MACHINE LEARNING BASED RENEWABLE ENERGY SOLUTIONS FOR SMART GRID TECHNOLOGIES

A Thesis

presented to

the Faculty of Engineering

at Notre Dame University-Louaize

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

DANNY KHOURY

Dr. Fakherdine Keyrouz, Thesis Supervisor

MAY 2019

Notre Dame University-Louaize

Faculty of Engineering

Department of Electrical, Computer and Communication Engineering

We hereby approve the thesis of

Danny Khoury

Candidate for the degree of Master of Science in Electrical and Computer Engineering

Dr. Fakherdine Keyrouz, Associate Professor, ECCE Fakherdine Keyrouz
Supervisor

Dr. Semaan Georges, Chairperson, ECCE Georges
Member of committee

Dr. Nassar Mendalek, Associate Professor, ECCE
Member of committee

Abstract

Smart grid engineering is the key for an optimized use of extensive energy resources which allows the hybrid renewable energy sources microgrid to be integrated and therefore dispatch their power generation to the grid over long distance DC transmission lines using the HDVC transmission technologies. However, the required number of generating units of wind-turbine generators and photovoltaic arrays, and the associated storage capacity for standalone and/or grid connected hybrid microgrid is determined using a sizing algorithm based on the observation that the state of charge of battery should be periodically invariant which results in a microgrid optimum cost.

However the intermittency of wind speed and solar irradiance are very challenging to power production from wind turbines and photovoltaic arrays. In this context, the Convolution Neural Network (CNN) can symbolize a practical and reliable tool to precisely monitor and predict the wind speed and solar irradiance outputs and accordingly manage the power transfer switching between areas that have surplus renewable energies to areas that have shortage in energies by initiating the Energy Management System (EMS) to dispatch power according to schedule.

The efficiency of the proposed model CNN was tested using meteorological data related to Lebanon, Beirut city. The experimental results indicate that the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values related to wind speed and solar irradiance are small, demonstrating very high forecasting accuracy.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would first like to thank my thesis advisor Dr. Fakherdine Keyrouz. He was always ready for help whenever I had a question about my research and he always pointed me in the right direction through his insightful comments.

My sincere thanks also go to Dr. Nassar Mendalek for enlightening me in writing the HRES microgrid section of this thesis.

I would like to express my sincere gratitude to Chairperson Dr. Semaan Georges for his continuous support, for his encouragement and guidance throughout My Master's Degree.

Last but not the least, I would like to thank my family especially my wife and daughter for being such a great inspiration for me.

Chapter 1: Introduction and Literature Review

Smart grid engineering is the key for an optimized use of extensive energy resources. It is a modernized electrical grid that uses analogue or digital information and communications technology (ICT) [3]. However, the requirements of smart grid are quite different and disruptive, and, therefore, the re-engineering of the existing grid is essential. The engineering work will result in enhancements and extension to the existing grid, as well as the development and deployment of an extensive two-way communication systems [1].

Nevertheless, the utility transmission lines, distribution feeders, switches breakers, and transformer will remain the core of the transmission and distribution (T&D) infrastructure. Thus, the "smarts" in the T&D systems are typically related to advances in the monitoring and control, and protection of the existing infrastructure [1]. The role that communication, networking, and middleware technologies will have in the transformation of existing electric power system into smart grids is exhaustive [2].

Hence, improving grid reliability and operation is mainly done by monitoring real-time power flow and improving voltage control to optimize efficient power delivery and eliminate waste and oversupply [1].

The fusion of renewable energy generation on the grid is a key topic of today's global power systems because it is aiming to reduce the CO2 emissions, to stop or at least reduce the global warming effect, new "CO2-free" technologies are being investigated to fulfil the global demand growth of energy [1].

Moreover, the required number of wind-turbine and photovoltaic array generating units, and the associated storage capacity for on-grid hybrid microgrid is developed using a simple algorithm based on the observation that the state of charge of battery should be periodically invariant [5].

In addition, the battery bank forms the energy storage system that can supply the load when there is shortage of electricity, and store the surplus power when the power

generated from other renewable energy source exceeds the load. Therefore, the energy storage system is essential to cover the shortage of the renewable energy's unpredictable and fluctuant nature. The optimal sizing of hybrid microgrid is done in the aim of minimizing the life cycle cost of the system while the given load power demand can be satisfied without rejection [5].

The hybrid renewable energy source microgrid can be interfaced with the AC bus line and afterwards to the utility grid directly or via a common DC bus by using the appropriate power converters. The AC bus-linked HRES configuration reduces the number of power conversion stages and losses in power transferred to the load/ utility [6].

However, under certain circumstances, it becomes economically beneficial where a large amount of power is to be transmitted over DC transmission lines and over a long distance. This breakeven distance for high voltage DC (HVDC) overhead transmission lines usually lies somewhere in a range of 480 - 640 km. The HVDC transmission system has the ability to interconnect two AC systems which are out of synchronism and operating at two different frequencies [7].

However, changes in the use of the transmission and distribution (T&D) system and operation in a smarter grid will create significant challenges. One of the challenges is to reduce the grid congestion, ensure stability and security and optimize the use of transmission assets and cost generation source. Thus, the grid must be equipped with transmission system that requires advanced technologies such as flexible AC transmission systems (FACTS) and HVDC to support the power flow control and ensure stability. Hence, the supply and demand are continuously balanced by dispatching the appropriate level of generation to satisfy load [1].

Electrical power transmission systems mainly consist of overhead line (OHL), extruded cable, mass-impregnated (MI) cable, and oil-filled (OF) cable, gas insulated line (GIL), superconducting cable and superconducting transmission line systems. The converter stations are required in order to control and manage the transmission power at different stages in a DC power transmission system. The voltage source converters (VSCs) stations are based on insulated-gate bipolar transistors (IGBTs), and HVDC classic stations with the use of converter transformers [20].

With the introduction of HVDC stations using line-commutated converters (Fig. 4) or voltage source converters (Fig. 5), electrical power can be transmitted over long distances at different power levels. IGBTs VSC uses high-frequency (up to 2,000 Hz) pulse-width modulation, and thus the use of small filters and independent control of active and reactive power are possible [20].

The VSC transmission system can transmit active power in two directions, with the same control setup and the same main circuit configuration. Hence, the active power transfer can be quickly reversed without any change of control mode, and without any filter switching or converter blocking. The power reversal is obtained by changing the direction of the DC current and not by changing the polarity of the DC voltage as with HVDC classic [20].

Utilizing Internet of Things (IoT) technology in smart grid is an important approach integrating advanced sensing and communication technologies that can effectively avoid or reduce the damage of natural disasters to the transmission lines, improving the reliability of power transmission and reducing economic loss. In order for IoT to sense and identify the physical world, it employs a variety of sensing devices such as radio frequency identification (RFID) devices, infrared sensors, Global positioning system (GPS), and laser scanners [18].

The system network topology can be cluster-chain type, where each sensor can communicate to the nearby backbone node directly, and the communication links between the sensors and the backbone nodes are generally unidirectional [18].

Each backbone node can communicate to maximum 256 sensors. The distance between the backbone nodes is several hundred meters, and the communication link between the backbone nodes is bidirectional. Parts of the backbone nodes are able to access to the public network through 3G, TD-LTE or power optical network.

The specific monitoring covers transmission tower leaning, conductor galloping, wind deviation, micro-meteorology, conductor icing, wind vibration, and conductor temperature [18].

Physically, in HRES configuration, a group of PV panels are interfaced through a DC–DC converter to regulate their fluctuating DC output. The wind turbine coupled with a permanent magnet synchronous generator (PMSG) generates a three-phase AC

voltage, and its amplitude and frequency vary with rotor speed. Therefore, the wind turbine generator is connected to the DC bus via a rectifier and DC–DC converter.

The storage battery is connected to the DC bus through a bidirectional DC–DC converter to maintain a stable supply–demand balance at its rated capacity. The common DC bus collects the regulated power from various RESs to supply and maintain a constant DC voltage at the input terminal of the DC–AC inverter. A single DC–AC inverter is used to interface the common DC bus to the AC bus connected to the utility grid [6].

One of the benefits of using smart grid solutions and applications, is that substations will be transformed from conventional to smart substations leading to more advanced local analytics, and the management of vast amount of data. Consequently, smart grid enables efficient transfer of surplus power generated from HRESs in existing locations to other remote locations.

The Artificial intelligence (AI) techniques, such as expert systems (ESs), fuzzy logic (FL), and artificial neural networks (ANNs) have enhanced power electronics and power engineering providing powerful tools for design, simulation, control, estimation, fault diagnostics, and fault-tolerant control in modern smart grid (SG) and renewable energy systems (RESs) [16].

The intermittency of wind speed and solar irradiance is very challenging for energy production via wind turbine and PV arrays to synchronize with the power system. The crucial role that the artificial neural network (ANN) technologies lies in the accuracy of forecasting wind and solar energies for a better effective power system management. The basic building block of ANNS is a simple processing unit called neuron organized in interconnected layers. The computational capabilities are determined by the connection weights, network architecture, and training algorithm [8].

With the recent developments in Neural Network (NN) and Deep Learning (DL) new approaches based on deeper architectures such n-layers Perceptron (MLP), Convolution Neural Networks (CNN) and Recurrent Neural Network (RNN) are being classified as the main three in optimal forecasting methods that extract and learn the shape of the wind and solar patterns. In this architecture, each layer of the network has only forward connections with the subsequent layer. Depending on the role and

position of a layer, input, hidden, and output layer types are the main elements of a Feed Forward Network (FNN). An input or data fed to the input layers of MLP is processed by the hidden layers and the result is delivered at the output [21].

Neural networks have an activation function on each neuron that acts on the inputs received and generates an output, plus a backpropagation algorithm that optimizes the weights on each connection in a process to find the optimal combination for the output. Neural networks are non-linear which allows them to produce better results than linear models on wind and solar data time series [21].

RNNs are designed to process sequential data by, most importantly, sharing parameters between the different layers and neurons, generating cycles in the graph sequence of the network. In this sense, RNNs can have extensive memory. In RNN each output is a function of the previous elements. Thus, the values in a specific step will influence its value in future steps. RNNs have the potential to learn from patterns in the time series to predict the future [21].

The effectiveness of using CNN for level energy load forecasting was tested and results obtained from the CNN were compared to those obtained by ANN, Support Vector Machines (SVM), Factored Restricted Boltzmann Machines (FCRBM), and other deep architectures Long Short Term Memories (LSTM). Experimental results showed that the CNN outperformed SVR while having comparable results to the ANN and deep learning methodologies [13].

CNNs have the advantage of processing big data and addressing it in form of a two dimensional matrix, widely applied in the field of image processing or time series. CNN weight sharing network structure which is similar to a biological NN can reduce the complexity of the network model and the number of weights [10]. Convolution networks extract relevant features from small areas of the matrix, by identifying short intervals of the time series that could bring relevant information to the prediction task. The information could be that some patterns are relevant for the future behaviour of wind and solar series [21].

The structure of CNN is a feed-forward neural network. It uses the back-propagation algorithm to optimize the network structure to solve unknown parameters in the network. It starts with the pre-processing of the samples and the required features

are extracted and classified or determined by regression analysis in order to obtain the output [10].

The pre-processing is implemented using convolution filters that act repeatedly on the receptive field and extract the local features of the input matrix with the convolution kernel of the matrix. Further, the pooling process calculates the average or maximum in each section of the convolved feature. After pooling, the dimension of the matrix is considerably decreased and the generalization ability of the model is increased [10].

In 1D convolution neural network, the convolution layers read the input multi-time series signal using a kernel filter that reads in small segments at a time and steps across the entire segment input field. The pooling layer, then, takes the feature map projections and filter them to the most essential elements such as using a signal averaging or signal maximizing process. The output of the convolution and pooling layers network is one or more fully connected (FC) layers that interpret the internal representation and output a vector representing the multi time steps predicted [23].

The development of the multi-step time series forecasting model with CNN starts with defining the prior hours/days subsequence data as input that enables the model to read and learn to extract features from.

The 1D CNN model data consists of [samples, timesteps, features] shape, each sample consists of a specified number of time steps with one feature each defined number of time steps. The recorded dataset is divided into training and testing data, so the shape of the training or testing dataset is [training/testing, timesteps, features].

The first step is to flatten the data to transform the 2D into 1D problem and then iterate over the time steps and each iteration moves along one timesteps and predicts the subsequent hours/days [23].

Furthermore, this multi-step series forecasting problem is an auto regression time series model which uses observations from previous steps as input to an equation to predict the value of the next time step which can result in accurate forecasts on a range of time series problems.

The built model consists of one convolution layer with different number of progressively incremented filters and a kernel size of 1. This means that the input sequence will be read with a convolutional operation one timestep at a time and this

operation will be performed according to the number of filters following by a pooling layer which will reduce these feature maps size before the internal representation is flattened to one long vector. Finally, the outcome is then interpreted by a fully connected layer before the output layer predicts the next hours/days in the sequence [23].

The Mean Squared Error (MSE) loss function is chosen to be the error metric of Root-Mean-Square-Error (RMSE) and Mean Absolute error (MAE) is used as evaluation indicator. In order to improve the training performance of the deep learning model, the adaptive moment estimation (Adam) was applied, which is an extension of stochastic gradient descent (DSG), to adjust parameters. Adam can iteratively update weights of neural networks based on training data [22].

The Rectified Linear Unit (ReLU) activation function was selected to limit output value range to [0, 1].

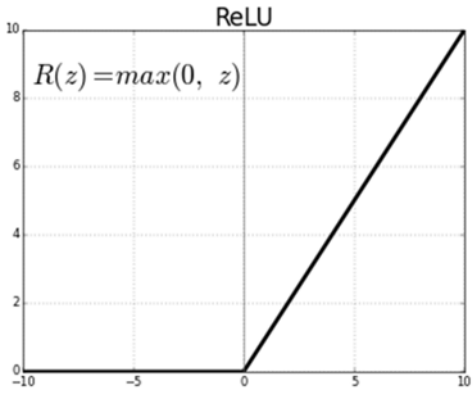


Figure 1 - The activation function - ReLU

1.1 SUMMARY AND IMPLICATIONS

In this context, the Convolution Neural Network (CNN) can symbolize a practical and reliable tool to precisely monitor and predict the wind speed and solar irradiance outputs and accordingly manage the power transfer switching between areas that have surplus renewable energies to areas that have shortage in energies by initiating the Energy Management System (EMS) to dispatch power according to schedule.

Table 9 represents the comparison between the 3 day wind speed forecast resulting from both old and proposed models using the MAE and RMSE error indicators.

The WindNet model in [22] reached respective RMSE and MAE values of 0.999978 and 0.800227. According to table 8, the proposed model produced more efficient results compared to old models reaching a MAE average of 0.693418 and RMSE average of 0.884.

Chapter 2: Smart Grid (SG)

Smart grid engineering is the key for a beneficial use of extensive energy resources, it is a modernized electrical grid that uses analogue or digital information and communications technology (ICT) [3]. However, the requirements of smart grid are quite different and disruptive, and therefore the re-engineering of the existing grid is essential. The engineering work will result in enhancements and extension to the existing grid, as well as the development and deployment of an extensive two-way communication systems [1].

Nevertheless, the transmission lines, distribution feeders, switches breakers, and transformer will remain the core of the utility transmission and distribution (T&D) infrastructure. Thus, the "smarts" in the T&D systems are typically related to advances in the monitoring, control, and protection of the existing infrastructure [1]. The role that communication, networking, and middleware technologies will have in the transformation of existing electric power system into smart grids is exhaustive [2].

As an example of the smart grid application benefits, the integration of the Advanced Metering Infrastructure (AMI) at the power distribution end of the network will shift the consumers' status from a passive to an active participant in the electricity network by giving them access to time of use rates and real time pricing signals that will help them save on electricity bills and cut their power usage in peak hours during a certain interval [1].

Hence, improving grid reliability and operation is mainly done by monitoring real-time power flow and improving voltage control to optimize efficient power delivery and eliminate waste and oversupply [1].

The fusion of renewable energy generation on the grid is a key topic of today's global power systems because it is aiming to reduce the CO2 emissions to stop or at least reduce the global warming effect, new "CO2-free" technologies are being investigated to fulfil the global demand growth of energy [1].

However, changes in the way the T&D system is utilized and operated in a smarter grid will create significant challenges. One of the challenges is to reduce the grid congestion, ensure stability and security and optimize the use of transmission

assets and cost generation source. Thus, the grid must be equipped with transmission system that requires advanced technologies such as Flexible AC transmission Systems (FACTS) and HVDC to support the power flow control and ensure stability. Hence, the supply and demand are continuously balanced by dispatching the appropriate level of generation to satisfy load [1].

2.1 SMART GRID MAJOR COMPONENTS

2.1.1 High level of perspective

The smart grid can be considered to contain the following major components:

- Smart sensing and metering technologies that provide faster and more accurate response for consumer options such as remote monitoring, time-of-use (TOU) pricing, and demand side management (DSM).
- An integrated, standards-based, two way communications infrastructure that provides and open architecture for real time information and control to every endpoint on the grid.
- Advanced control methods that monitor critical components, enabling rapid diagnosis and precise responses appropriate to any event in a "self-healing" manner.
- A software system architecture with improved interfaces, decision support analytics, and advanced visualization that enhances human decision making, effectively transformation grid operators and angers into knowledge workers [1].

2.1.2 Interoperability Between different SG components

The interoperability between different smart grid components is vital, the below framework defines the components at three levels:

1. The electrical infrastructure level.
2. The smart infrastructure level.
3. The smart grid solution level.

Table 1 - Smart Grid Technology Framework – Functionality [1].

	Utility Enterprise Applications				
Smart Grid Solutions	**Operational Efficiency**	**Reliability and Security**	**Energy Efficiency**	**Alternative Energy**	**Consumer Participation**
Smart Infrastructure	**Engineering and Operational Systems**				
	Communications infrastructure				
	Smart Sensors, Controllers, and Meters				
Electrical Infrastructure	**T&D Infrastructure**				
	Alternative energy sources RES, Storage				
	Energy consumer home area network				

The Smart Grid is composed of four main subsystems:

1. Power generation.
2. Transmission.
3. Distribution.
4. Utilization.

And three types of networks:

1. Wide area network (WAN).
2. Neighbourhood area network (NAN).
3. Home area network (HAN).

The power flows through the subsystems while the information flows through the networks as shown in figure 2 [4].

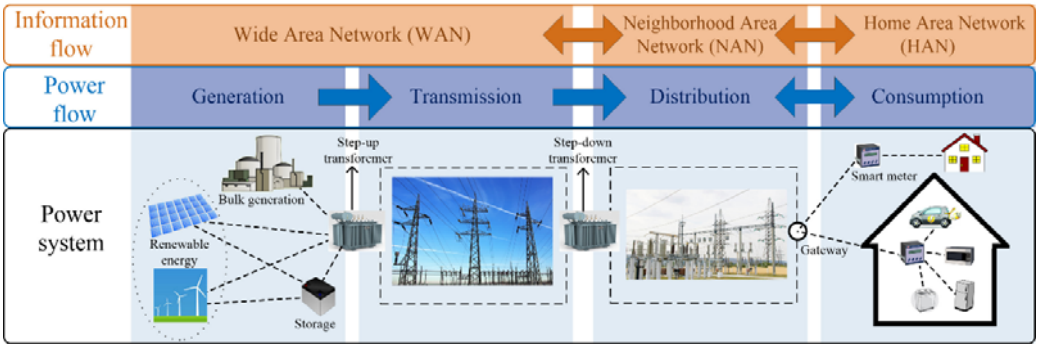


Figure 2 - Smart Grid Components [4].

