Politecnico di Torino

Department of CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master's Degree course in Mechatronic Engineering

Master's Degree Thesis

Prevision model for energy production in solar concentrator using Artificial Neural Network

Candidate:

Leonardo Ricci

Supervisor:

Eng. Davide Papurello

Academic Year 2020/2021

A mio padre, che non vedeva l'ora

che arrivasse questo momento.

Goditi lo spettacolo.

Ti porterò sempre con me.

Abstract

During the last few decades, technology and energy demands are playing a greater role in modern society. So far, most of the energy production comes from oil, natural gas, and coal, all of which are depleting in nature. Renewable energies are crucial for the future of both humanity and the world's ecosystems. Amongst them, solar energy (SE) is the one that has been the most widely adopted and received the most financial investment. SE is produced either by Photovoltaic (PV) or Concentrator Solar Power (CSP); the first system is the most prevalent, thanks to a well-established technology and a drastic reduction of costs allowing mass production. CSP, however, requires greater funding for technology penetration and need a considerable surface. Despite these factors, CSP is getting more and more attentions thanks to the use of Thermal Energy Storage (TES). This storage allows the decoupling of solar randomness and intermittency. There are two main ways to enhance the efficiency of these systems: by developing and implementing new materials or by studying the operating conditions for optimizing and predicting performances. In recent years, Artificial Neural Network (ANN) and Artificial Intelligence have become essential for this purpose. Their capability of identifying "hidden" correlations between inputs and outputs, outperforms the former and more complex methods for solving statistical analyses. This thesis aims to find the optimal ANN coupled with a Solar Parabolic Dish. The investigation was done through a backpropagation artificial neural network (BPANN) with different learning algorithms, i. e. Levenberg-Marquardt, Bayesian regularization, Resilient Backpropagation, and Scaled Conjugate Gradient. The net was fed with seven atmospheric condition parameters (humidity, temperature, pressure, wind velocity, wind direction, global radiation and rain), and it yielded the concentrator temperature parameter as output. The seeking of the best model was conducted analysing different architectures, from single to double hidden layers and from one to twenty hidden neurons. The evaluation criteria, used to classify prediction accuracy and goodness of model, were: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Bias Error (MBE), correlation coefficient (R) and R-squared (R^2). The analyses were supported by Taylor diagrams which enhance the architectures evolution trends through plots. Among the 160 architectures studied, Bayesian regularization characterized by two hidden layers with nineteen and fifty-seven hidden neurons respectively is the best model for this application. All the errors of all models can be found in the result section.

Contents

List of Figures

List of Tables

Acronyms

ADALINE	Adaptive Linear Neuron
ANN	Artificial Neural Network
AQHI	Air Quality Health Index
AQI	Air Quality Index
BPANN	Backpropagation Artificial Neural Network
BR	Bayesian Regularization
CERES	Cloud and the Earth's Radiation Energy System
CNRS	National Scientific Research Council
CSP	Concentrated Solar Power
EU	European Union
FAO	Food and Agriculture Organization
FFNN	Feedforward Neural Network
GHG	Greenhouse-Gas
HL	Hidden Layer
LFC	Linear Fresnel Concentrator
LFR	Linear Fresnel Reflector
LHS	Latent Heat Storage
LM	Levenberg-Marquardt
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MBE	Mean Bias Error
MLR	Multiple Linear Regression
PFC	Point Focus Concentrator
PTC	Parabolic Trough Collectors
RMSE	Root Mean Squared Error
RPROP	Resilient Propagation
SCG	Scaled Conjugate Gradient
SHS	Sensible Heat Storage
SPD	Solar Parabolic Dish
SPT	Solar Power Tower
STTF	Short-Term Temperature Forecasting
TES	Thermal Energy Storage
TRCM	Typhoon Rainfall Climatology Model

