, volume 3. Pearson Education Upper Saddle River.

Isermann, R. (1997). Supervision, fault-detection and fault-diagnosis methods an introduction. *Control engineering practice*, 5(5):639–652.

Isermann, R. and Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Jensen, N. O. (1983). *A note on wind generator interaction.*

Johnson, K. E., Pao, L. Y., Balas, M. J., and Fingersh, L. J. (2006). Control of variable-speed wind turbines: standard and adaptive techniques for maximizing energy capture. *Control Systems, IEEE*, 26(3):70–81.

Jun, W., Shitong, W., and Chung, F.-l. (2011). Positive and negative fuzzy rule system, extreme learning machine and image classification. *International Journal of Machine Learning and Cybernetics*, 2(4):261–271.

Kumar, A. and Stol, K. (2010). Simulating feedback linearization control of wind turbines using high-order models. *Wind Energy*, 13(5):419–432.

Laouti, N., Sheibat-Othman, N., and Othman, S. (2011). Support vector machines for fault detection in wind turbines. In *Proceedings of IFAC world congress*, volume 2, pages 7067–707.

Liu, G. P. (2012). *Nonlinear identification and control: a neural network approach.* Springer Science & Business Media.

Mahmoud, M., Jiang, J., and Zhang, Y. (2003). *Active fault tolerant control systems: stochastic analysis and synthesis*, volume 287. Springer Science & Business Media.

Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis. *Computers, IEEE Transactions on*, 100(12):1182–1191.

Manjock, A. (2005). Design codes FAST and ADAMS for load calculations of onshore wind turbines, 2005. *National Renewable Energy Laboratory (NREL): Golden, Colorado, USA.*

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441.

Medsker, L. and Jain, L. C. (1999). *Recurrent neural networks: design and applications.* CRC press.

Odgaaard, P. F. and Shafiei, S. E. (2015). Evaluation of wind farm controller based fault detection and isolation. *IFAC-PapersOnLine*, 48(21):1084–1089.

Odgaard, P. and Patton, R. (2012). FDI/FTC wind turbine benchmark modelling. In *Workshop on Sustainable Control of Offshore Wind Turbines; Patton, RJ, Ed.*

Odgaard, P. F. and Stoustrup, J. (2012). Results of a wind turbine FDI competition. In *8th IFAC symposium on fault detection, supervision and safety of technical processes*, pages 102–107.

Odgaard, P. F. and Stoustrup, J. (2013). Fault tolerant wind farm control. A benchmark model. In *Control Applications (CCA), 2013 IEEE International Conference on*, pages 412–417. IEEE.

Odgaard, P. F. and Stoustrup, J. (2015). A benchmark evaluation of fault tolerant wind turbine control concepts. *Control Systems Technology, IEEE Transactions on*, 23(3):1221–1228.

Odgaard, P. F., Stoustrup, J., and Kinnaert, M. (2013). Fault-tolerant control of wind turbines: A benchmark model. *Control Systems Technology, IEEE Transactions on*, 21(4):1168–1182.

Ozdemir, A. A., Seiler, P., and Balas, G. J. (2011). Wind turbine fault detection using counter-based residual thresholding. In *Proceedings of IFAC world congress*, volume 18, pages 8289–8294.

Parker, M. A., Ng, C., and Ran, L. (2011). Fault-tolerant control for a modular generator–converter scheme for direct-drive wind turbines. *Industrial Electronics, IEEE Transactions on*, 58(1):305–315.

Patton, R. J. (2015). Fault-tolerant control. *Encyclopedia of Systems and Control*, pages 422–428.

Patton, R. J., Frank, P. M., and Clarke, R. N. (1989). *Fault diagnosis in dynamic systems: theory and application.* Prentice-Hall, Inc.

Rovatti, R. (1996). Takagi-Sugeno models as approximators in sobolev norms: the siso case. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 1060–1066. IEEE.

Simani, S. (2012). Application of a data-driven fuzzy control design to a wind turbine benchmark model. *Advances in Fuzzy Systems*, 2012:1.

Simani, S. (2015). Overview of modelling and advanced control strategies for wind turbine systems. *Energies*, 8(12):13395–13418.

Simani, S. and Castaldi, P. (2013). Data-driven and adaptive control applications to a wind turbine benchmark model. *Control Engineering Practice*, 21(12):1678–1693.

Simani, S. and Castaldi, P. (2014). Active actuator fault-tolerant control of a wind turbine benchmark model. *International Journal of Robust and Nonlinear Control*, 24(8-9):1283–1303.

Simani, S., Fantuzzi, C., and Patton, R. J. (2003). *Model-Based Fault Diagnosis Techniques.* Springer.

Simani, S., Fantuzzi, C., Rovatti, R., and Beghelli, S. (1999). Parameter identification for piecewise-affine fuzzy models in noisy environment. *International Journal of Approximate Reasoning*, 22(1):149–167.

Simani, S., Farsoni, S., and Castaldi, P. (2014a). Fault tolerant control of an offshore wind turbine model via identified fuzzy prototypes. In *Control (CONTROL), 2014 UKACC International Conference on*, pages 486–491. IEEE.

Simani, S., Farsoni, S., and Castaldi, P. (2014b). Residual generator fuzzy identification for wind farm fault diagnosis. In *Proceedings of 19th IFAC World Congress 2014*, pages 4310–4315.

Simani, S., Farsoni, S., and Castaldi, P. (2015a). Fault diagnosis of a wind turbine benchmark via identified fuzzy models. *Industrial Electronics, IEEE Transactions on*, 62(6):3775–3782.

Simani, S., Farsoni, S., and Castaldi, P. (2015b). Fault-tolerant control of an offshore wind farm via fuzzy modelling and identification. *IFAC-PapersOnLine*, 48(21):1345–1350.

Simani, S., Farsoni, S., and Castaldi, P. (2015c). Wind turbine simulator fault diagnosis via fuzzy modelling and identification techniques. *Sustainable Energy, Grids and Networks*, 1:45–52.

Simani, S., Farsoni, S., Castaldi, P., and Mimmo, N. (2015d). Active fault-tolerant control of offshore wind farm installations. *IFAC-PapersOnLine*, 48(21):1351–1356.

Stamatis, D. H. (2003). *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press.

Svärd, C., Nyberg, M., and Stoustrup, J. (2011). Automated design of an FDI system for the wind turbine benchmark. *JCSE-Journal of Control Science and Engineering*, 2012:19.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *Systems, Man and Cybernetics, IEEE Transactions on*, (1):116–132.

Zhang, X., Zhang, Q., Zhao, S., Ferrari, R. M., Polycarpou, M. M., and Parisini, T. (2011). Fault detection and isolation of the wind turbine benchmark: An estimation-based approach. In *Proceedings of IFAC world congress*, volume 2, pages 8295–8300.