2.2 SMART SUBSTATION

Substations will be transformed from conventional to smart substations leading to more advanced local analytics, and the management of large amount of data. Consequently, smart grid enables efficient transfer of surplus power generated from HRESs in existing location to other remote locations.

The general functions of a substation include the following:

1. Voltage transformation.
2. Connection point for transmission and distribution power line.
3. Switchyard for electrical transmission and/or distribution system configuration.
4. Monitoring point control center.
5. Protection of power lines and apparatus.
6. Communication with other substations regional control center.

The operational Real-time data are instantaneous values of power system analog and status points such as Volts, Amps, MW, MVAR, circuit breaker status and switch position. These data are time critical and are used to protect, monitor, and control the power system field equipment.

The capability to exchange data and control signals with another devices over a communications link are through microprocessor-based intelligent electronic devices (IEDs) [1].

Figure 3 conceptually illustrates the major components of a SCADA system. The SCADA master hardware and software is typically located centrally at the control

center. The primary SCADA system is often redundant, with a local backup system and/or remote backup at another site. Various types of communications links to the remote terminal units (RTUs) are used.

a. Master Station.

b. SCADA.

c. Communication links (based on wireless and wired technologies).

d. Remote Terminal Units (RTU).

e. Intelligent Electronic Devices (IED).

f. Programmable Logic Controllers (PLC).

g. Smart Meters.

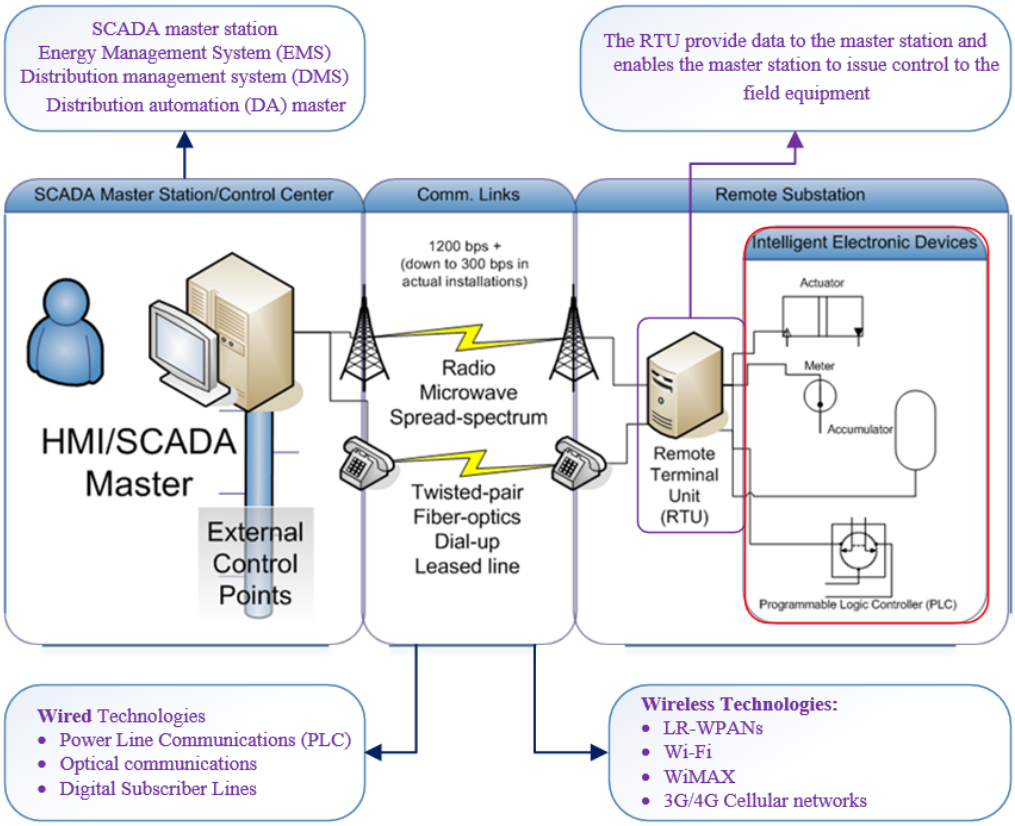


Figure 3 - Major components of a SCADA system [1].

2.2.1 Master Station

The master station is a computer system responsible for communicating with the field equipment and includes a human machine interface (HMI).

A Large utility Master Station has the following:

a. One or more data acquisition servers (DAS) or front-end processors (FEP) that interface with the field devices via the communications system.

b. Real-time data server(s) that contains real-time database(s) (RTDB).

c. Historical server(s) that maintains historical database.

d. Application server(s) that runs various Energy Management System (EMS) applications.

e. Operator workstations with an HMI.

In general, master stations can be divided into five different categories based on their functionality. However, in some cases the functions can cross over from one type of system to another:

- SCADA master station.
- SCADA master station with automatic generation control (AGC).
- Energy Management System (EMS).
- Distribution management system (DMS).
- Distribution automation (DA) master.

SCADA master station primary functions:

- Data acquisition.
- Remote control.
- User interface.
- Areas of responsibility.
- Historical data analysis.
- Report writer.

SCADA/AGC system primary functions (in addition to SCADA master station):

- AGC.
- Economic dispatch (ED)/hydroallocator.
- Interchange transaction scheduling.

EMS primary functions (in addition to SCADA/AGC system):

- Network configuration/topology processor.
- State estimation.
- Contingency analysis.
- Three phase balanced operator power flow.
- Optimal power flow.
- Dispatcher training simulator.

DMS primary functions:

- Interface to automated mapping/facilities management (AM/FM) or geographic information system (GIS).
- Interface to customer information system (CIS).
- Interface to outage management.
- Three phase unbalanced operator power flow.
- Map series graphics.

DA system primary functions:

- Two-way distribution communications.
- Fault identification/fault isolation/service restoration.
- Voltage reduction.
- Load management.
- Power factor control.
- Short-term load forecasting.

All types of master stations are interfaced with the field devices. Historically, in electric utilities, these devices were RTUs, in recent years, with the proliferation of IEDs, many of these devices are taking over the RTU functionality.

2.2.2 SCADA

SCADA refers to a system or a combination of systems that collects data from various sensors at a plant or in other remote locations and then sends these data to a central computer system, which then manages and controls the data and remotely controls devices in the field. RTU (remote terminal units) are used to collect analog and status telemetry data from field devices, as well as communicate control commands to the field devices [1].

The RTUs are usually installed at the power plants, transmission and distribution substations, distribution feeder equipment, etc.

2.2.3 Remote Terminal Unit (RTU)

The RTU is a microprocessor-based device that interfaces with a SCADA system by transmitting telemetry data to the master station and changing the state of connected devices based on control messages received from the master station or (in some modern systems) commands generated by the RTU itself.

The RTU provides data to the master station and enables the master station to issue controls to the field equipment. Typical RTUs have physical hardware inputs to interface with field equipment and one or more communication ports.

Different RTUs process data in different ways, but in general there are several internal software modules that are common among most RTUs:

- Central RTDB that interfaces with all other software modules.
- Physical I/O application—acquires data from the RTU hardware components that interface with physical I/O.
- Data collection application (DCA)—acquires data from the devices with data communications capabilities via communication port(s). For example, IEDs.
- Data processing application (DPA)—presents data to the master station or HMI.
- Some RTUs also have data translation applications (DTA) that manipulate data before they are presented to the master station or support stand-alone functionality at the RTU level.

2.2.4 Intelligent Electronic Devices (IEDs)

IEDs are microprocessor-based devices with the capability to exchange data and control signals with another device over a communications link. They perform protection, monitoring, control, and data acquisition functions in generating stations, substations, and along feeders.

IEDs are a key component of substation integration and automation technology like integrating protection, control, and data acquisition functions and they can help

utilities improve reliability, gain operational efficiencies, and enable asset management programs including predictive maintenance, life extensions, and improved planning [1].

2.2.5 Sensors

Original copper-wired analog devices can now be replaced by optical fiber-based sensors for monitoring and metering. The most advantages of these type sensors are higher accuracy, reduced size, higher performance, wide dynamic range high bandwidth, and low maintenance.

Their main functionality in the smart system is to collect data from power equipment at substation yard such as transformers, circuit breaker and power lines [1].

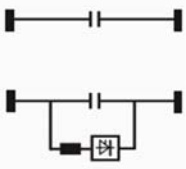
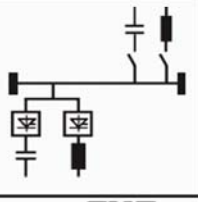
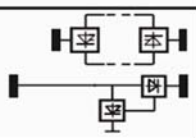
Chapter 3: Transmission Systems in Smart Grid

The Role of transmission systems in smart grid is to integrate renewable energy resources to a greater extent in the future and to increase the efficiency of conventional power generation as well as power transmission and distribution without loss of system security. In order to sustain generation, transmission, distribution, and utilization in balance, the grids must be controlled using power electronics technologies such as HVDC and FACTS which provide a wide range of applications with different solutions. The FACTS and HVDC technologies were developed to improve the performance of weak AC systems and to make long-distance distribution feasible. FACTS and HVDC applications will result in efficient, low-loss AC/DC hybrid grids and will ensure better controllability of the power flow through dynamic fast control which makes the grid able to take in more regenerative and distributed energy source very efficiently [1].

3.1 FACTS AND HVDC: OVERVIEW OF FUNCTIONS

Table 2 summarizes the impact of FACTS and HVDC on load flow, stability, and voltage quality when using different devices [1].

Table 2 - FACTS and HDVC: Overview of Functions [1].

Principle	Devices	Scheme	Impact on System Performance		
			Load Flow	Stability	Voltage Quality
Variation of the line impedance Series compensation	FSC (fixed series compensation) TPSC (thyristor protected series compensation) TCSC (thyristor controlled series compensation)		● ● ●●	●●● ●●● ●●●	● ● ●
Voltage control Shunt compensation	MSC/R (mechanically switched capacitor/ reactor) SVC (static VAr compensator) STATCOM (static synchronous compensator)		○ ○ ○	● ●● ●●	●● ●●● ●●●
Load-flow control	HVDC – B2B, LDT HVDC VSC UPFC (unified power flow controller)		●●● ●●	●●● ●●●	●●● ●●●

Electrical power transmission systems mainly consist of overhead line (OHL), extruded cable, mass-impregnated (MI) cable, oil-filled (OF) cable, gas insulated line (GIL), superconducting cable and superconducting transmission line systems [20].

The Converter stations are required in order to control and manage the transmission power at different stages in a DC power transmission system. The VSCs stations are based on insulated-gate bipolar transistors (IGBTs), and HVDC classic stations with the use of converter transformers. With the introduction of HVDC stations using line-commutated converters (Fig. 4), or voltage source converters (Fig. 5) electrical power can be transmitted over long distances at different power levels [20].

3.2 VOLTAGE SOURCE CONVERTER (VSC)

IGBTs VSC uses high-frequency (up to 2,000 Hz) pulse-width modulation, and thus the use of small filters and independent control of active and reactive power are possible. The VSC transmission system can transmit active power in two directions, with the same control setup and the same main circuit configuration. Hence, the active power transfer can be quickly reversed without any change of control mode, and without any filter switching or converter blocking. The power reversal is obtained by changing the direction of the DC current and not by changing the polarity of the DC voltage as with HVDC classic [20].

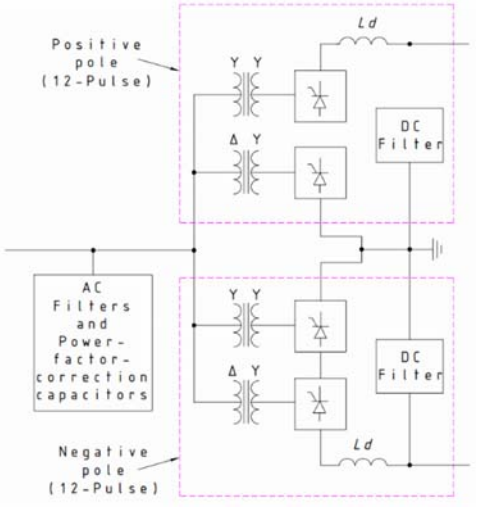


Figure 5 - HVDC classic station [7].

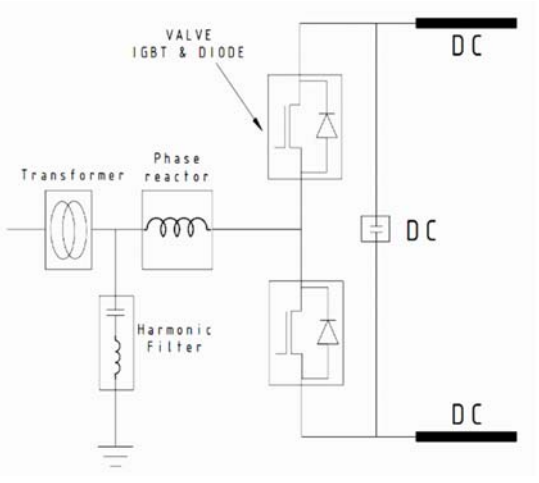


Figure 4 - Voltage Source Converter [20].

3.3 UTILIZING INTERNET OF THINGS (IOT) TECHNOLOGY IN SG

Utilizing Internet of Things (IoT) technology in smart grid is an important approach integrating advanced sensing and communication technologies that can effectively avoid or reduce the damage of natural disasters to the transmission lines, improving the reliability of power transmission and reducing economic loss. In order for the IoT to sense and identify the physical world, it employs a variety of sensing devices such as Radio Frequency Identification (RFID) devices, infrared sensors, Global Positioning System (GPS), and laser scanners [18].

The system network topology can be cluster-chain type, where each sensor can communicate to the nearby backbone node directly, and the communication links between the sensors and the backbone nodes are generally unidirectional. Each backbone node can communicate to maximum 256 sensors. The distance between the backbone nodes is several hundred meters, and the communication link between the backbone nodes is bidirectional. Parts of the backbone nodes are able to access to the public network through 3G, TD-LTE or power optical network. As shown in figure 6 [18].

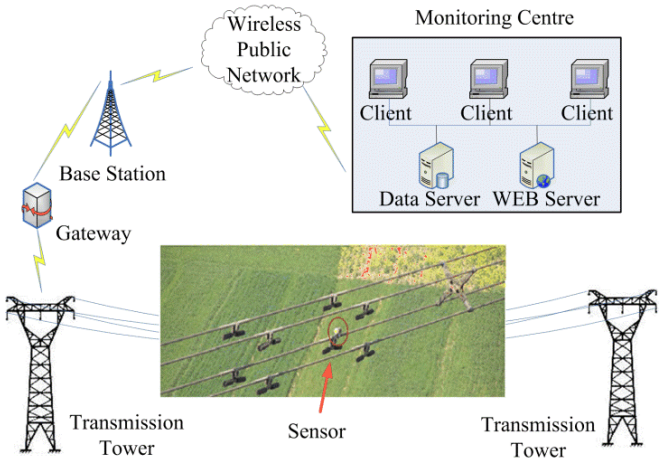


Figure 6 - Data transmission network [18].

3.3.1 Monitoring Contents

The specific monitoring covers are transmission tower leaning, conductor galloping, wind deviation, micro-meteorology, conductor icing, wind vibration, and conductor temperature.

a. Transmission tower leaning:

The leaning sensor transmits the status of the transmission tower to the nearby backbone node, which merges the data from several leaning sensors together to form the information of transmission tower leaning and realize real-time monitoring and early warning.

b. Conductor Galloping:

The number of vertical and horizontal half-waves of galloped conductor are analyzed and the motion track is calculated. Thus, the conductor is in galloping danger. Hence, the discharge between phase conductors and tower collapse can be avoided.

c. Wind deviation:

The wind deviation data results from the wind velocity sensor and the acceleration sensor can provide the field test data for the wind deviation verification of the conductor.

d. Micro-meteorology:

The temperature, humidity, wind velocity, sunshine, and rainfall can be recorded by the wireless sensors along the conductor or on the tower.

e. Conductor icing:

Conductor icing can be determined according to the result of the micro-meteorology sensors and the tension sensors. Thus, the ice flashover can be eased or avoided.

f. Wind vibration:

The vibration frequency and amplitude can be recorded and analyzed. The wind velocity, wind direction, environment temperature, humidity, and the weariness life of the conductor can also be analyzed.

g. Conductor temperature:

The operating temperature of conductor can be collected by the wireless temperature sensors along the conductor.

The devices include temperature, humidity, wind, and other meteorological sensor, vibration sensor, ultrasonic sensor, tower leaning sensor, infrared sensor, leakage current sensor, camera and the backbone node, which build the monitoring system for power transmission line and tower. The devices deployments are shown in figure 8 and figure 9 and the sensor deployment scheme of the power transmission line and tower are shown in figure 7, 8, & 9 [18].

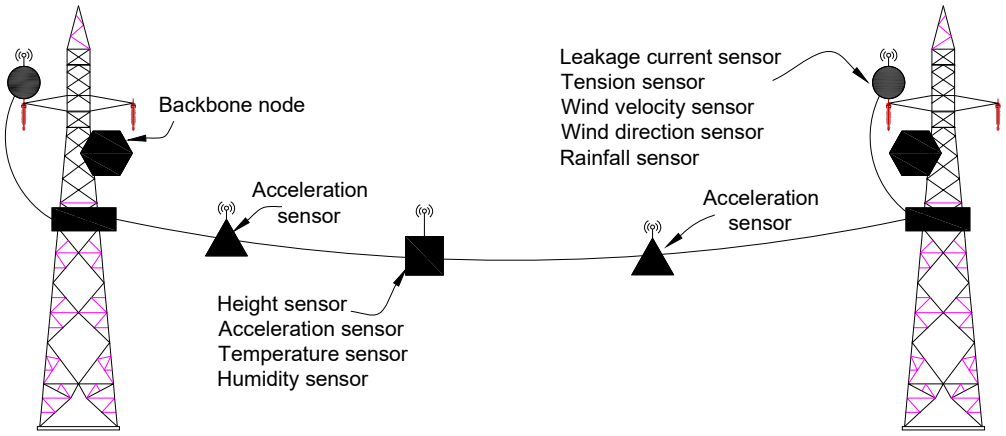


Figure 7 - The sensor deployment scheme of the power transmission line tower [18].

Figure 9 - Wireless sensors [18].

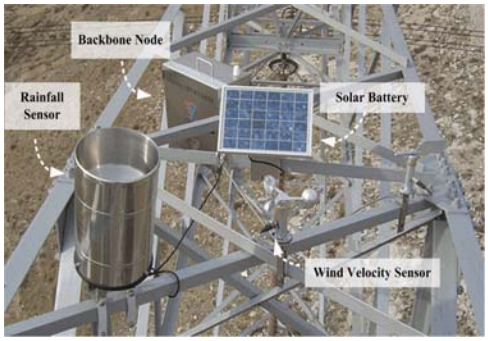


Figure 8 - Deployment of IoT devices [18].

Chapter 4: Communication Systems

The Smart grid (SG) architecture presenting power systems, power flow and information flow has three types of networks as illustrated in figure 2 [4]:

4.1 WIDE AREA NETWORK (WAN)

Wide Area Network (WAN) for automation, distribution and for covering long-haul distances by providing communication links between the NANs and the utility systems to transfer information. The primary functions of WAN are:

- The core network provides communication to utility control centers with high data rate though fiber optics or cellular communications.
- The back –haul networks provide broadband connections and monitoring devices to NANs.