1 Introduction

1.1 The importance of renewables energy

During the last few decades, technology and energy demands are playing one of the most important roles in modern society. So far, most of the energy production comes principally from fossil fuels (i.e. oil, natural gas, and coal) and nuclear power. Even if they are widely used, their utilization has highlighted several environmental, economic and social questions. All of them are considered non-renewable energy since they are depleting in nature and their amount is finite. Moreover, it is widely known that fossil fuels are responsible for the majority of greenhouse gas (GHG) emissions, such as CO_2 emissions, determining the increase of the global mean temperature, called global warming [1]. The goal to stabilize the concentration of the GHGs in the atmosphere was reached in 1997 with the signing of the Kyoto Protocol, which defined also the human processes and activities, involved in global warming [2]. In particular, 65% of global GHG emissions derived from burning fossil fuels and industrial processes [3]. Despite GHG emissions and global warming, fossil fuel utilization has also an important role on other pollution forms in all environmental compartments. Even if nuclear energy is not defined as a possible energy resource in Italy, also in Germany the energy policy has been revised in favour to other energy forms. This nuclear abandonment has been strongly accelerated after the Fukushima disaster [4]. Hence, renewable energy forms are crucial in this energy substitution. As the physicist Cesare Marchetti points out: i. all historical energy transitions occur with the parallel improvement and diffusion of technological innovations*; ii. "the introduction of new primary energies requires 10 to 20 years of observation before understanding the long-term market behavior"* [5]. Post the Kyoto Protocol, many environmental policies of several Countries have been changed or implemented in order to achieve many goals, such as i. GHG emission and other pollutant reductions; ii. the protection, restoration and preservation of the ecosystems, i.e. environments and their biodiversity; iii. a more efficient use of the natural resources, promoting e.g. material recycling [6]. Indeed, this energy transition from nonrenewable energy forms to renewable energy technologies is one of the most priority of European Union [1]. EU policies are oriented on renewable energy valorization promoting the environmental sustainability concept through large investments, including their diffusion and the research of new and efficient technologies. For this purpose the European Green Deal [7] is aimed to make the continent climate-neutral by 2050. Eurostat

has reported how renewable energy trend is important and necessary in electricity, heating and cooling, and transport. This attention has led to the doubling of renewables share in the period 2004-2019 with gross final energy consumption of 19.7%. The two largest renewable expansions, in electricity production, are wind and hydro power with 35%. The remaining percentage is composed of 13% as solar power, 8% as solid biofuels and 9% as other renewable sources. A particular attention is devoted to fast and huge growth of solar energy (SE) passing from 7.4 TWh of 2008 to 125.7 TWh of 2019 [8].

1.2 Brief Concentrator Solar Power (CSP) history

The industrial revolution had a big impact on solar energy technology development and allowed scientists to conduct many types of research and experiments. The Stirling cycle was one of the most important developments of this period. It aroused great interest and around 1870, John Ericsson coupled it with point-focusing concentrators [9]. The first attempts of transforming solar energy into thermal energy began in the second half of the nineteenth century. Auguste Mouchot invented a solar concentrator that drove a steam engine; he showed his engine at the Universal Exposition in Paris 1878 [10]. This machine presented some critical issues, notably that it was small and the boiler re-radiated too much energy. Subsequently, the engineer Alessandro Battaglia, obtained a patent in Genoa in 1886 for the invention of the "Multiple solar collector". This device managed to overcome the limitations of Auguste Mouchot's design as the boiler was separated from the mirror. Unfortunately, his work remained unknown until 2008 when it was 'rediscovered' by the Central State Archive during a reorganization and categorization of his archives [11]. In the following years, his research and ideas on solar technology were brought to the U.S. by French scientists. The First World War contributed to a greater acceptance of solar technology through the construction and installation of solar engines which were mainly used for heating and pumping water. This interest in solar technology is illustrated by the many patents of solar heaters and collectors that were granted in that period. Later, Frank Shumann built the world's first solar thermal power station in Maadi, Egypt to pump water [9]. His design boosted many technological advancements including absorption plates with dual panes separated by a one-inch air space. These advances were revisited in the 1970s when an interest in solar thermal energy resumed. Giovanni Francia, a Turin mathematician, realized that solar heat could be exploited for a far more complex and advanced machine however, it would be necessary to increase the flux density and temperature. In this context, Francia began working on two designs: the

honeycomb-cell and the Fresnel reflectors [12]. The first design consisted of a large number of long, thin parallel tubes of transparent material for solar radiation and opaque material for the thermal rays emitted by the hot surface; this modification allowed minimization of re-irradiation and convection losses. Furthermore, Giovanni Francia made a crucial observation: it was much easier to build flat mirrors, rather than a large curved ones, both in terms of the complexity of the manufacturing and the overall cost. In the system designed by Francia the receiver is fixed, supported by sturdy towers, and capable of gathering all the sunlight from the area below. In 1961, at the conference held by United Nation 'Food and Agriculture Organization' (FAO) in Rome, Francia presented the results obtained through tests conducted on the honeycomb structures. He reported that it had reached a temperature of 600°C. The tests were performed at the first experimental solar station located in Cesana Torinese. Giovanni Francia was the first person in the world to apply the Fresnel reflector concentrator concept to an actual linear and point focus systems (LFCs and PFCs). He realized the first LFC prototype, in Sant'Ilario (GE). It was capable of producing steam at 150 atm and 500 °C. In 1964, in Lacédémone-Marseilles, in collaboration with Marcel Perrot, and with support from France's National Research Council (CNRS), NATO and COMPLES (Coopération Méditerranée pour l'Energie Solaire) he designed a solar station with a compact linear Fresnel reflector(CLFR). Based on Francia's studies, the Eurelios (CA) solar tower plant was designed and built. The inauguration took place on April 14, 1981. It remained active until 1985. The results, published in 1991, highlighted that the plant was not profitable because the cost of electricity was slightly higher than the maintenance costs [13]. Eurelios's archetype had begun to spread worldwide and as a result many countries started to invest in solar plant technology.

1.3 Brief ANN historical introduction

The first keystone is assigned to McCulloch and Pitts in 1943, who stated that "*for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time*" [14]. Their work was able to process a series of Boolean input and if an input takes the value '1', which indicates as "neuron fire". Donald Olding Hebb, in The Organization of Behavior (1949), studied the first learning hypothesis; according to him "*Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. ... When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased*" [15]. In 1954,

Gabor developed the learning filter, which exploits the gradient descendent to get "optimal" weights that minimize mean square error between an observed and a forecasted value. In 1958, Rosenblatt created the perceptron [16]. This new design allowed to optimize McCulloch and Pitts studies, by introducing input weights and a sum. Two years later, Widrow and Hoff developed Adaptive Linear Neuron (ADALINE) which is a simple neural network trained by means of the Least Mean Square error [17]. During these years, Rosenblatt has continued studying these algorithm and, in 1961, proposed the backpropagation [18]. Although his proposal was rejected because the step function was not differential everywhere, which is a requirement for the application of backpropagation. Only eight yeas later, backpropagation approach got attentions. Minsky and Papert, 1969, demonstrated the limits of perceptron whose learning method could not be applied in networks with more than two layers (input and output). Moreover they enriched their thesis with the sigmoid activation function, introduced some years before by Cowan [19], thanks to its capability of smoothing the weighted sum and to classify the output. Furthermore their investigation brought to a second result: computers of those time were not able to process big amount of data [20]. This technological limit made neural network studies slower and unattractive. Only in 1986, David E. Rumelhart, Geoffrey E. Hinton & Ronald J. William with their "Learning representations by back-propagating errors" made ANN of great attentions. Learning became more interesting, but more difficult once the authors introduced hidden units whose states are not specified by the task; according to Rumelhart et al., "*The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behavior*" [21]. On the contrary, in previous perceptrons input-hidden unit connections were fixed by hand, hence the states were determined which is equivalent to a mislearning. From that moment on, the massive development of computer technology, both in computational power and in memory, has enhanced ANNs growing. Newer types of architecture and more efficient algorithm have flourished. Nowadays, ANNs can be implemented in myriad of fields. These main chronological passages have led to our modern view of ANN as reported in Fig. 1 and Fig. 2 [22].