4.2 NEIGHBORHOOD AREA NETWORK (NAN)

Neighborhood Area Network (NAN) for connecting multiple HANs to local access points. The primary functions of NAN are:

- Collect consumers' energy consumption data from smart meter.
- Transmits the collected data to the utility companies through WAN.
- It is comprised of two gateways, NAN and HAN gateways.
- NAN gateway serves as an access point to provide single hop connection to HAN gateways in a hybrid access manner.
- The HAN gateway transmit their energy consumption data to NAN.

4.3 HOME AREA NETWORK (HAN)

Home area network (HAN) extends communication to end points within the end-user home or business. The primary functions of HAN are:

- The commissioning function that identifies and manages new devices.

- The control function enables communication among smart devices by establishing the links and performs the operation for different layers in the smart grid.
- HAN is comprised of a variety of smart devices, such as:
  - Home gateway which connected to smart meters and periodically collects power consumption data of the home appliances.
  - Smart meters.
  - Sensors.
  - Actuator nodes.
  - Smart home appliances.
  - Electric vehicles [4].

4.4 COMMUNICATION TECHNOLOGIES AND STANDARDS

Communication technologies can be classified into two categories:

A. Wired technologies

B. Wireless technologies.

Traditionally, wired technologies are considered beneficial to wireless technologies in terms of reliability, security and bandwidth because cables are easier to protect from interference. In addition, the wired based equipment's are generally cheaper as well as the maintenance cost compared to wireless solutions.

On the other hand, wireless networks apply low installation costs with minimal cabling, which provide network connectivity over wide areas even in areas where they lack pre-existing communication infrastructure.

4.4.1 Wired technologies

a. Power line communications (PLC)

- PLC technologies utilize existing power cables for information exchange. Thus, making the PLC system cost-effective and straightforward solution to grid communication needs.

- However, the data transmissions faces a number of technical challenges due to the signal propagation characteristics of typical power cables, such as high signal attenuation, disruptive interference from other power signals, including nearby electric appliances or external electromagnetic sources.

b. Optical communications

- This communication technology has the ability to transmit data packets over distances in the order of several kilometers.
- The robustness against electromagnetic and radio interference, making it suitable for high-voltage environments. A special type of optical cables, called Optical Power Ground Wire (OPGW) which combines the functions of grounding and optical communications, allowing long-distance transmissions at high data rates.

c. Digital subscriber lines (DSL)

It enables digital data transmissions over telephone lines. The main advantage of DSL technologies is that electric utilities can interconnect residential users to control centers avoiding the additional cost of deploying their own communication infrastructure [3].

4.4.2 Wireless technologies

a. ***802.15.4-based networks***. The IEEE 802.15.4 technology is the reference standard that specifies the physical and MAC layers for low-rate, low-power and low-cost wireless personal area networks (LR-WPANs).

b. ***Wi-Fi (IEEE 802.11-based networks).*** The set of wireless communication technologies mostly used for home and local area networking (WLANs).

c. ***WiMAX (IEEE 802.16-based networks).*** It was firstly released in 2001 to support long-distance (up to 7–10 km) broadband (up to 100 Mbps) wireless communications, especially in rural and suburban areas.

d. ***3G/4G cellular networks***. One of the main advantages of public cellular networks over other wireless communication technologies is the larger coverage area.

However, the drawback of cellular data services is that they are relatively expensive.

Finally, tables 3 and 4 summarize the main features of wired and wireless communication technologies used in the smart grid solutions [3].

Table 3 - Summary of wired communication technologies for SG [3].

Family	Data Rate / Coverage	Applicability	SG Application Areas	Advantages	Disadvantages
Power Line Carrier (PLC)	**2 – 3 Mbps / 1 - 3Km**	✓ HAN ✓ NAN	• Low voltage distribution; • Automatic meter reading	• Cost effective • Low installation cos; • Wide availability • Utility's own ownership and control • Dedicated network	• Noisy and harsh medium • Sensitive to disturbances • Signal quality gets affected by the type and number of devices, wiring distance between nodes and network topology • Cost of ownership • Complexity of management
Digital Subscriber Lines (DSL)	**1 – 100 Mbps / 5 - 28Km**	✓ WAN ✓ NAN ✓ HAN	**Smart Metering**	• High speed • Low latency • Low installation cost high data rate, high capacity • Long range • Wide availability	• Quality is distance dependent • High installation cost in low density (rural) areas • Unreliable
Optical Communications	**Up to 100 Tbps / 10 – 60 Km**	✓ WAN ✓ NAN	**Physical network infrastructure**	• Long distance communication • Ultra-high bandwidth • Robustness against radio and electromagnetic interference • High reliability	• High deployment cost of fiber installation • High cost of terminal equipment • Difficult to upgrade • Not suitable for metering applications

Table 4 - Summary of wireless communication technologies for SG [3].

Family	Data Rate / Coverage	Applicability	SG Application Areas	Advantages	Disadvantages
WPAN	256 Kbps / 10 – 75 m	✓ V2G ✓ HAN ✓ AMI	• Smart Metering • Home automation	• Very low power consumption • Cheap Equipment • Suitable for devices with low memory and computing power	• Low bandwidth • Do not scale to large networks
Wi-Fi	Up to 600 Mbps / Up to 1 Km	✓ V2G ✓ HAN ✓ AMI	• Energy Monitoring • Home automation • Automatic meter Metering	• Low-Cost network deployments (unlicensed spectrum) • Cheap equipment • High flexibility, suitable for different use cases	• High interference since it operated in a very crowded unlicensed spectrum • Power consumption might be too high for many smart grid devices
WiMAX	100 Mpbs for mobile users, 1 Gbps for fixed users / 0-5 km (optimum) 5-30 Km (acceptable) 30-100 Km (reduced performance)	✓ WAN ✓ HAN ✓ AMI	• Real time pricing • Automatic meter reading • Outage detection and restoration	• Suitable for thousands of simultaneous users • Longer distances that Wi-Fi • A connection-oriented control of the channel bandwidth	• Network management is complex • High cost of terminal equipment • Use of licensed spectrum
3G / 4G Cellular Networks	1 Gbps down and 500 Mbps up / 0-5 km (optimum) 5-30 Km (acceptable) 30-100 Km (reduced performance)	✓ WAN ✓ NAN ✓ HAN	• Monitoring and management of Distribution Energy resources • SCADA	• Able to support tens of millions of devices • Low power consumption of terminal equipment • Cellular operators are launching smart grid specific service solutions • Use of licensed spectrum reduces interference	• Cellular operators cab charge utilities high prices to use their I networks • Use of licensed spectrum increase cost • Difficult to ensure delay

Chapter 5: Control Strategies of Hybrid Renewable Energy Microgrid

The combination of RES, such as PV arrays or wind turbines, and battery storage, are generally classified as hybrid energy systems (HES) and generally recognized as a viable alternative to conventional remote area power supplies (RAPS).

For many applications, the combination of renewable and conventional energy sources employ two or more different sources of energy, they contribute a very high degree of reliability as compared to single-source systems [17].

In order to balance the production and demand of electrical energy within the combination network of the existing power grid and RES, the control strategies employed via the use of technologies and processes and advanced control protocols have to be implemented. These controls are needed to enhance the reliability of energy supply of the intermittent nature of the renewable sources (fluctuating sunshine and wind profile). The controls include strategies for the optimal power harnessing and effective energy management, power/voltage device control, intelligent control of energy transformation, and line faults management. A microgrid can connect and disconnect from the grid to enable it to operate in either grid-connected or stand-alone mode using the microgrid central switch. The disconnection from the grid can be as a result of faults events, voltage collapse. All RESs within the microgrid must be well regulated to present peer-to-peer and plug-and-play characteristic [19].

The HRESs must have the ability to mitigate the power quality issues to supply high-quality and more reliable steady power. The power quality and system stability can be achieved by an appropriate control technique embedded into the power converter control circuit [6].

5.1 INTERFACING CONFIGURATION OF HRES SOURCES

The HRES microgrid can be interfaced with the AC bus line and afterwards to the utility grid directly or via a common DC bus by using the appropriate power converters. The AC bus-linked HRES configuration reduces the number of power conversion stages and losses in power transferred to the load/utility [6].

Physically, in HRES configuration, a group of PV panels are interfaced through a DC–DC converter to regulate their fluctuating DC output. The wind turbine coupled with a permanent magnet synchronous generator (PMSG) generates a three-phase AC voltage, and its amplitude and frequency vary with rotor speed. Therefore, the wind turbine generator is connected to the DC bus via a rectifier and DC–DC converter. The storage battery is connected to the DC bus through a bidirectional DC–DC converter to maintain a stable supply–demand balance at its rated capacity. The common DC bus collects the regulated power from various RESs to supply and maintain a constant DC voltage at the input terminal of the DC–AC inverter. A single DC–AC inverter is used to interface the common DC bus to the AC bus connected to the utility grid [6].

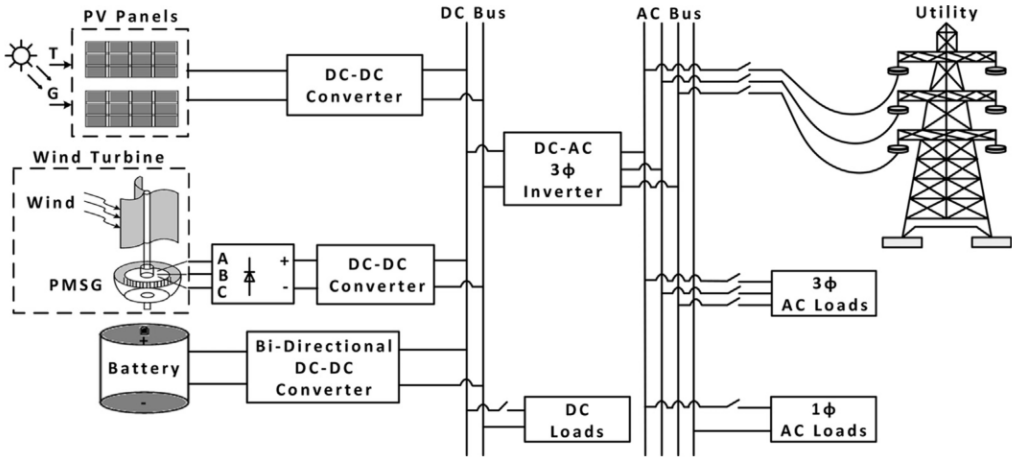


Figure 10 - Typical configuration of the DC and AC bus linked HRES [6].

The advantages of the configuration depicted in figure 10 are:

- The power converters require a simple control algorithm since the source side converters are only dedicated to harvest the maximum power from the RESs by means of Maximum Power Point Tracking (MPPT).
- The converters are responsible to control the matching of the real power output among the RESs and regulate the DC bus voltage.
- The output of the RESs can be flexibly regulated (either buck or boost) by the DC–DC converter to meet the requirement of the DC–AC inverter input.

5.2 CONTROL CONCEPTS

The HRES sources must be properly controlled by specific power converter control schemes to perform voltage and power regulation in standalone and grid-connected mode. Figure 11 illustrates the control concept schemes of the standalone and grid-connected systems.

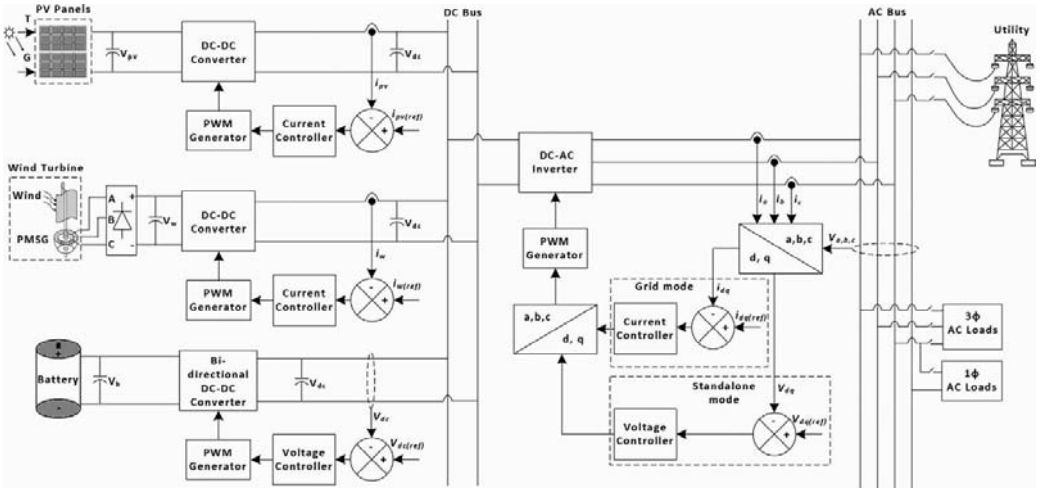


Figure 11 - Control concept of the DC and AC bus linked HRES [6].

In the Stand-alone mode, one of the sources producing stable output power must be functioned in voltage-controlled mode to regulate the DC bus voltage, and the remaining sources should be operated in current-controlled mode to control the coordination of power among the RESs [6]. The inverter on the AC side must be operated in voltage-controlled mode to keep the voltage and frequency of the AC bus constant. Therefore, the voltage and frequency (*Vf*) control scheme must be implemented in the inverter control circuit to perform the voltage and frequency regulation.

However, in the Grid-Connected mode, the inverter must be operated in the current control mode to regulate the active and reactive power injected into the grid and to supply the AC loads (i.e.: the inverter control circuit must be adopted with an active and reactive power (PQ) control scheme) [6].

5.3 POWER CONVERTER TOPOLOGIES AND CONTROL STRATEGIES FOR A SINGLE INVERTER INTERFACED HRES

The overall power converter control scheme in standalone mode is shown in Figure 12, whereas the inverter control scheme in grid-connected mode is shown in

Figure 12 (a). In this HRES configuration, the maximum power point tracking (MPPT) feature is also realized to all sources that are connected in parallel to the common DC. The storage battery is directly connected to the DC bus to smooth the power fluctuation and the stability of the DC bus voltage. A supervisory energy management with a real-time control system monitors and controls the power supply–demand balance to ensure the stable operation of the entire HRES [6].

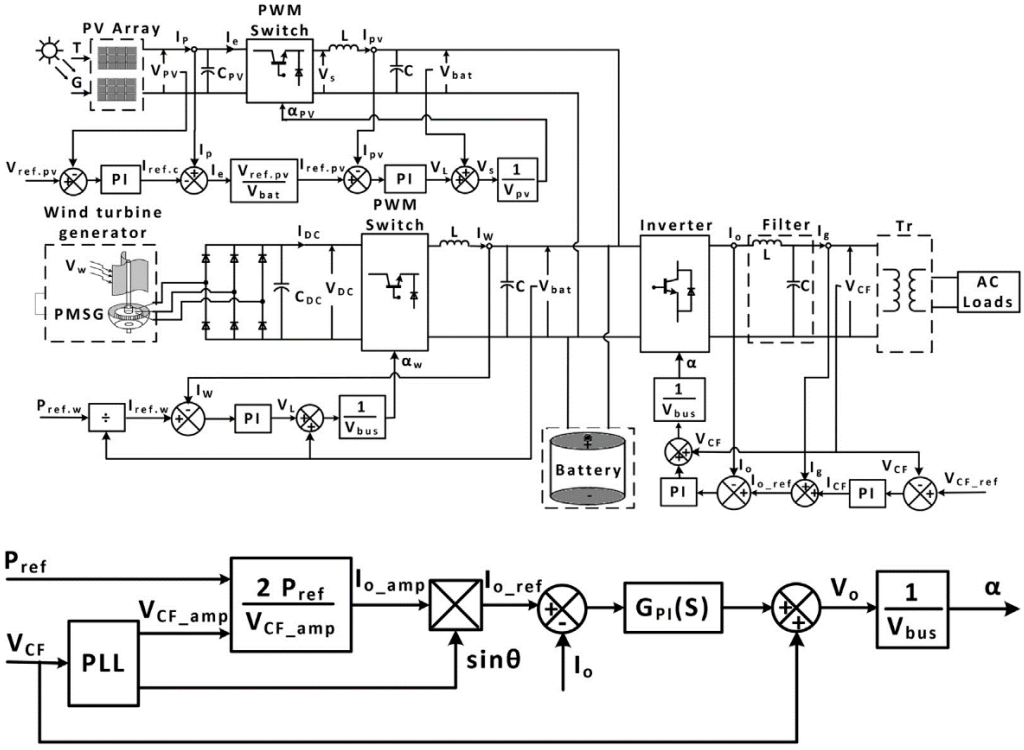


Figure 12 - Overall power converter control scheme for the standalone,
(a). Inverter control strategy for the grid-connected mode [6].

5.4 FUTURE APPLICATION

The trend in installing a standalone or grid-connected HRES is important in areas which don't have access to the power grid or areas which have power grid. However, further research and development for HRESs microgrid is required for the future power grid mix to address the following challenges:

- Minimize the power conversion losses.
- Enhance the performance of the HRES advanced control strategies to supply quality power.

- The existing power grids based on conventional power generation systems are constructed without considering the integration of HRESs. Thus, the standalone HRES must rely on storage batteries to:
  - Store excess power.
  - Mitigate several technical issues, such as voltage fluctuations, voltage flicker and fluctuating power loads.

Chapter 6: RES Microgrid Sizing

Renewable energy currently accounted for 19% of global final energy demand in 2015, having risen by 0.17% per year since 2010 and contribute to the majority of the greenhouse gas emissions reduction that is needed between now and 2050 for limiting average global surface temperature increase below 2 °C.

Table 5 represents the required growth of renewable energy technologies between 2015 and 2050 for energy transition [15].

Table 5 – Breakdown of renewable energy growth, 2017-2050

Key renewable energy technologies	Units	Recent growth	Future growth requirement
		2017(e)	2018–50
Wind	GW/yr	53	154
Solar PV	GW/yr	99	210
Modern biomass (end-use)			
Liquid biofuels and biogas	billion liters/ yr	5	24
Solar thermal	million m^2	30	283
Hydro	GW/yr	25	17
Hydrogen			
Geothermal heat	PJ/yr	26	173
CSP	GW/yr	1	20
Others (incl. marine and hybrid)	GW/yr	0.1	28
Geothermal (power)	GW/yr	0.6	7
Bioenergy (power)	GW/yr	5	12

The power system considered here is composed of three parts: WT, PV and battery bank. The two former units generate electricity, in accordance with the local wind and solar energy resources, to supply the load. The battery bank forms the energy storage system that can supply the load when there is shortage of electricity, and store the surplus power when the power generated exceeds the load. The energy storage system is essential to cover the shortage of the renewable energy's unpredictable and fluctuant nature, but its existence brings difficulties to the sizing problem [5].

6.1 ESTIMATING THE OUTPUT POWER OF WIND TURBINE

The speed of wind is often represented by a random variable. The Weibull distribution function with two parameters is commonly used to describe wind speed data. It provides a convenient representation of the wind speed data for wind energy calculation purposes. The general representation of the Weibull distribution is given by [1]:

$$f(V_{wind}) = (k/c)(V_{wind}/c)^{k-1}\exp\left(-(V_{wind}/c)^{k}\right), \tag{1}$$

$$k = (\sigma_w/V_{mean})^{-1.086} \quad and \quad c = \frac{V_{mean}}{\Gamma(1+1/k)}$$

Where the Γ() is the gamma function, V_{mean} is the average value of wind speed data, and σ_w is the standard deviation of wind speed data [1].