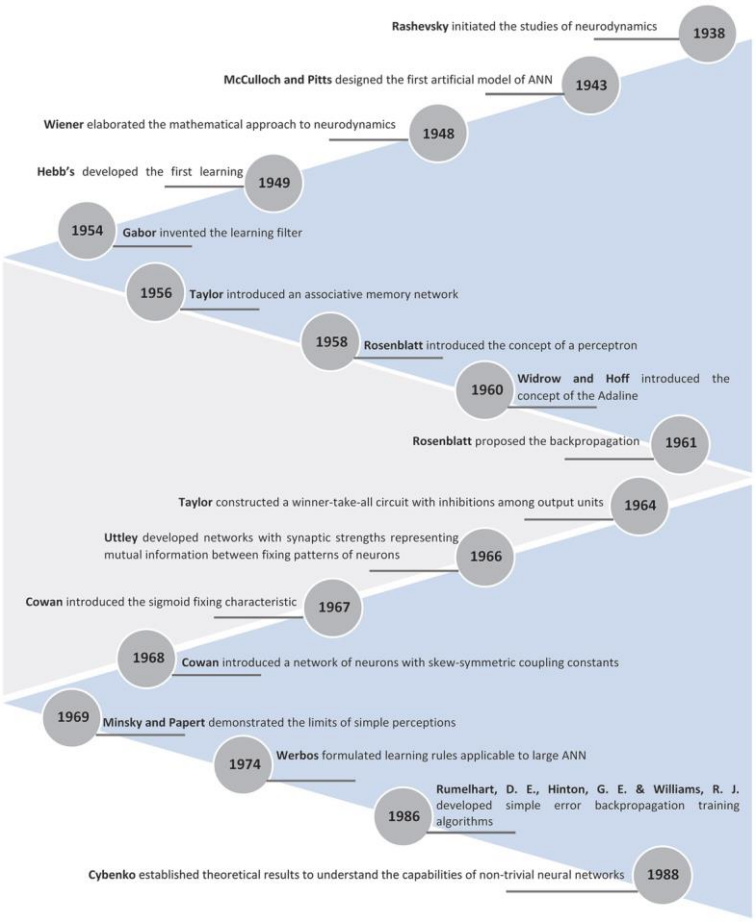


Fig. 1 ANN evolution timeline (1938–1988)

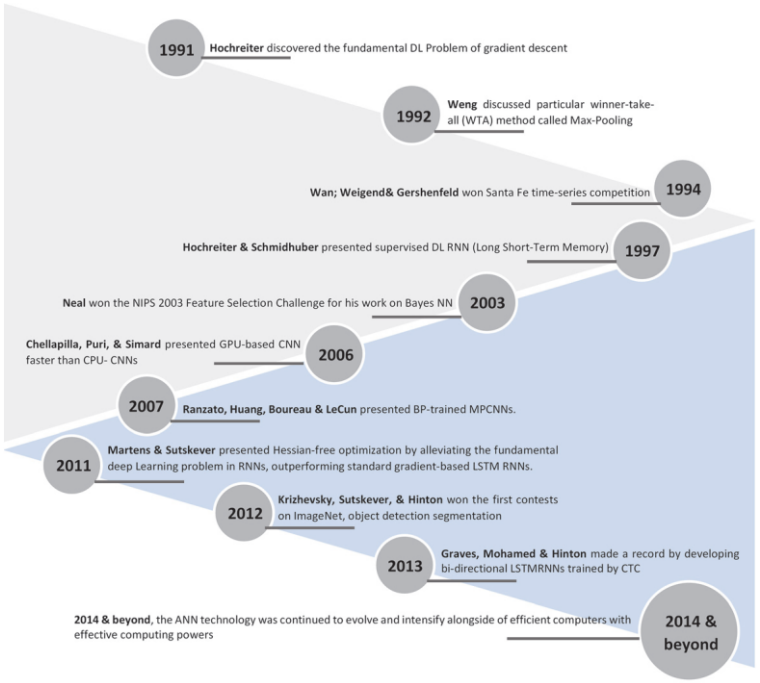


Fig. 2 ANN evolution timeline (after 1988).

1.4 Objective and structure of the thesis

This thesis aims to provide the most accurate ANN model for a dish solar concentrator, to predict energy production using weather parameters as input and solar concentrator temperature as output. The investigation includes the examination of 160 ANN architecture performances.