The available wind generator power Pout can be expressed by a function of V_{wind}:

$$P_{out}(V_{wind}) = \begin{cases} P_{rated} \cdot \frac{\left(V_{wind}^{k} - V_{in}^{k}\right)}{\left(V_{in}^{k} - V_{rated}^{k}\right)} & \text{if } V_{in} \le V_{wind} \le V_{rated}, \\ P_{rated} & \text{if } V_{rated} \le V_{wind} \le V_{out}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Where,

- P_{rated} is rated power of the wind turbine
- V_{in} is the cut-in speed
- V_{rated} is the rated wind speed
- V_{out} is the cut-out wind speed
- k is the shape parameter

Finally, the Average output power of the wind turbine can expressed using [5]:

$$P_{wind} = \int_{0}^{\infty} P_{out}(V_{wind}) \cdot f(V_{wind}) \cdot d(V_{wind}) \tag{3}$$

6.2 ESTIMATING THE OUTPUT POWER OF PHOTVOLTAIC (PV) ARRAY

When there are different level of temperatures and irradiation effects on the PV module, the I-V characteristic of PV module could be different. The power generated by the PV array is not only dependent on the solar irradiation but also dependent on the ambient temperature of the site. The maximum output power at different radiation and temperature is calculated by the following equation [5]:

$$P_S(G, \Delta T) = k_1 \cdot A_S \cdot G \cdot \left(1 - k_T \Delta T\right) \tag{4}$$

Where,

- ΔT is the temperature error of the PV module between the cell temperature T_c and the reference cell T_{cref} in degrees Celsius
- k_t is the temperature coefficient
- k_1 is the PV module generation efficiency
- A_s is the total area of the PV Module in m^2

Because of cloud cover and other insolation reducing phenomena, the solar irradiance G can also be represented by a random variable It has been shown that the beta distribution with parameters α and β can describe the solar irradiance G [5].

$$f_G(G) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \left(\frac{G}{G_{max}}\right)^{\alpha - 1} \left(1 - \frac{G}{G_{max}}\right)^{\beta - 1} \tag{5}$$

$$\alpha = \mu_s \left[\frac{\mu_s(1 - \mu_s)}{\sigma_s^2} - 1\right] \quad and \quad \beta = (1 - \mu_s)\left[\frac{\mu_s(1 - \mu_s)}{\sigma_s^2} - 1\right. \tag{6}$$

Where,

- G_{max} is the maximum solar irradiance during the time
- μ is the mean value
- σ^2 is the variance

Finally, the Average output power of the PV array can expressed using:

$$P_S = \int_0^{G_{max}} P_S(G, \Delta T) \cdot f_G(G) \cdot d(G) \tag{7}$$

6.3 PERFORMANCE OF THE BATTERY

Energy storage is necessary because the generation of renewable energy is inherently intermittent. The stored energy can supply the load when there is a lack of electricity, and store surplus power when the generated power exceeds the load. The battery behavior is mainly characterized by the State of Charge (SOC). The SOC at the moment t can be obtained using the following equation [1]:

$$SOC(t) = SOC(t-1) + \frac{N_{wind} \cdot P_{wind}(t) + N_{PV} \cdot P_S(t) - P_L(t)}{V_b C_b} \quad (8)$$

Where,

- PL(t) is the electric power demand at moment t,
- V_b is the battery voltage
- C_b is the capacity of battery bank
- N_{wind} denote the number of WT
- N_{pv} is the number of PV

6.4 RES MICROGRID SIZING ALGORITHM

The flowchart diagram of the proposed algorithm for getting feasible configurations is shown below and described in the following sequence:

- ✓ The input data for this program consists of: the hourly wind speed and solar radiation data.
- ✓ The daily load profiles [P(t); t = 1; 2;……; 24].
- ✓ The specifications of wind turbine, PV array and battery.
- ✓ The initialization includes the calculation of power generated by both the given wind turbine and the PV array, which is based on the collection of the wind speed and solar radiation data at the same hour every day to get the average daily curves.
- ✓ N_{max} is the maximum number of wind turbines allowed in the wind turbine alone system to meet load demand.

Firstly, all the configurations of the system (N_{max}, N_{pv}, and C_b) satisfying the demand can be calculated by the algorithm. Then, these solutions were applied to equations referred in the system cost analysis section to obtain the corresponding H_{system}. Finally, the optimal solution that has the smallest H_{system} can be easily obtained [5].

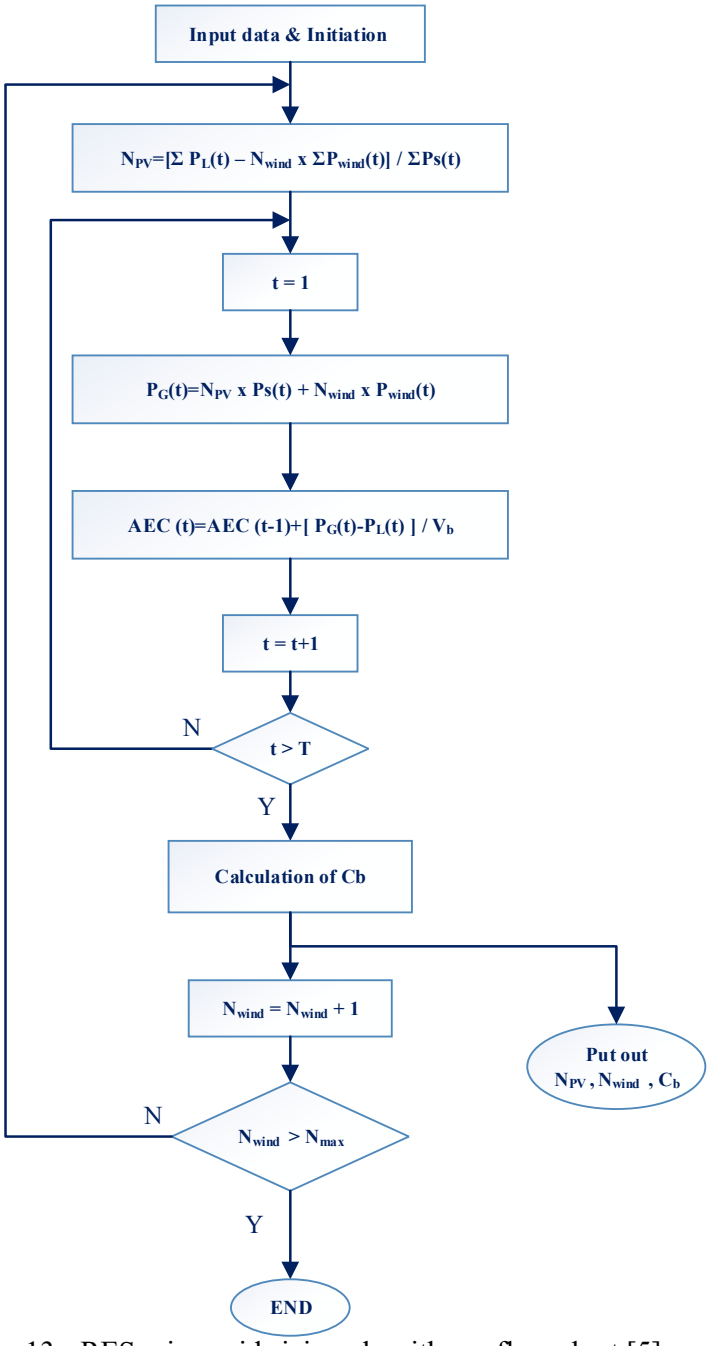


Figure 13 - RES microgrid sizing algorithm – flow chart [5].

6.5 CUSTOM NEURAL NETWORK – SIZING THE WIND TURBINE NUMBERS

In this section, the maximum number of wind (N_{wmax}) related to all seasons is not manually set but instead is calculated according to trained network as described below in sequence:

1. **Define the input and corresponding target output vectors**
   - Inputs = [x:y]: input vector.
   - Outputs = [w:z]: corresponding target output vector.
2. **Create Custom Neural Network**
   - Net=network(numInputs,numLayers,biasConnect,inputConnect,layerConnect,outputConnect).
     - numInputs: number of inputs.
     - numLayers: number of layers.
     - biasConnect: numLayers-by-1 Boolean vector.
     - inputConnect: numLayers-by-numInputs Boolean matrix.
     - layerConnect: numLayers-by-numLayers Boolean matrix.
     - outputConnect: 1-by-numLayers Boolean vector.
3. **Define the number of hidden layer neurons**
   - net layers {1} size = 5.
4. **Define the layer transfer function**
   - net layers{1} transferFcn = 'logsig'.
     - ‘logsig’ is a transfer function Transfer functions calculate a layer's output from its net input.
5. **Define the network training**
   - net trainFcn = 'trainlm'.
     - ‘trainlm’ is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization (a

popular alternative to the Gauss-Neuton method of finding the minimum of a function that is a sum of squares of nonlinear functions).

- ‘trainlm’ is often the fastest backpropagation algorithm in the toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms.

- net performFcn = 'mse'.
  - ‘mse’ is a network performance function. It measures the network's performance according to the mean of squared errors.
- net = train(net,inputs,outputs).
  - Train shallow neural network.

6. **Set the network response after training**
   - Output = net(inputs).
7. **View the customized neural network**
   - View (net).

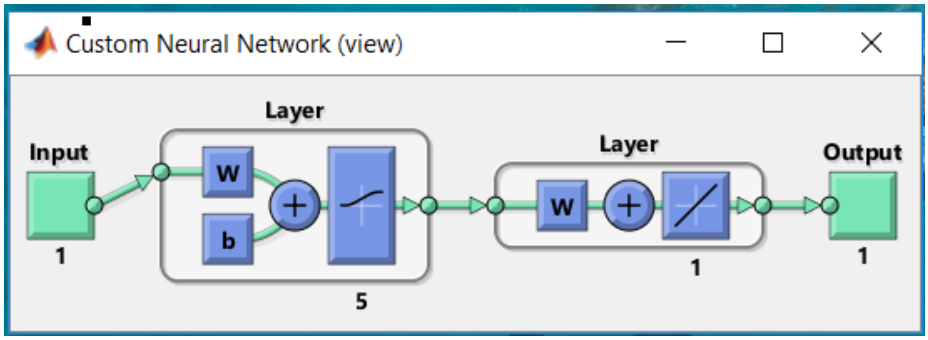


Figure 14 - Custom Neural network net – wind turbine number forecast

6.6 ANALYSIS OF THE MICROGRID SYSTEM COST

The optimal sizing of hybrid microgrid is given in the sense that the system life cycle cost is minimized while the given load power demand can be satisfied without load rejection [5].

It is essential that the microgrid power system to be assembled under the optimum configuration considering both investment and system power reliability. The analysis of cost is necessary to utilize the renewable energy resources efficiently and economically. The system life cycle cost is expressed by the following equations:

$$
\begin{aligned}
H_{system} &= H_{cap} + H_{ope} + H_{rep} \\
H_{cap} &= N_{wind} \cdot C_w + N_{PV} \cdot C_{PV} + C_b \cdot C_{bat} \\
H_{ope} &= C_{ope} \cdot T_L \\
H_{rep} &= C_b \cdot C_{bat} \cdot \sum_{n=1}^{T_{bat}} \frac{1}{(1+i)^{T_L \times n}} + N_{wind} \cdot C_w \cdot \sum_{n=1}^{T_w} \frac{1}{(1+i)^{T_L \times n}}
\end{aligned}
\tag{9}
$$

Where,

- N_{wind}, N_{PV} and C_b are the number of wind turbines, PV arrays and batteries combined in the hybrid system respectively.
- C_w is the initial capital cost of wind turbines.
- C_{PV} is the initial capital cost of PV arrays.
- C_{bat} is the initial capital cost of batteries.
- $C_{ope} = k_c * H_{cap}$.
- K_c is specified percentage of the initial capital.
- T_L is the life time of the system and often equals to the life time of the PV array as it has longer service life than the other two components.
- T_{bat} is the service time of the battery.
- T_w is the service time of the wind turbine.
- i = (e d)=(1 + d) is the actual interest rate, e is the nominal rate and d is the escalation rate.

Finally, if the specifications of wind turbine, PV array and battery are given, the cost can be calculated by the unit sizing in terms of the function H_{system} (N_{wind}, N_{PV}, C_b).

Chapter 7: The Artificial Intelligence Techniques for Wind Speed Forecasting

The intermittency of wind speed and solar irradiance is very challenging for energy production via wind turbine and PV arrays to synchronize with the power system. The crucial role of the artificial neural network (ANN) technologies lies in the accuracy of forecasting wind and solar energies for a better effective power system management. Given that wind and solar energies are highly variable in different locations and seasons, smart wind speed forecasting is a challenging task, and if done properly gives an accurate prediction of the surplus green power produced by the wind turbines and PV arrays.

7.1 THE ARTIFICAL INTELLIGENCE (AI) TECHNIQUES: OVERVIEW

The Artificial intelligence (AI) techniques, such as expert systems (ESs), fuzzy logic (FL), and artificial neural networks (ANNs) have enhanced power electronics and power engineering providing powerful tools for design, simulation, control, estimation, fault diagnostics, and fault-tolerant control in modern smart grid (SG) and renewable energy systems (RESs) [16].

The basic building block of ANNS is a simple processing unit called neuron organized in interconnected layers. The computational capabilities are determined by the connection weights, network architecture and training algorithm [8].

With the recent developments in Neural Network (NN) and Deep Learning (DL) new approaches based on deeper architectures such n-layers Perceptron (MLP), Convolution Neural Networks (CNN) and Recurrent Neural Network (RNN) are being classified as the three main classes in optimal forecasting methods that extract and learn the shape of the wind and solar patterns. In this architecture, each layer of the network has only forward connections with the subsequent layer. Depending on the role and position of a layer, input, hidden and output layer types are the main elements of a Feed-Forward Network (FNN). An input or data fed to the input layers of MLP is treated by the hidden layers and the result is delivered at the output [21].

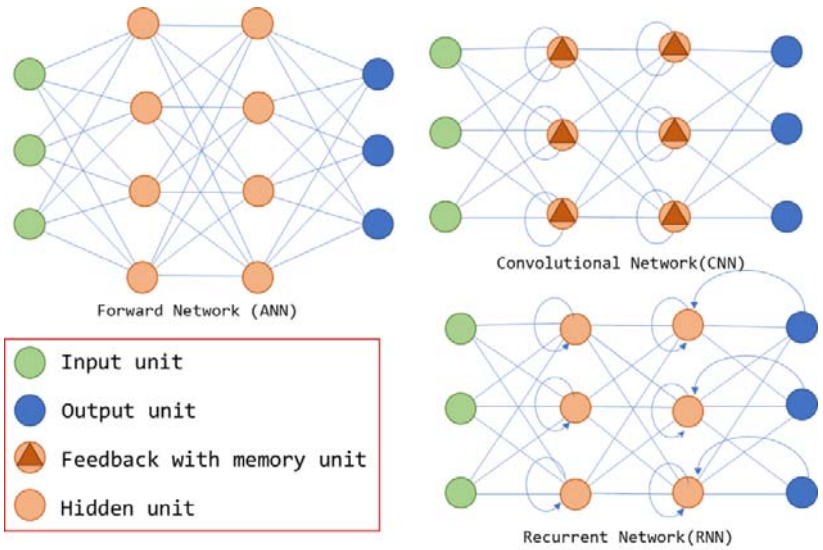


Figure 15 - Topology of the different Neural Network architectures [21].

The goal of an MLP network is to approximate some function f*, when there are multiple layers, each layer is a function of the function:

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{11}$$

In this three-layer example each function is a layer in the network (one, two, and three). This number defines the depth of the model, being the last layer of the output layer.

Neural Networks have an activation function on each neuron that acts on the inputs received and generates an output, plus a backpropagation algorithm that optimizes the weights on each connection in a process to find the optimal combination for the output. Neural networks are non-linear which allows them to produce better results than linear models on wind and solar data time series [21].

RNN are designed to process sequential data by, most importantly, sharing parameters between the different layers and neurons, generating cycles in the graph sequence of the network. In this sense, RNN can have extensive memory. In an RNN each output is a function of the previous elements. Thus, the values in a specific step will influence its value in future steps. RNN networks have the potential to learn from patterns in the time series to predict the future [21].

Nevertheless, generalization capability of the forecast model can be improved to cover any site different from the one trained upon by considering the following issues:

a. Data preprocessing used as input fed to the model such as time-series, NN or fuzzy system.

b. Design and training of forecast models must be easy-to-operate and applicable to all sites.

c. Selection of number of input data is crucial because unnecessary input data will rather slow down the process.

d. Selection of number of outputs depends on whether a single forecast or multiple forecasts for a particular look-ahead time is needed.

e. Implementation and validation of forecast models is very important for NNs, because once they are trained on training set, they should provide good forecasts for out-of-sample test data too and for this, some performance evaluation error criteria (like MAE, MSE, RMSE, etc..) are used.

f. Comparative evaluation based on their time-scale classification under three categories:

   - Very short-term forecasting: From few seconds to 30 minutes ahead.
   - Short-term forecasting: From 30 minutes to 6 hours ahead.
   - Medium-term forecasting: From 6 hours to 1 day ahead.
   - Long-term forecasting: From 1 day to 1 week ahead [14].

7.2 CONVOLUTIONAL NEURAL NETWORKS (CNN)

The effectiveness of using Convolutional Neural Networks (CNN) for level energy load forecasting was tested and results obtained from the CNN were compared to those obtained by ANN, Support Vector Machines (SVM), Factored Restricted Boltzmann Machines (FCRBM), and other deep architectures Long Short Term Memories (LSTM). Experimental results showed that the CNN outperformed SVR while having comparable results to the ANN and deep learning methodologies [13].

CNNs has the advantage of processing big data and addressing it in a form of a two dimensional matrix, widely applied in the field of image processing or time series.

CNN weight sharing network structure which is similar to a biological NN can reduce the complexity of the network model and the number of weights [10].

Convolution networks extract relevant features from small areas of the matrix, by identifying short intervals of the time series that could bring relevant information to the prediction task. The information could be that some patterns are relevant for the future behavior of wind and solar series [21].

The structure of CNN is a feed-forward neural network. It uses the back-propagation algorithm to optimize the network structure to solve unknown parameters in the network. It starts with the pre-processing of the samples and the required features are extracted and classified or determined by regression analysis in order to obtain the output. [10]

The pre-processing is implemented using convolution filters that act repeatedly on the receptive field and extract the local features of the input matrix with the convolution kernel of the matrix. Further, the pooling process calculates the average or maximum in each section of the convolved feature. After pooling, the dimension of the characteristics statistics is greatly decreased and the generalization ability of the model is increased. Thus, the basic structure of CNN consists of convolution layer and pooling layer. The neurons in the convolution layer are locally connected with the previous layer, and their local features are extracted which their local sensitivity and extracted features are processed again by the pooling layer [10].

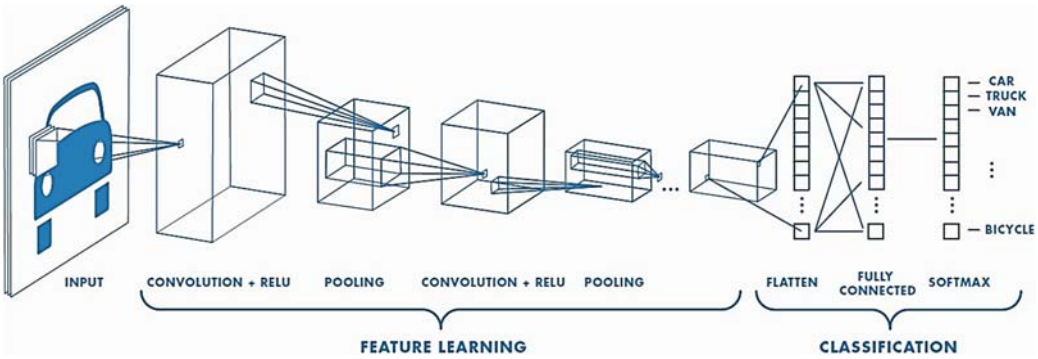


Figure 16 - CNN structure

In 1D convolution neural network, the convolution layers read the input multi-time series signal using a kernel filter that reads in small segments at a time and steps across the entire segment input field. The pooling layer, then, takes the feature map projections and filter them to the most essential elements such as using a signal averaging or signal maximizing process. The output of the convolution and pooling layers network is one or more fully connected (FC) layers that interpret the internal representation and output a vector representing the multi-step time predicted [23].

The development of the multi-step time series forecasting model with CNN starts with defining the prior hours/days subsequence data as input that enables the model to read and learn to extract features from.

The 1D CNN model data consists of [samples, timesteps, features] shape, each sample consists of specified number of time steps with one feature for the defined number of time steps. The recorded dataset is divided into training and testing data, so the shape of the training or testing dataset is [training/testing, timesteps, features]. The first step is to flatten the data to transform the 2D into 1D problem and then iterate over the time steps and each iteration moves along one time steps and predicts the subsequent hours/days [23].

Furthermore, this multi-step series forecasting problem is an auto regression time series model which uses observations from previous time steps as input to an equation to predict the value at the next time step which can result in accurate forecasts on a range of time series problems.

The built model consists of one convolution layer with different number of filters progressively incremented and a kernel size of 1. This means that the input sequence will be read with a convolutional operation one time step at a time and this operation will be performed according to the number of filters following by a pooling layer which will reduce these feature maps their size before the internal representation is flattened to one long vector.

Finally, the outcome is then interpreted by a fully connected layer before the output layer predicts the next hours/days in the sequence [23].

Table 6 - The parameter setting of the proposed model

Parameter	***Setting***
1D convolution filter number	32
1D convolution kernel size	1
1D convolution filter number	64
1D convolution kernel size	1
1D convolution filter number	96
1D convolution kernel size	1
1D convolution filter number	112
1D convolution kernel size	1
1D convolution filter number	128
1D convolution kernel size	1
1D convolution filter number	144
1D convolution kernel size	1
Dense layer activation function	ReLU
Optimizer	Adaptive Moment Estimation (Adam)
Verbose	0
Epochs	20
Batch size	4
Loss Function	MSE

The Mean Squared Error (MSE) loss function is chosen to be the error metric of Root-Mean-Square-Error (RMSE) and Mean Absolute Error (MAE) is used as evaluation indicator.

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}_n| \tag{12}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{N}}$$

In order to improve the training performance of the deep learning model, the adaptive moment estimation (Adam) was applied, which is an extension of stochastic gradient descent (DSG), to adjust parameters. Adam can iteratively update weights of neural networks based on training data [22].

The Rectified Linear Unit (ReLU) activation function was selected to limit output value range to [0,1]

$$\mathrm{ReLU}(x) = \max(0, x) \tag{13}$$

The small batch size and the stochastic nature of the algorithm means that the same model will learn a slightly different mapping of inputs to outputs each time it is trained. This means results may vary when the model is evaluated.

Table 7 - Algorithm of the proposed code

Algorithm	***The algorithm of the Proposed code***
1:	Load and prepare dataset
2:	Fill in missing values
3:	Split the dataset into train and test sets
4:	Convert history into inputs and outputs
5:	Build the CNN model
6:	Set verbose=0; epochs=20; batch size =4
7:	Train the model on batch
8:	Make a forecast
9:	Evaluate the model through the Walk-forward validation
10:	Plot the actual vs. predicted
11:	Print the MAE, RMSE values of error indicators
12:	Terminate

In this context, the Convolution Neural Network (CNN) can symbolize a practical and reliable tool to precisely monitor and predict the wind speed and solar irradiance outputs and accordingly manage the power transfer switching between areas that have surplus renewable energies to areas that have shortage in energies by initiating the Energy Management System (EMS) to dispatch power according to schedule.

7.3 POWER GENRATION, TRANSMISSION, AND DISTRIBUTION – OVERALL SINGLE LINE DIAGRAM

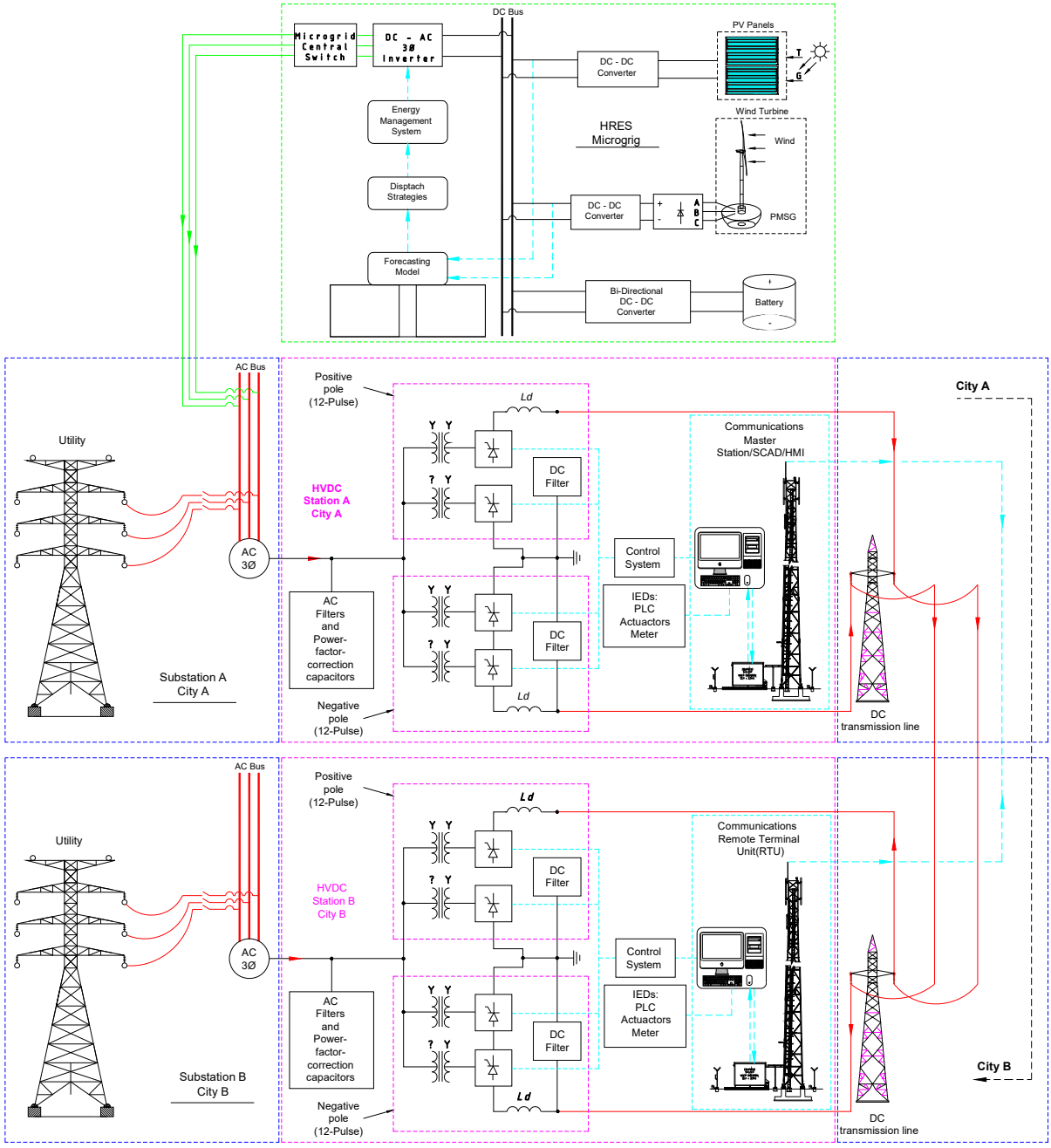


Figure 17 - Power generation, transmission, and distribution – overall single line diagram

Chapter 8: The Case Study: Lebanon, Beirut

Based on historical weather data of wind speed and solar irradiance related to Lebanon, Beirut City [24] [25], the 3 days forecast results related to wind speed and solar irradiance are depicted in figures 18 and 19.

Table 8 - Lebanon, Beirut - Solar irradiance and wind speed max/min/avg. data

Location	Solar irradiance ($kw/m^2/day$)			Wind Speed (m/s)		
Lebanon, Beirut	**MAX**	**MIN**	**AVG.**	**MAX**	**MIN**	**AVG.**
Latitude	8.48	0.43	5.088	25	0.56	3.54
33^0 53’ 19.0680” N						
Longitude						
35^0 29’ 4.7280” E						
Elevation						
100 m						

The 1D convolution neural network model built according to table 7, is trained using real pre-processing data extracted from the historical weather websites. The data extracted covers a period of 28270 hours related to wind speed [24] and 1160 days related to solar irradiance [25], from December 2016 to April 2019. The dataset covering 28200 hours were used to train the model to predict the wind speed the next 70 hours (~3 days). In addition, the dataset covering the 1157 days were used to train the model to predict the solar irradiance the next 3 days.

The comparison of the wind speed 3 days forecast results with persistence models based on the MAE and RMSE indicators are tabulated in table 8, demonstrating the effectiveness of the proposed model that provides the most forecasting accuracy.

8.1 RESULTS

8.1.1 Comparison of wind speed 3 days forecasting results in terms of MAE

Table 9 - Comparison of wind speed 3 days forecast results in terms of MAE

Test	MLP	WindNet	Proposed Model	
3 Days Forecast	**MAE**	**MAE**	**MAE**	**RMSE**
# 1	0.951965	0.906002	0.672649	0.843
# 2	0.749479	0.726946	0.687472	0.881
# 3	0.883512	0.919904	0.697633	0.9
# 4	0.725489	0.735706	0.691511	0.874
# 5	1.00744	0.956887	0.706766	0.902
# 6	0.845219	0.743931	0.676663	0.871
# 7	0.877877	0.867812	0.705473	0.888
# 8	0.832765	0.769551	0.684255	0.879
# 9	0.800525	0.744625	0.686370	0.875
# 10	0.742297	0.644439	0.750552	0.956
# 11	0.751773	0.786698	0.668264	0.859
Average	**0.833486**	**0.800227**	**0.693418**	**0.884**

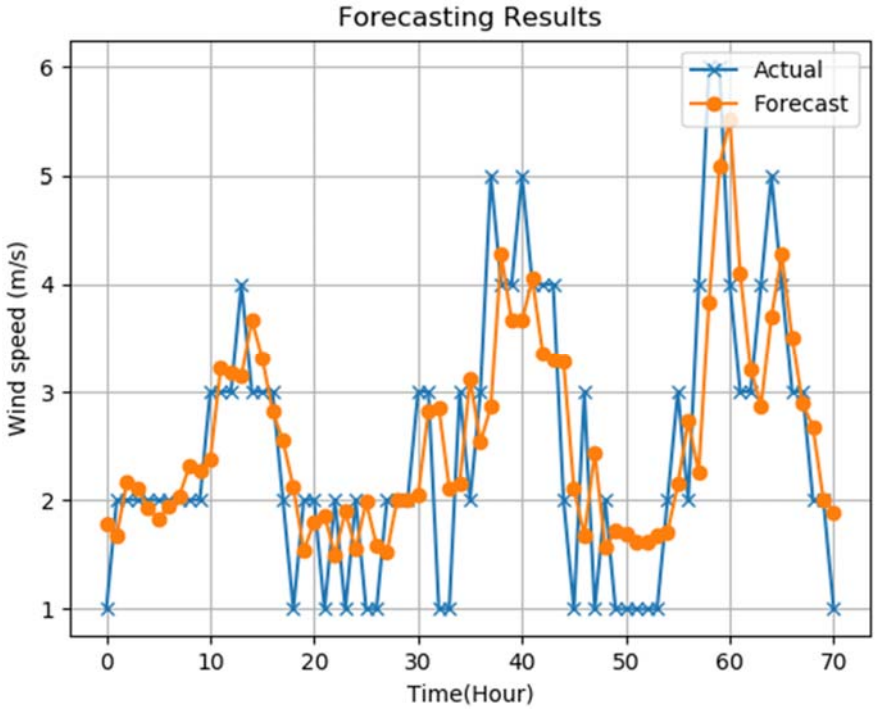


Figure 18 - Wind speed 3 days actual vs. forecast

8.1.2 Solar irradiance 3 days forecasting results in terms of MAE

Table 10 - Solar irradiance 3 days forecast results

Test	Proposed Model (Solar irradiance)	
3 Days Forecast	**MAE**	**RMSE**
# 1	0.516827	0.583
# 2	0.372195	0.393
# 3	0.481563	0.563
# 4	0.443098	0.479
# 5	0.352257	0.390
# 6	0.465694	0.500
# 7	0.579304	0.688
# 8	0.450965	0.489
# 9	0.299923	0.338
Average	**0.440203**	**0.491**

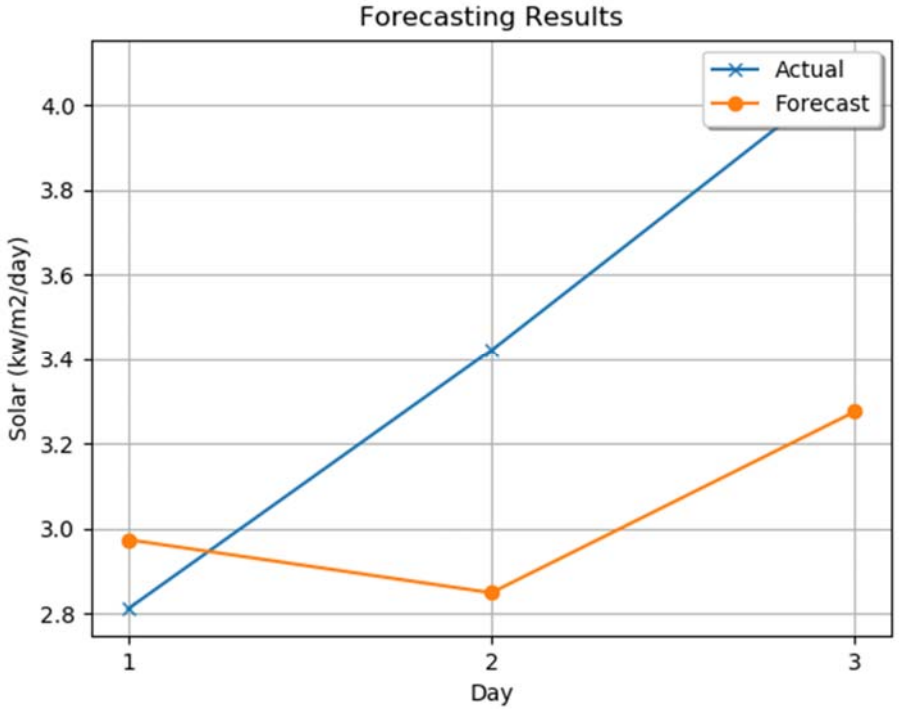


Figure 19 - Solar irradiance 3 days actual vs. forecast

In addition, based on historical weather data of wind speed and solar irradiance related to Lebanon, Beirut City [24] [25], the HRES microgird is sized based on the demand load and the output PV/Wind turbine energy sources calculated providing that the battery's state of charge (SOC) is maintained constant at 50 % as shown in figure 26 during all seasons. The results are depicted in figures 20, 21, 22, 23, 24, and 25.

8.1.3 Power demand load and available energy capacity within the battery related to Beirut City in each season

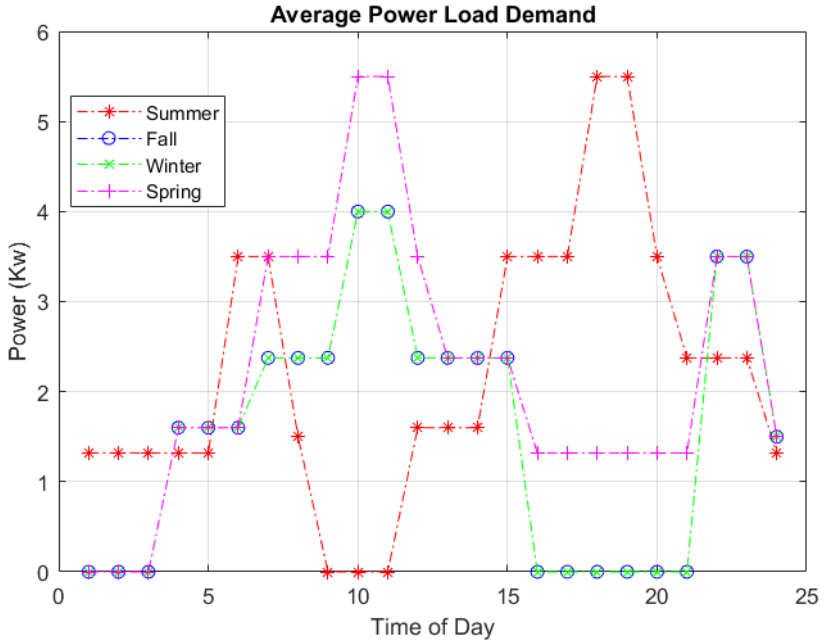


Figure 20 - Average power load demand

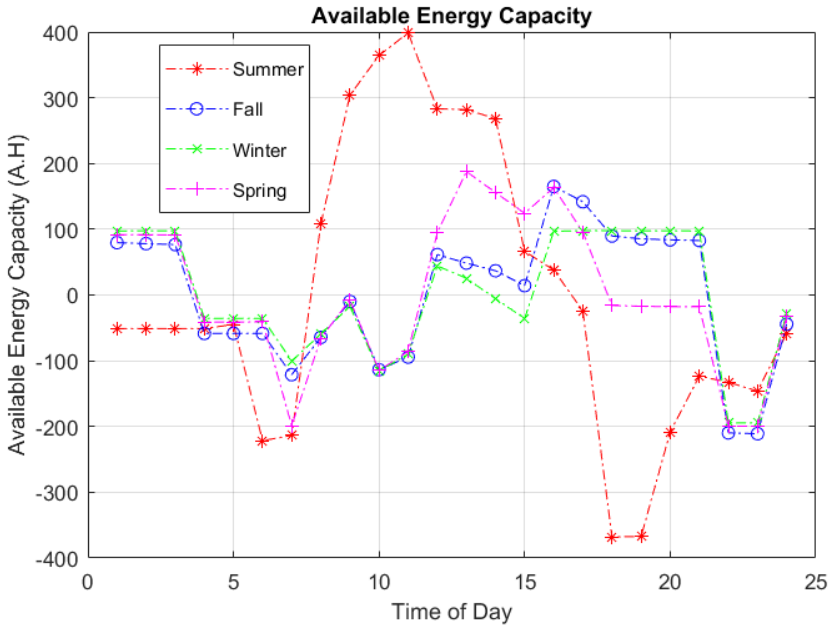


Figure 21 - Available Energy capacity in the battery

8.1.4 Average output power distribution of wind and solar related to Beirut City in each season

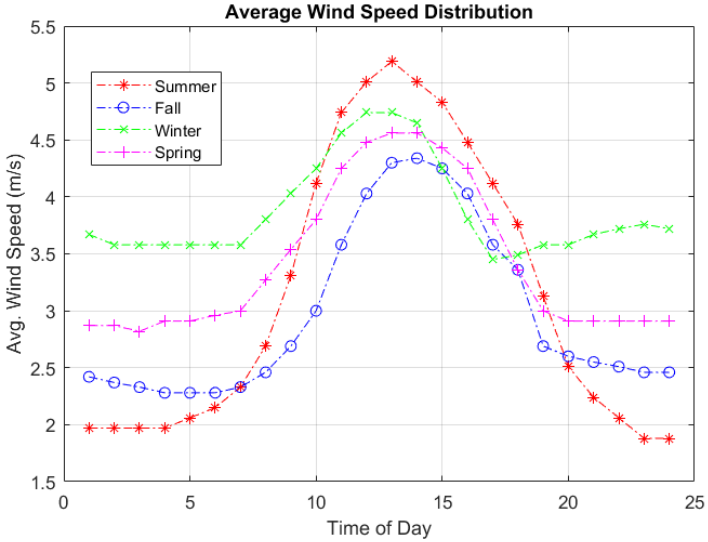


Figure 22 - Average wind speed distribution

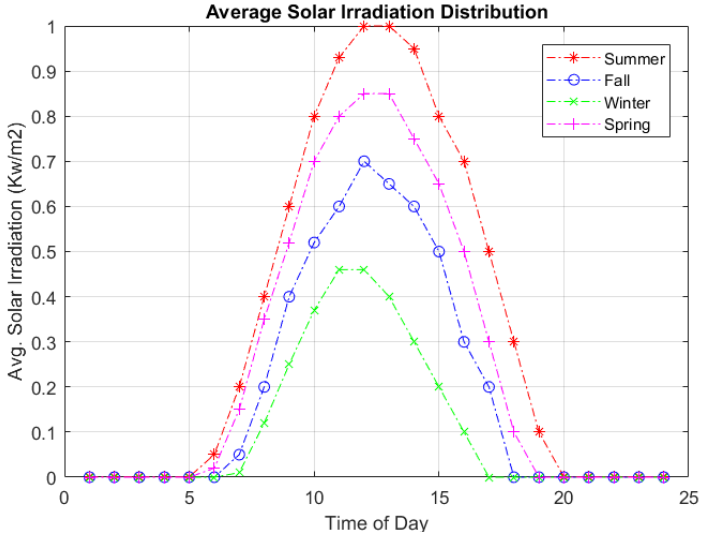


Figure 23 - Average solar irradiance distribution

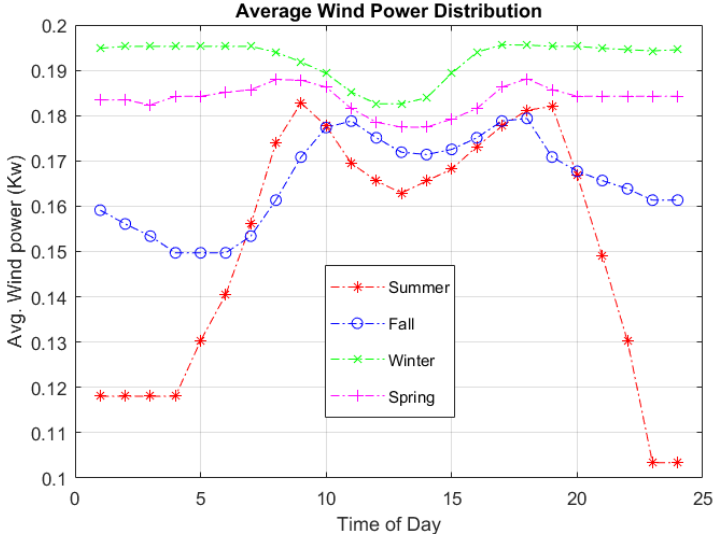


Figure 24 - Average wind power distribution

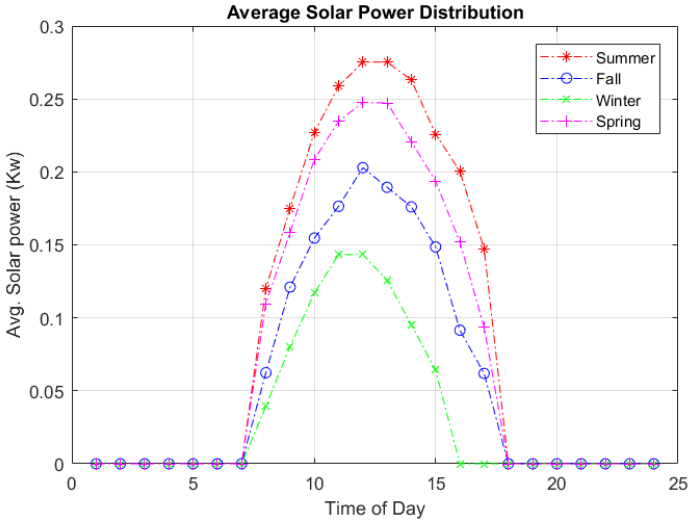


Figure 25 - Average solar power distribution

8.1.5 Battery's SOC cycle invariance

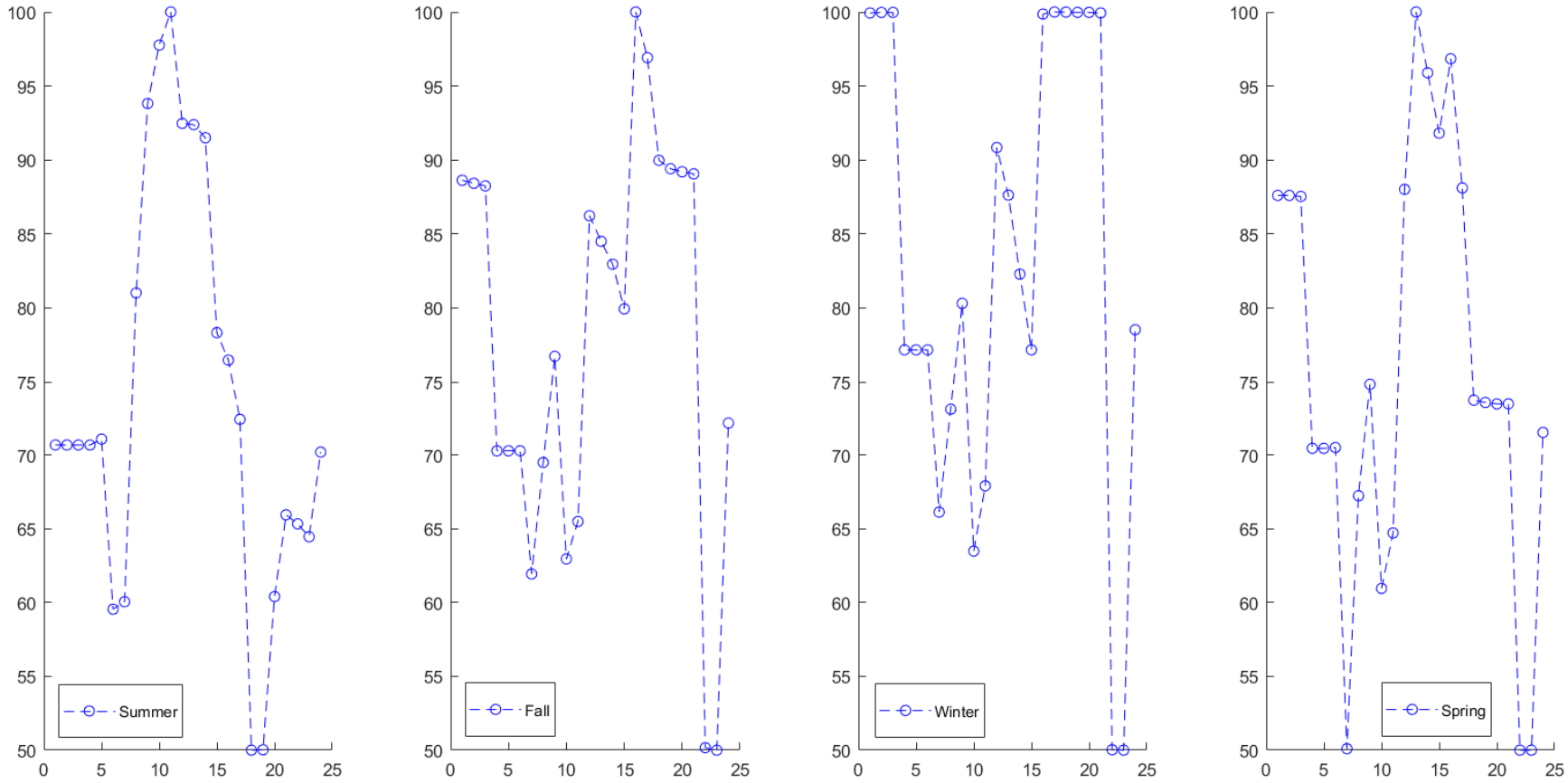


Figure 26 - Battery state of charge during all seasons

8.1.6 System Life Cycle Cost ($) with respect to Battery Capacity bank (KAH) related to Beirut City in each season – Ex: C_w=$ 500, C_{bat}=$ 135, C_{pv}=$ 235

Season	N_{wind}	N_{PV}	# of Batteries	Battery Bank (A H)	System Cost ($)
Summer	**1**	**12**	**19**	**1903.67**	**20738.12**
	2	11	18	1829.47	21467.10
	3	10	18	1755.26	22738.28
	4	9	17	1681.06	23467.26
	5	8	16	1606.86	24196.24
	6	8	15	1532.66	25630.21
Fall	**1**	**13**	**10**	**1018.80**	**18411.51**
	2	11	9	924.051	18774.46
	3	10	9	876.65	20521.83
	4	8	8	835.16	20884.80
	5	7	8	793.67	22632.16
	6	5	8	752.18	23674.53

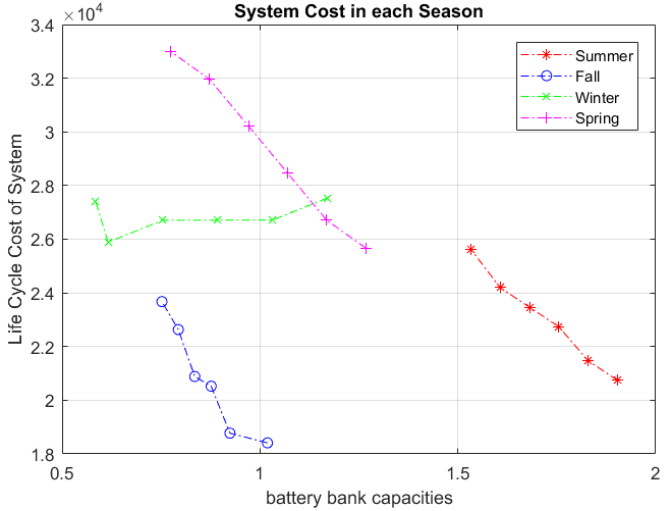


Figure 27 - System life cycle cost

Season	N_{wind}	N_{PV}	# of Batteries	Battery Bank (A H)	System Cost ($)
Winter	1	21	12	1170.02	27533.0
	2	18	10	1031.54	26713.32
	3	15	9	893.05	26710.25
	4	12	8	754.56	26707.19
	5	**9**	**6**	**616.07**	**25887.50**
	6	7	6	584.68	27406.05
Spring	**1**	**14**	**13**	**12667.29**	**25674.50**
	2	12	12	11682.83	26715.42
	3	11	11	10698.37	28461.33
	4	10	10	971.40	30207.24
	5	9	9	872.94	31953.15
	6	7	8	774.50	32994.06

Microgrid optimal Size	The optimal configuration of system with lower life cycle cost				
	N_{wind}	N_{PV}	# of Batteries	Battery Bank (A H)	System Cost ($)
	5	**9**	**6**	**616.07**	**25887.50**

Chapter 9: Conclusion

Smart Grid (SG) optimizes the management of energies and real-time balance between electrical energy supply and demand sides. Thus, the reliability of power dispatching process highly depends on the energy management system and the energy sources availability.

The most important features in developing the smart grid are the integration of renewable energy sources, such as solar and wind, to increase the electricity consumption and to guarantee the sustainability of the power system.

However the intermittency of wind speed and solar irradiance are very challenging to power production from wind turbines and photovoltaic arrays respectively. In this context, the Convolution Neural Network (CNN) can symbolize a practical and reliable tool to precisely monitor and predict the wind speed and solar irradiance outputs and accordingly manage the power transfer switching between areas that have surplus renewable energies to areas that have shortage in energies by initiating the Energy Management System (EMS) to dispatch power according to schedule.

In addition, the effectiveness of the proposed model CNN was tested using meteorological data related to Beirut city, Lebanon. The experimental data were divided into training and testing datasets. The large amount of training dataset has proven the reliability of the model when tested on actual data recorded in the past.

The experimental results indicate that the MAE and RMSE values are respectively 0.693418 and 0.884 related to wind speed, and respectively 0.440203 and 0.491 related to solar irradiance, consequently demonstrating very high forecasting accuracy.

References

1. S. Borlase, “Smart Grids Infrastructure, Technology, and Solution”, © 2013 by Taylor & Francis Group, LLC, CRC Press is an imprint of Taylor & Francis Group, an Informa business, International Standard Book Number-13: 978-1-4398-2910-3
2. E. Ancillotti, R. Bruno, M. Conti, “The role of communication systems in smart grids: Architectures, technical solutions and research challenges”, Elsevier B.V. All rights reserved, Energy Procedia 83, (2015), 428 – 433.
3. M. Hossain, N. Madlool, N. Rahim, J. Selvaraj, A.K. Pandey, Abdul Faheem Khan, “Role of smart grid in renewable energy: An overview”, Elsevier B.V. All rights reserved, Computer Communications 36 (2013) 1665–1697.
4. Y. Saleem, N. Crespi, M. Rehmani, and R. Copeland, “Internet of Things-aided Smart Grid: Technologies, Architectures, Applications, Prototypes, and Future Research Directions”; IEEE ,1704.08977©2019
5. J. Li, W. Wei, and J. Xiang, “A Simple Sizing Algorithm for Stand-Alone PV/Wind/Battery Hybrid Microgrids”, Energies 2012, 5, 5307-5323; doi:10.3390/en5125307
6. P. Arul, V. Ramachandaramurthy, R. Rajkumar, “Control strategies for a hybrid renewable energy system: A review”; Elsevier Ltd. All rights reserved, Renewable and Sustainable Energy Reviews 42 (2015), 597–608
7. N. Mohan, “Power Electronics: Converters, Applications, and Design”, Third edition, Copyright © 2003 John Wiley & Sons, Inc. All rights reserved.
8. H. Masrur, M. Nimol, M. Faisal, G. Mostafa, “Short Term Wind Speed Forecasting Using Artificial Neural Network: A Case Study”; IEEE ,78-1-5090-6122-8/16/©2016
9. C. Dumitru, A. Gligor, “Daily Average wind Energy Forecasting using Artifical Neural Networks”, Elsevier B.V. All rights reserved, Procedia Engineering, 181(2017), 829 – 836
10. A. Zhu, X. Li, Z. Mo, H. Wu, “Wind Power Prediction Based on a Convolutional Neural Network”; IEEE , 978-1-5386-1871-4/17/ ©2016
11. A. R. Finamore, V. Galdi, V. Calderaro, A. Piccolo, G. Conio, S. Grasso, “Artificial Neural Network Application in Wind Forecasting: an One-Hour-Ahead Wind Speed Prediction”; DIIn - University of Salerno, Fisciano (SA), Italy, I.V.P.C. SERVICE S.R.L., Avellino, Italy.

12. W. He, “Load Forecasting via Deep Neural Networks”; Elsevier B.V. All rights reserved, Procedia Engineering, 122 (2017), 308 – 314

13. K. Amarasinghe, D. Marino, Milos Manic “Deep Neural Networks for Energy Load Forecasting”; IEEE , 978-1-5090-1412-5/17/ © 2017

14. S. Soman, H. Zareipour, P. Mandal, “A Review of Wind Power and Wind Speed Forecasting Methods With Different Time Horizons”; IEEE, 10.1109/NAPS.2010.5619586, © 2010

15. D. Gielen, F. Boshell, D. Saygin, M. Bazilian, N. Wagner, R. Gorini, “The role of renewable energy in the global energy transformation”; Elsevier Inc. All rights reserved, 24(2019)38-50

16. B. K. Bose, “Artificial Intelligence Techniques in Smart Grid and Renewable Energy Systems—Some Example Applications”; IEEE, 0018-9219 © 2017

17. M. Rashid, “Alternative energy in Power ELectronics”, © 2014 by Butterworth-Heinemann, ISBN: 9780124167148

18. A. Egwebe, M. FAzeli, P. Igic, P. Holland “Load sharing methods for inverter-based systems in islanded microgrids: A review”; Electronics and Energetics, Vol. 30, N^0 2, June 2017, pp. 145-160, DOI: 10.2298/FUEE1702145E.

19. R. Liu, “Progress of Long-Distance DC Electrical Power Transmission”; IEEE, 78-1-5090-5736-8/17/ © 2017

20. J. Manero, J. Bejar, U. Cortes “Wind Energy Forecasting with Neural Networks: A Literature Review”; ISSN 2007-9737, doi: 10.13053/CyS-22-4-3081, Vol. 22, No. 4, 2018

21. CJ. Huang, PH. Kuo, “A Short-Term Wind Speed Forecasting Model by Using Artificial Neural Networks with Stochastic Optimization for Renewable Energy Systems”; Energies 2007-9737, doi: 10.3390/en11102777, 2018

22. J. Brownlee, PhD., “Develop convolutional Neural Networks for Multi-step Time Series Forecasting”;https://machinelearningmastery.com/how-to-develop-convolutional-neural-networks-for-multi-step-time-series-forecasting, 2018

23. https://weathersparkcom/y/148700/Average-Weather-at-Beirut-International-Airport-Lebanon-Year-Round; [Weather data extracted source are implemented in Matlab matrix]

24. https://www.timeanddate.com/weather/lebanon/beirut/historic?month=7&year=2017; [Weather data extracted source are implemented in python wind speed forecast script]

25. https://power.larc.nasa.gov/data-access-viewer/; [Solar irradiance data extracted source are implemented in python solar irradiance forecast script]

26. F. Keyrouz, D. Khoury, S. Georges, “A Unified Platform for Simplifying the Design for Small-Scale Solar and Wind Installations in Lebanon”; IEEE, 10.1109/REDEC.2018.8598116 © 2019

Appendices

Appendix A

1D Convolution Neural Network for wind speed time series forecasting using Python

```
#----------------------------1D_Conv_Wind_speed_DK-----------------------------------------------
from numpy import nan
from math import sqrt
from numpy import isnan
from pandas import read_csv
from pandas import to_numeric
from numpy import split
from numpy import array
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot
from keras.models import Sequential
from keras.layers import Bidirectional
from keras.layers import Dense, Embedding, Dropout, LSTM
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
import matplotlib.pyplot as plt

# fill missing values with a value at the same time one day ago
def fill_missing(values):
    # one_day = 60 * 24
        for row in range(values.shape[0]):
            for col in range(values.shape[1]):
                if isnan(values[row, col]):
                    values[row, col] = values[row - 1, col]



# read from csv file and fill missing values
from pandas import read_csv
# load the new file
dataset = read_csv('Python_20190507.csv', header=0, infer_datetime_format=True, parse_dates=['date
time'], index_col=['datetime'])
# mark all missing values
dataset.replace('?', nan, inplace=True)
# make dataset numeric
dataset = dataset.astype('float32')
# fill missing
fill_missing(dataset.values)
print(dataset.shape)
print(dataset.head())
# save
dataset.to_csv('Python_20190507_hours.csv')

# split a univariate dataset into train/test sets
def split_dataset(data):
    # split into standard weeks
    train, test = data[0:-71], data[-71:]
    train = array(split(train, len(train)))
    test = array(split(test, len(test)))

    return train, test
```

```
55)
56) # load the new file
57) dataset = read_csv('Python_20190507_hours.csv', header=0, infer_datetime_format=True, parse_dates=
    ['datetime'], index_col=['datetime'])
58) train, test = split_dataset(dataset.values)
59) # validate train data
60) print(train.shape)
61) print(train[0, 0, 0], train[-1, -1, 0])
62) # validate test
63) print(test.shape)
64) print(test[0, 0, 0], test[-1, -1, 0])
65)
66) #----------------------------CHECK-------------------------------------------------------------
67)
68) # evaluate one or more weekly forecasts against expected values
69) def evaluate_forecasts(actual, predicted):
70)     RMSET = list()
71)     actualvalues= list()
72)     forecastevalues =list()
73)     #maedk = list()
74)     # calculate an RMSE values and RMSE average
75)     for i in range(actual.shape[1]):
76)         # calculate mse and mae
77)         mse = mean_squared_error(actual[:, i], predicted[:, i])
78)         mae = mean_absolute_error(actual[:, i], predicted[:, i])
79)         actualvalues = actual[:,0]
80)         forecastevalues = predicted[:,0]
81)         ape = ((abs(forecastevalues - actualvalues))/actualvalues)*100
82)
83)         # calculate rmse
84)         rmse = sqrt(mse)
85)         # store
86)         RMSET.append(rmse)
87)
88)
89)     # calculate overall RMSE
90)     s = 0
91)     print(actual.shape[1])
92)     for row in range(actual.shape[0]):
93)         for col in range(actual.shape[1]):
94)             s += (actual[row, col] - predicted[row, col])**2
95)     RMSEA = sqrt(s / (actual.shape[0] * actual.shape[1]))
96)
97)     return actualvalues, forecastevalues, RMSEA, RMSET, mae, ape
98)
99)
100)    # summarize RMSET
101)    def summarize_RMSET(name, RMSEA, RMSET):
102)        s_RMSET = ', '.join(['%.1f' % s for s in RMSET])
103)        print('%s: [%.3f] %s' % (name, RMSEA, s_RMSET))
104)
105)    # convert history into inputs and outputs
106)    def to_supervised(train, n_input, n_out=1):
107)        # flatten data
108)        data = train.reshape((train.shape[0]*train.shape[1], train.shape[2]))
109)        X, y = list(), list()
110)        in_start = 0
111)        # step over the entire history one time step at a time
112)        for _ in range(len(data)):
113)            # define the end of the input sequence
114)            in_end = in_start + n_input
115)            out_end = in_end + n_out
116)            # ensure we have enough data for this instance
117)            if out_end < len(data):
118)                x_input = data[in_start:in_end, 0]
119)                x_input = x_input.reshape((len(x_input), 1))
```

```
120)                X.append(x_input)
121)                y.append(data[in_end:out_end, 0])
122)            # move along one time step
123)            in_start += 1
124)
125)        return array(X), array(y)
126)
127)    # train the model
128)    def build_model(train, n_input):
129)        # prepare data
130)        train_x, train_y = to_supervised(train, n_input)
131)        # define parameters
132)        verbose, epochs, batch_size = 0, 20, 4
133)        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.shape[2], train_y.shape[1]
134)
135)        # define model
136)        model = Sequential()
137)        model.add(Conv1D(filters=32, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
138)        model.add(MaxPooling1D(pool_size=1))
139)        model.add(Conv1D(filters=64, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
140)        model.add(MaxPooling1D(pool_size=1))
141)        model.add(Conv1D(filters=96, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
142)        model.add(Conv1D(filters=112, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
143)        model.add(MaxPooling1D(pool_size=1))
144)        model.add(Conv1D(filters=128, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
145)        model.add(MaxPooling1D(pool_size=1))
146)        model.add(Conv1D(filters=128, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
147)        model.add(MaxPooling1D(pool_size=1))
148)        model.add(Conv1D(filters=144, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
149)        model.add(MaxPooling1D(pool_size=1))
150)        model.add(Flatten())
151)        model.add(Dense(10, activation='relu'))
152)        model.add(Dense(n_outputs))
153)        model.compile(loss='mse', optimizer='adam')
154)        # fit network
155)        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, verbose=verbose)
156)
157)        return model
158)
159)    # make a forecast
160)    def forecast(model, history, n_input):
161)        # flatten data
162)        data = array(history)
163)        data = data.reshape((data.shape[0]*data.shape[1], data.shape[2]))
164)        # retrieve last observations for input data
165)        input_x = data[-n_input:, 0]
166)        # reshape into [1, n_input, 1]
167)        input_x = input_x.reshape((1, len(input_x), 1))
168)        # forecast the next week
169)        yhat = model.predict(input_x, verbose=0)
170)        # get the vector forecast
171)        yhat = yhat[0]
172)
173)        return yhat
174)
175)    # evaluate a single model
176)    def evaluate_model(train, test, n_input):
177)        # fit model
```

```
178)        model = build_model(train, n_input)
179)        # history is a list of weekly data
180)        history = [x for x in train]
181)        # walk-forward validation over each week
182)        predictions = list()
183)        for i in range(len(test)):
184)            # predict the week
185)            yhat_sequence = forecast(model, history, n_input)
186)            # store the predictions
187)            predictions.append(yhat_sequence)
188)            # get real observation and add to history for predicting the next week
189)            history.append(test[i, :])
190)            # evaluate predictions days for each week
191)        predictions = array(predictions)
192)        actualvalues, forecastevalues, RMSEA, RMSET, mae, ape = evaluate_forecasts(test[:, :, 0],
    predictions)
193)
194)        return actualvalues, forecastevalues, RMSEA, RMSET, mae, ape
195)
196)    # load the new file
197)    dataset = read_csv('Python_20190507_hours.csv', header=0, infer_datetime_format=True, parse_da
    tes=['datetime'], index_col=['datetime'])
198)    # split into train and test
199)    train, test = split_dataset(dataset.values)
200)    # evaluate model and get RMSET
201)    n_input = 24
202)    actualvalues, forecastevalues, RMSEA, RMSET, mae, ape = evaluate_model(train, test, n_input)
203)
204)    print('APE',ape)
205)    print('MAE',mae)
206)
207)    # summarize RMSET
208)    summarize_RMSET('cnn', RMSEA, RMSET)
209)
210)    plt.figure(1)
211)    plt.title("Forecasting Results")
212)    plt.xlabel("Time(Hour)")
213)    plt.ylabel("Wind speed (m/s)")
214)    plt.plot(actualvalues, marker='x', label='Actual')
215)    plt.plot(forecastevalues, marker='o', label='Forecast')
216)    plt.grid()
217)    plt.legend(loc="upper right", shadow=True)
218)    plt.show()
```

Appendix B

1D Convolution Neural Network for solar irradiance time series forecasting using Python

```
#----------------------------1D_Conv_Solar_Irradiance_DK-----------------------------------
-------------------------
from numpy import nan
from math import sqrt
from numpy import isnan
from pandas import read_csv
from pandas import to_numeric
from numpy import split
from numpy import array
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot
from keras.models import Sequential
from keras.layers import Bidirectional
from keras.layers import Dense, Embedding, Dropout, LSTM
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
import matplotlib.pyplot as plt

# fill missing values with a value at the same time one day ago
def fill_missing(values):
    # one_day = 60 * 24
        for row in range(values.shape[0]):
            for col in range(values.shape[1]):
                if isnan(values[row, col]):
                    values[row, col] = values[row - 1, col]


# read from csv file and fill missing values
from pandas import read_csv
# load the new file
dataset = read_csv('Python_20190508_Solar.csv', header=0, infer_datetime_format=True, parse_da
tes=['datetime'], index_col=['datetime'])
# mark all missing values
dataset.replace('?', nan, inplace=True)
# make dataset numeric
dataset = dataset.astype('float32')
# fill missing
fill_missing(dataset.values)
print(dataset.shape)
print(dataset.head())
# save
dataset.to_csv('Python_20190508_Solar_days.csv')

# split a univariate dataset into train/test sets
def split_dataset(data):
    # split into standard weeks
    train, test = data[0:-3], data[-3:]
    train = array(split(train, len(train)))
    test = array(split(test, len(test)))

    return train, test

# load the new file
dataset = read_csv('Python_20190508_Solar_days.csv', header=0, infer_datetime_format=True, par
se_dates=['datetime'], index_col=['datetime'])
train, test = split_dataset(dataset.values)
```

```
# validate train data
print(train.shape)
print(train[0, 0, 0], train[-1, -1, 0])
# validate test
print(test.shape)
print(test[0, 0, 0], test[-1, -1, 0])

#-----------------------------CHECK-------------------------------------------------------------

# evaluate one or more weekly forecasts against expected values
def evaluate_forecasts(actual, predicted):
    RMSET = list()
    actualvalues= list()
    forecastevalues =list()
    #maedk = list()
    # calculate an RMSE values and RMSE average
    for i in range(actual.shape[1]):
        # calculate mse and mae
        mse = mean_squared_error(actual[:, i], predicted[:, i])
        mae = mean_absolute_error(actual[:, i], predicted[:, i])
        actualvalues = actual[:,0]
        forecastevalues = predicted[:,0]
        ape = ((abs(forecastevalues - actualvalues))/actualvalues)*100

        # calculate rmse
        rmse = sqrt(mse)
        # store
        RMSET.append(rmse)


    # calculate overall RMSE
    s = 0
    print(actual.shape[1])
    for row in range(actual.shape[0]):
        for col in range(actual.shape[1]):
            s += (actual[row, col] - predicted[row, col])**2
    RMSEA = sqrt(s / (actual.shape[0] * actual.shape[1]))

    return actualvalues, forecastevalues, RMSEA, RMSET, mae, ape


# summarize RMSET
def summarize_RMSET(name, RMSEA, RMSET):
          s_RMSET = ', '.join(['%.1f' % s for s in RMSET])
          print('%s: [%.3f] %s' % (name, RMSEA, s_RMSET))

      # convert history into inputs and outputs
      def to_supervised(train, n_input, n_out=1):
          # flatten data
          data = train.reshape((train.shape[0]*train.shape[1], train.shape[2]))
          X, y = list(), list()
          in_start = 0
          # step over the entire history one time step at a time
          for _ in range(len(data)):
              # define the end of the input sequence
              in_end = in_start + n_input
              out_end = in_end + n_out
              # ensure we have enough data for this instance
              if out_end < len(data):
                  x_input = data[in_start:in_end, 0]
                  x_input = x_input.reshape((len(x_input), 1))
                  X.append(x_input)
                  y.append(data[in_end:out_end, 0])
              # move along one time step
              in_start += 1

```

```
            return array(X), array(y)

        # train the model
        def build_model(train, n_input):
            # prepare data
            train_x, train_y = to_supervised(train, n_input)
            # define parameters
            verbose, epochs, batch_size = 0, 20, 4
            n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.shape[2], train_y.sh
    ape[1]

            # define model
            model = Sequential()
            model.add(Conv1D(filters=32, kernel_size=1, activation='relu', input_shape=(n_times
    teps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Conv1D(filters=64, kernel_size=1, activation='relu', input_shape=(n_times
    teps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Conv1D(filters=96, kernel_size=1, activation='relu', input_shape=(n_times
    teps,n_features)))
            model.add(Conv1D(filters=112, kernel_size=1, activation='relu', input_shape=(n_time
    steps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Conv1D(filters=128, kernel_size=1, activation='relu', input_shape=(n_time
    steps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Conv1D(filters=128, kernel_size=1, activation='relu', input_shape=(n_time
    steps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Conv1D(filters=144, kernel_size=1, activation='relu', input_shape=(n_time
    steps,n_features)))
            model.add(MaxPooling1D(pool_size=1))
            model.add(Flatten())
            model.add(Dense(10, activation='relu'))
            model.add(Dense(n_outputs))
            model.compile(loss='mse', optimizer='adam')
            # fit network
            model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, verbose=verbose)

            return model

        # make a forecast
        def forecast(model, history, n_input):
            # flatten data
            data = array(history)
            data = data.reshape((data.shape[0]*data.shape[1], data.shape[2]))
            # retrieve last observations for input data
            input_x = data[-n_input:, 0]
            # reshape into [1, n_input, 1]
            input_x = input_x.reshape((1, len(input_x), 1))
            # forecast the next week
            yhat = model.predict(input_x, verbose=0)
            # get the vector forecast
            yhat = yhat[0]

            return yhat

        # evaluate a single model
        def evaluate_model(train, test, n_input):
            # fit model
            model = build_model(train, n_input)
            # history is a list of weekly data
            history = [x for x in train]
            # walk-forward validation over each week
```

```
            predictions = list()
            for i in range(len(test)):
                # predict the week
                yhat_sequence = forecast(model, history, n_input)
                # store the predictions
                predictions.append(yhat_sequence)
                # get real observation and add to history for predicting the next week
                history.append(test[i, :])
                # evaluate predictions days for each week
            predictions = array(predictions)
            actualvalues, forecastevalues, RMSEA, RMSET, mae, ape = evaluate_forecasts(test[:, :, 0], predictions)

            return actualvalues, forecastevalues, RMSEA, RMSET, mae, ape

        # load the new file
        dataset = read_csv('Python_20190508_Solar_days.csv', header=0, infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
        # split into train and test
        train, test = split_dataset(dataset.values)
        # evaluate model and get RMSET
        n_input = 24
        actualvalues, forecastevalues, RMSEA, RMSET, mae, ape = evaluate_model(train, test, n_input)


        print('MAE',mae)
        # summarize RMSET
        summarize_RMSET('RMSE', RMSEA, RMSET)

        plt.figure(1)
        days  = ['1', '2', '3']
        plt.title("Forecasting Results")
        plt.xlabel("Day")
        plt.ylabel("Solar (kw/m2/day)")
        plt.plot(days,actualvalues, marker='x', label='Actual')
        plt.plot(days,forecastevalues, marker='o', label='Forecast')
        plt.grid()
        plt.legend(loc="upper right", shadow=True)
        plt.show()
```

Appendix C

Hybrid renewable energy sources microgrid sizing using Matlab

```
%Initialization.................................
    Vmean=0;
    subsigma=0;
    myus=0;
    subsigmas=0;
    Vwind=0; Gs=0; Tair=0; Pl=0;
    producttestbat=0;
    producttestwind=0;

    fVwindn=0;
    PoutVwindn=0;
    fVwindntotal=0;
    PoutVwindntotal=0;
    VwindSummer=0; VwindFall=0; VwindWinter=0; VwindSpring=0;
    GsSummer=0;GsFall=0; GsWinter=0; GsSpring=0;
    PoutVwindnSummer=0; PoutVwindnFall=0;PoutVwindnWinter=0;
    PoutVwindnSpring=0;
    PoutVwindnSummer=0;PoutVwindnFall=0;PoutVwindnWinter=0;
    PoutVwindnSpring=0;
    PwindaverageSummer=0;PwindaverageFall=0;PwindaverageWinter=0;
    PwindaverageSpring=0;
    fgsSummer=0; fgsFall=0; fgsWinter=0; fgsSpring=0;
    PsnSummer=0;PsnFall=0;PsnWinter=0;PsnSpring=0;
    PsolaraverageSummer=0;PsolaraverageFall=0;PsolaraverageWinter=0;
    PsolaraverageSpring=0;
    batterybankcapacitySummer=0;batterybankcapacityFall=0;
    batterybankcapacityWinter=0;batterybankcapacitySpring=0;
    PLSummer=0;PLFall=0;PLWinter=0;PLSpring=0;
    PbSummer=0;PbFall=0;PbWinter=0;PbSpring=0;
    AECSummer=0;AECFall=0;AECWinter=0;AECSpring=0;
    SOCSummer=0;SOCFall=0;SOCSWinter=0;SOCSpring=0;
    HsystemSummer=0;HsystemFall=0;HsystemWinter=0;HsystemSpring=0;
    HsystemsummarySummer=0; HsystemsummaryFall=0; HsystemsummaryWinter=0;
    HsystemsummarySpring=0;
    matrix1=zeros(24,1);matrix2=zeros(24,1);matrix3=zeros(24,1);matrix4=zeros(24,1);

 %.....................................................

% Datasheet of the ..................... PV Pmodule
%Dimension (m2)
As=1.94;
%module efficiency %
k1=16.23/100;
%kT is the temperature coefficient 0.4%
kT=0.004;

% Cell temperature
% Nominal operation cell temperature according to PV panel datasheet
Tnoct =44;
%reference of NOCT
Tcref=25;
%Max solar irradiation
Gmax=1.01;
%.....................................................

% Datasheet of the ..................... wind turbine 5Kw
%rated wind speed  (m/s)
Vrated=10;
%cut-in speed  (m/s)
Vin=1.5;
```

```
%cut-out wind speed (m/s)
Vout=50.0;
%Rated Power (KW)
Prated=1.0;
%.....................................................

Vb=12;
batterysize=100;
% System Cost and lifetime.....................
Tlife=25;
Tw=12;
Tbat=4;
Cw=500;
Cpv=235;
Cbat=135;
% percentage of the initial capital
kc=0.08;
%nominal rate before the inflation
e=0.06;
%escalation rate
d=0.03;
ia=(e-d)/(1+d);
%Nmax iteration is the maxium number of wind turbine
%allowed in the wind turbine alone system to meet load demand
Nwmax=6;
%Matrix Data includes all data.............

% |Time|/|Wind Speed|/|Solar Irradiation|/|Air temperature|/|Power load|

for x=1:4
    if x==1
matrix = [
1   1.97    0.001       25.84   1.32;
2   1.97    0.001       25.56   1.32;
3   1.97    0.001       25.28   1.32;
4   1.97    0.001       25.14   1.32;
5   2.06    0.001       25      1.32;
6   2.15    0.05        25      3.5;
7   2.33    0.2         25.84   3.5;
8   2.69    0.4         26.95   1.5;
9   3.31    0.6         27.5    0;
10  4.12    0.8         27.78   0;
11  4.74    0.93        28.34   0;
12  5.01    1.0         28.62   1.6;
13  5.19    1.0         28.62   1.6;
14  5.01    0.95        28.89   1.6;
15  4.83    0.8         29.03   3.5;
16  4.48    0.7         28.89   3.5;
17  4.12    0.5         28.62   3.5;
18  3.76    0.3         28.34   5.5;
19  3.13    0.1         28.06   5.5;
20  2.51    0.001       27.5    3.5;
21  2.24    0.001       26.95   2.375;
22  2.06    0.001       26.67   2.375;
23  1.88    0.001       26.39   2.375;
24  1.88    0.001       25.98   1.32;];
matrix1=matrix(:,5);
    else if x==2
       matrix = [

1.00    2.42    0       23.06   0;
2.00    2.37    0       22.78   0;
3.00    2.33    0       22.64   0;
4.00    2.28    0       22.5    1.6;
5.00    2.28    0       22.23   1.6;
6.00    2.28    0       22.23   1.6;
```

```
345)    7.00    2.33    0.05    22.5    2.375;
346)    8.00    2.46    0.2     23.89   2.375;
347)    9.00    2.69    0.4     25      2.375;
348)    10.00   3       0.52    25.56   4;
349)    11.00   3.58    0.6     25.84   4;
350)    12.00   4.03    0.7     26.12   2.375;
351)    13.00   4.3     0.65    26.39   2.375;
352)    14.00   4.34    0.6     26.39   2.375;
353)    15.00   4.25    0.5     26.39   2.375;
354)    16.00   4.03    0.3     26.12   0;
355)    17.00   3.58    0.2     25.56   0;
356)    18.00   3.36    0       25      0;
357)    19.00   2.69    0       24.73   0;
358)    20.00   2.6     0       24.17   0;
359)    21.00   2.55    0       23.89   0;
360)    22.00   2.51    0       23.62   3.5;
361)    23.00   2.46    0       23.34   3.5;
362)    24.00   2.46    0       23.06   1.5;];
363)    matrix2=matrix(:,5);
364)    else if x==3
365)    matrix = [
366)
367)    1.00    3.67    0       13.06   0;
368)    2.00    3.58    0       12.92   0;
369)    3.00    3.58    0       12.92   0;
370)    4.00    3.58    0       12.78   1.6;
371)    5.00    3.58    0       12.64   1.6;
372)    6.00    3.58    0       12.5    1.6;
373)    7.00    3.58    0.01    12.64   2.375;
374)    8.00    3.8     0.12    13.34   2.375;
375)    9.00    4.03    0.25    14.45   2.375;
376)    10.00   4.25    0.37    15.56   4;
377)    11.00   4.56    0.46    16.12   4;
378)    12.00   4.74    0.46    16.12   2.375;
379)    13.00   4.74    0.4     16.25   2.375;
380)    14.00   4.65    0.3     16.39   2.375;
381)    15.00   4.25    0.2     16.53   2.375;
382)    16.00   3.8     0.1     15.84   0;
383)    17.00   3.45    0       15.28   0;
384)    18.00   3.49    0       14.73   0;
385)    19.00   3.58    0       14.17   0;
386)    20.00   3.58    0       13.89   0;
387)    21.00   3.67    0       13.62   0;
388)    22.00   3.72    0       13.48   3.5;
389)    23.00   3.76    0       13.34   3.5;
390)    24.00   3.72    0       13.06   1.5;];
391)    matrix3=matrix(:,5);
392)            else if x==4
393)                    matrix = [
394)
395)    1.00    2.87    0       17.5    0;
396)    2.00    2.87    0       17.23   0;
397)    3.00    2.82    0       17.09   0;
398)    4.00    2.91    0       16.95   1.6;
399)    5.00    2.91    0       16.67   1.6;
400)    6.00    2.96    0.02    16.67   1.6;
401)    7.00    3       0.15    17.23   3.5;
402)    8.00    3.27    0.35    18.34   3.5;
403)    9.00    3.54    0.52    19.45   3.5;
404)    10.00   3.8     0.7     19.73   5.5;
405)    11.00   4.25    0.8     20.28   5.5;
406)    12.00   4.48    0.85    20.56   3.5;
407)    13.00   4.56    0.85    20.84   2.375;
408)    14.00   4.56    0.75    21.12   2.375;
409)    15.00   4.43    0.65    21.25   2.375;
410)    16.00   4.25    0.5     21.12   1.32;
```

```
    17.00   3.8     0.3     20.56   1.32;
    18.00   3.36    0.1     20.28   1.32;
    19.00   3       0       19.73   1.32;
    20.00   2.91    0       19.17   1.32;
    21.00   2.91    0       18.75   1.32;
    22.00   2.91    0       18.34   3.5;
    23.00   2.91    0       18.06   3.5;
    24.00   2.91    0       17.78   1.5;];
    matrix4 = matrix(:,5);
                end
            end
            end

           A= matrix1.';
           B= matrix2.';
           C= matrix3.';
           D= matrix4.';
           neuralinputs = horzcat(A,B,C,D);

    % Train inputs/targets values
    inputs = [1.3 1.3 1.3 1.3 1.3 3.5 3.5 1.5 0 0 0 1.6 1.6 1.6 3.5 3.5 3.5 5.5 5.5 3.5 2.3 2.3 2.3 1.3];
    targets= [2.0 2.0 2.0 2.0 2.0 4.0 4.0 2.0 1.0 1.0 1.0 2.0 2.0 2.0 4.0 4.0 4.0 6.0 6.0 4.0 3.0 3.0 3.0 2.0];

    net=network(1,2,[1;0],[1;0],[0 0;1 0],[0 1]);

    net.layers{1}.size=5;
    net.layers{1}.transferFcn='logsig';

    net=configure(net,inputs,targets);

    % network training
    net.trainFcn = 'trainlm';
    net.performFcn = 'mse';
    net = train(net,inputs,targets);

    % network response after training

    newtargets = sim(net,neuralinputs);
    Nwmax=round(max(newtargets),1);
    % view(net);

        end

    % end of Season.....................................

    nbrmatrix = length(matrix(:,1));
    NBR=nbrmatrix;
    time=zeros(nbrmatrix,1);
    PoutVwindn=zeros(nbrmatrix,1);
    fVwindn=zeros(nbrmatrix,1);
    Tc=zeros(nbrmatrix,1);
    Gs=zeros(nbrmatrix,1);
    deltaTemp=zeros(nbrmatrix,1);
    Psn=zeros(nbrmatrix,1);
    fgs=zeros(nbrmatrix,1);
    PL=zeros(nbrmatrix,1);
    Pg=zeros(nbrmatrix,1);
    Pb=zeros(nbrmatrix,1);
    AEC=zeros(nbrmatrix,1);
    AECmax=zeros(nbrmatrix,1);
    AECmin=zeros(nbrmatrix,1);

    DOD=50/100;
    Npv=zeros(Nwmax,1);
```

```
475)    batterybankcapacity=zeros(Nwmax,1);
476)    Cb=zeros(Nwmax,1);
477)    E=zeros(Nwmax,1);
478)    D=zeros(Nwmax,1);
479)    Hrep=zeros(Nwmax,1);
480)    Hope=zeros(Nwmax,1);
481)    Hcap=zeros(Nwmax,1);
482)    Nwind=zeros(Nwmax,1);
483)    Hsystem=zeros(Nwmax,1);
484)    Hsystemoptimal=zeros(Nwmax,1);
485)    SOC=zeros(nbrmatrix,1);
486)    initialstate=zeros(nbrmatrix,1);
487)    time=matrix(:,1);
488)    Vwind=matrix(:,2);
489)    Gs=matrix(:,3);
490)    Tair=matrix(:,4);
491)    PL=matrix(:,5);
492)
493)    % Wind Variables:
494)    % the standard deviation & Mean values of wind speed data input.....
495)    Vmean=sum(single(Vwind))/NBR;
496)    subsigma=(sum(single(Vwind))-Vmean)^2;
497)    subsigmat=subsigma/(NBR);
498)    sigmaw = sqrt(subsigmat);
499)
500)    % Solar Variable:
501)    % the standard deviation & Mean values of Solar data input.................
502)    myus=sum(single(Gs))/NBR;
503)    subsigmas=(sum(single(Gs)-myus)^2);
504)    sigmas=sqrt(subsigmas/NBR);
505)
506)        for j=1:Tbat
507)            producttestbat=producttestbat+(1/((1+ia)^Tlife*j));
508)        end
509)        for u=1:Tw
510)            producttestwind=producttestwind+1/((1+ia)^(Tlife*u));
511)        end
512)
513)    for n=1:nbrmatrix
514)    %.....................Estimating the output power of Wind turbine power.............
515)    % equation (1).............................Weibull distribution function
516)
517)    % K is the shape factor
518)    k=(sigmaw/Vmean)^(-1.086);
519)    gammak=gamma(1+1/k);
520)    % c is the scale factor
521)    c=(Vmean)/(gammak);
522)    fVwindn(n)=((k/c)*((Vwind(n)/c)^(k-1)))*exp(-(Vwind(n)/c)^k);
523)
524)    % equation (3).........................Weibull distribution function ct'd
525)    if Vwind(n)>=Vin && Vwind(n)<=Vrated
526)       PoutVwindn(n) = Prated*((Vwind(n)^k-Vin^k) / (Vin^k-Vrated^k));
527)    else if Vrated<=Vwind(n) && Vwind(n)<=Vout
528)         PoutVwindn(n) = Prated;
529)    else PoutVwindn(n) = 0;
530)         end
531)    end
532)
533)    % A= (Prated*Vin^k)/(Vrated^k-Vout^k);
534)    % B= Prated/(Vrated^k-Vout^k);
535)    %DannyK=B*Vwind(n)^k;
536)    %fun1=@(Vwind)(DK*fVwindn);
537)    %fun2=@(Vwind)(fVwindn);
538)    %Pwindaverage=integral(fun1,Vin,Vrated,'ArrayValued',true);
539)    %+Prated*integral(fun2,Vrated,Vout,'ArrayValued',true);
540)
```

```
fun4=@(Vwind)(PoutVwindn.*fVwindn);
Pwindaverage=abs(integral(fun4,0,Vout,'ArrayValued',true));

%.................Estimating the output power of Wind turbine power.............

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%.................Estimating the output power of PV array.....................

Tc(n)=Tair(n)+(Tnoct-20)*(Gs(n)/0.8);
%Temperature error of the PV module between
%cell tempeature Tc and the reference cell temperature Tcref in Celsius
deltaTemp(n)=Tc(n)-Tcref;
%the maximum output power at different radiation and temperature
Psn(n)=k1*As*Gs(n)*(1-kT*deltaTemp(n));

%equation (6)................beta distribution
Cst=(myus*(1-myus)/sigmas^2)-1;
alpha=myus*Cst;
beta=(1-myus)*Cst;
zeta=alpha+beta;

fgs(n)=betacdf(Gs(n),alpha,beta);
fun3=@(Gs)(Psn.*fgs);

Psolaraverage=integral(fun3,0,Gmax,'ArrayValued',true);

end

% Algorithm.....................................................

for ma=1:Nwmax
    Nwind(ma)=ma;
end
for ma=1:Nwmax
    Npv(ma)=ma;
end

for s=1:Nwmax

    %equation (17)................Number of PV panels
    totalpowerload=sum(single(PL));
    totalPwindaverage=sum(single(Pwindaverage));
    totalPsolaraverage=sum(single(Psolaraverage));

    Npv(s)=(((totalpowerload)-(Nwind(s)*totalPwindaverage))/totalPsolaraverage);

    for t=1:nbrmatrix
    Pg(t)=Npv(s)*Psolaraverage(t)+Nwind(s)*Pwindaverage(t);
    %Pb is the power flowing in and out the battery in KW

    Pb(t)=Nwind(s)*Pwindaverage(t)+Npv(s)*Psolaraverage(t)-PL(t);

    % AEC is the available Energy capacity in KA.H
     AEC(t)=(Pg(t)-PL(t))/Vb;

    end

    AECmax=max(AEC);
    AECmin=min(AEC);
    minus=AECmax-AECmin;
    maxp=max(abs(Pb)/Vb);
    maxd=max(minus,maxp)/DOD;
    batterybankcapacity(s)=(maxd);
    Cb(s)=round(((batterybankcapacity(s))*1000/batterysize),0);
    Npv(s)=round(Npv(s)/As,0);
```

```
607)
608)    % Battery State of Charge percentage %
609)        SOC=(AEC/batterybankcapacity(s));
610)        SOCmax=max(SOC);
611)        SOCmin=min(SOC);
612)        initialstate=(SOCmax-SOCmin);
613)
614)    % Analysis of the system Cost.................
615)
616)    Hcap(s)=Nwind(s)'*Cw+Npv(s)'*Cpv+Cb(s)'*Cbat;
617)    Cope(s)=Hcap(s)'*kc;
618)    Hope=Cope'*Tlife;
619)    D(s)=Cb(s)'*(Cbat*producttestbat);
620)    E(s)=Nwind(s)'*(Cw*producttestwind);
621)    Hrep=E+D;
622)    Hsystem(s)=Hcap(s)+Hope(s)+Hrep(s);
623)    if Npv(s)<0
624)    Npv(s)=0;
625)    else Npv(s);
626)    end
627)    end
628)    Hsystemsummary=zeros(Nwmax,5);
629)    Hsystemsummary(:,1)=Nwind;
630)    Hsystemsummary(:,2)=Npv;
631)    Hsystemsummary(:,3)=Cb;
632)    Hsystemsummary(:,4)=batterybankcapacity;
633)    Hsystemsummary(:,5)=Hsystem;
634)    Hsystemoptimal(:,1)=min(HsystemSummer)
635)    Hsystemoptimal(:,2)=min(HsystemFall)
636)    Hsystemoptimal(:,3)=min(HsystemWinter)
637)    Hsystemoptimal(:,4)=min(Hsystem)
638)    tt=0;
639)    Hsytemopitmalfinal=0;
640)    tt = max(Hsystemoptimal)
641)    Hsytemopitmalfinal = max(tt)
642)    SOC(:,1)=SOC(:,1)+abs(SOCmin)+(1-DOD);
643)
644)    if(x==1)
645)        PoutVwindnSummer=PoutVwindn;
646)        PwindaverageSummer=Pwindaverage;
647)        VwindSummer=Vwind;
648)        GsSummer=Gs;
649)        fgsSummer=fgs;
650)        PsnSummer=Psn;
651)        PsolaraverageSummer=Psolaraverage;
652)        batterybankcapacitySummer=batterybankcapacity;
653)        PLSummer=PL;
654)        PbSummer=Pb;
655)        AECSummer=AEC;
656)        SOCSummer=SOC;
657)        HsystemSummer=Hsystem;
658)        HsystemsummarySummer=Hsystemsummary;
659)            else if (x==2)
660)            PoutVwindnFall=PoutVwindn;
661)            PwindaverageFall=Pwindaverage;
662)            VwindFall=Vwind;
663)            GsFall=Gs;
664)            fgsFall=fgs;
665)            PsnFall=Psn;
666)            PsolaraverageFall=Psolaraverage;
667)            batterybankcapacityFall=batterybankcapacity;
668)            PLFall=PL;
669)            PbFall=Pb;
670)            AECFall=AEC;
671)            SOCFall=SOC;
672)            HsystemFall=Hsystem;
```

```
673)            HsystemsummaryFall=Hsystemsummary;
674)                elseif (x==3)
675)                PoutVwindnWinter=PoutVwindn;
676)                PwindaverageWinter=Pwindaverage;
677)                GsWinter=Gs;
678)                VwindWinter=Vwind;
679)                fgsWinter=fgs;
680)                PsnWinter=Psn;
681)                PsolaraverageWinter=Psolaraverage;
682)                batterybankcapacityWinter=batterybankcapacity;
683)                PLWinter=PL;
684)                PbWinter=Pb;
685)                AECWinter=AEC;
686)                SOCWinter=SOC;
687)                HsystemWinter=Hsystem;
688)                HsystemsummaryWinter=Hsystemsummary;
689)                    elseif (x==4)
690)                    PoutVwindnSpring=PoutVwindn;
691)                    PwindaverageSpring=Pwindaverage;
692)                    GsSpring=Gs;
693)                    VwindSpring=Vwind;
694)                    fgsSpring=fgs;
695)                    PsnSpring=Psn;
696)                    PsolaraverageSpring=Psolaraverage;
697)                    batterybankcapacitySpring=batterybankcapacity;
698)                    PLSpring=PL;
699)                    PbSpring=Pb;
700)                    AECSpring=AEC;
701)                    SOCSpring=SOC;
702)                    HsystemSpring=Hsystem;
703)                    HsystemsummarySpring=Hsystemsummary;
704)                    end
705)                end
706)            end
707)        end
708)    end
709)
710)    %Plotting.....................................
711)
712)    figure(1);
713)    plot(time,VwindSummer,'-.r*',time,VwindFall,'-.bo',time,VwindWinter,'-.gx',time,VwindSpring,'-
    .m+');
714)    grid on
715)    title('Average Wind Speed Distribution');
716)    xlabel('Time of Day');
717)    ylabel('Avg. Wind Speed (m/s)');
718)
719)    figure(2);
720)    plot(time,abs(PwindaverageSummer),'-.r*',time,abs(PwindaverageFall),'-
    .bo',time,abs(PwindaverageWinter),'-.gx',time,abs(PwindaverageSpring),'-.m+');
721)    grid on
722)    title('Average Wind Power Distribution');
723)    xlabel('Time of Day');
724)    ylabel('Avg. Wind power (Kw)');
725)
726)    figure(3);
727)    plot(time,GsSummer,'-.r*',time,GsFall,'-.bo',time,GsWinter,'-.gx',time,GsSpring,'-.m+');
728)    grid on
729)    title('Average Solar Irradiation Distribution');
730)    xlabel('Time of Day');
731)    ylabel('Avg. Solar Irradiation (Kw/m2)');
732)
733)    figure(4);
734)    plot(time,abs(PsolaraverageSummer),'-.r*',time,abs(PsolaraverageFall),'-
    .bo',time,abs(PsolaraverageWinter),'-.gx',time,abs(PsolaraverageSpring),'-.m+');
735)    grid on
```

```
736)    title('Average Solar Power Distribution');
737)    xlabel('Time of Day');
738)    ylabel('Avg. Solar power (Kw)');
739)
740)    figure(5);
741)    plot(time,PLSummer,'-.r*',time,PLFall,'-.bo',time,PLWinter,'-.gx',time,PLSpring,'-.m+');
742)    grid on
743)    title('Average Power Load Demand');
744)    xlabel('Time of Day');
745)    ylabel('Power (Kw)');
746)
747)    figure(6);
748)    plot(batterybankcapacitySummer,HsystemSummer,'-.r*',batterybankcapacityFall,HsystemFall,'-
       .bo',batterybankcapacityWinter,HsystemWinter,'-.gx',batterybankcapacitySpring,HsystemSpring,'-
       .m+');
749)    grid on
750)    title('System Cost in each Season');
751)    xlabel('battery bank capacities');
752)    ylabel('Life Cycle Cost of System');
753)
754)    figure(7);
755)    plot(time,AECSummer*1000,'-.r*',time,AECFall*1000,'-.bo',time,AECWinter*1000,'-
       .gx',time,AECSpring*1000,'-.m+');
756)    grid on
757)    title('Available Energy Capacity');
758)    xlabel('Time of Day');
759)    ylabel('Available Energy Capacity (A.H)');
760)
761)    figure(8);
762)    plot(time,SOCSummer*100,'b--o');
763)    grid on
764)    title('Battery State of Charge');
765)    xlabel('Time Horizon');
766)    ylabel('Battery State of Charge (%)');
767)    axis([1 24 0 100]);
768)
769)    figure(9);
770)    plot(time,SOCFall*100,'b--o');
771)    grid on
772)    title('Battery State of Charge');
773)    xlabel('Time Horizon');
774)    ylabel('Battery State of Charge (%)');
775)    axis([1 24 0 100]);
776)
777)    figure(10);
778)    plot(time,SOCWinter*100,'b--o');
779)    grid on
780)    title('Battery State of Charge');
781)    xlabel('Time Horizon');
782)    ylabel('Battery State of Charge (%)');
783)    axis([1 24 0 100]);
784)
785)    figure(11);
786)    plot(time,SOCSpring*100,'b--o');
787)    grid on
788)    title('Battery State of Charge');
789)    xlabel('Time Horizon');
790)    ylabel('Battery State of Charge (%)');
791)    axis([1 24 0 100]);
792)
793)    Summerfig=hgload('figure8.fig');
794)    Fallfig=hgload('figure9.fig');
795)    Winterfig=hgload('figure10.fig');
796)    Springfig=hgload('figure11.fig');
797)
798)    figure(14);
```

```
799)    h(1)=subplot(1,4,1);
800)    h(2)=subplot(1,4,2);
801)    h(3)=subplot(1,4,3);
802)    h(4)=subplot(1,4,4);
803)
804)    copyobj(allchild(get(Summerfig,'currentaxes')),h(1));
805)    copyobj(allchild(get(Fallfig,'currentaxes')),h(2));
806)    copyobj(allchild(get(Winterfig,'currentaxes')),h(3));
807)    copyobj(allchild(get(Springfig,'currentaxes')),h(4));
808)
809)    l(1)=legend(h(1),'Summer');
810)    l(2)=legend(h(2),'Fall');
811)    l(3)=legend(h(3),'Winter');
812)    l(4)=legend(h(4),'Spring');
```