Chapter 2 describes the state of the art of both CSPs and ANNs; chapter 3 describes all Materials and Methods used in this thesis; chapter 4 and chapter 5 are devoted to the overall results and discussion. Chapter 6 corresponds to the conclusion of this work. The thesis includes Appendix which contains the MATLAB codes.

2 State of the Art

2.1 Concentrated Solar plant

In CSP plants, energy is generated indirectly by concentrating solar radiation. They are made of several components such as heliostats, a receiver, solar concentrators, a thermal energy storage (TES), and a source of energy production, typically turbines [23]. The power generation system consists of concentrating the sunlight onto a receiver, that carries a heat transfer medium. This medium is heated to a high temperature and later it will pass into a steam turbine. Nowadays four types of CSP are found i. Solar Parabolic Dish (SPD); ii. Parabolic Trough Collectors (PTC); iii. Solar Power Tower (SPT) and, iv. Linear Fresnel Reflectors (LFR). The design of the entire plant can be quite diverse due to the high number of variables that can be chosen. The most important are the layout, the structure, the heat transfer medium, the heat transfer mechanism and application. The final configuration is determined by the best trade-off between cost and energy production; at the moment the main costs of the plants come from heliostats and tanks due to the high number required and the manufacturing complexity. The challenge to keep low the costs and maximize efficiency is tackled with studies in the chemical and engineering field.

2.1.1 Heliostats

Heliostats are the mirrors of the plant used to focus sunlight onto the receiver. Their shape depends on the type of plants: for SPD heliostat is bowl-shaped Fig. 3, for PTC they are gutter-shaped Fig. 4, for SPT and LFR they are flat Fig. 5 Fig. 6, respectively. One of the most important factors that must be taken into account in the design of the plant is wind interference; this element must be considered as it has an impact upon both the stress and strain in the heliostat structure and misalignments with the receiver. Emes et al. showed that smaller designs reduce these problems and for this reason it requires a proper trade-off wind exposure/dimensions [24]. Moreover, the mirrors must be cleaned periodically due to dust and particle deposits on the surface. Other characteristics to evaluate are the field layout, the qualifications (intended for testing), and the control which is made through the alt-azimuth mount for the solar tracking [25].

Fig. 3 Solar Parabolic Dish
https://www.promes.cnrs.fr/index.php?page=eurodish-system#prettyPhoto

Fig. 4 Parabolic Trough Collectors
https://www.almecogroup.com/it/pages/239-concentratori-di-energia-solare

Fig. 5 Solar Power Tower
https://www.almecogroup.com/it/pages/246-eliostati

Fig. 6 Linear Fresnel Reflectors SolarPACES—Solar Power and Chemical Energy Systems

2.1.2 Receiver

2.1.2.1 Tubular

In CSP plants, tubular receivers are most commonly used because of their simple fabrication and their capability of working with high temperatures and pressures. These types of receivers are divided into two big families: External and Cavity Fig. 7; both take the name from the position of the tubes, outside and inside a chamber, respectively. The first case is characterized by high radioactive losses due to the exposure to the environment, by a slightly greater reflection loss, and by frequent maintenances because of degradation. The second case, on the contrary, minimizes all these effects however the heating is more difficult and expensive relative to the same absorber area [26]. The most significant advantage of this design comes

from the exploitation of the receiver mass and the door of the cavity because they provide a sort of thermal inertia which allows a faster startup of the system [27].

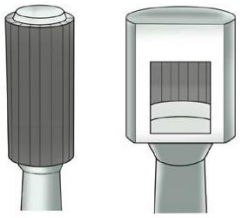
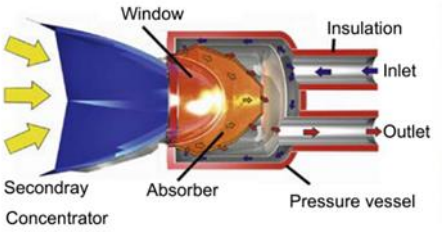


Fig. 7 Schematics of tubular (left) external and (right) cavity receivers

2.1.2.2 Volumetric

In these receivers there is direct contact between the heat absorber surface and the working fluid; the usage of porous ceramics structures makes heat transfer more efficient [28]. Open volumetric receivers use the air in the environment as heat fluid; once heated up it passes through a heat exchanger to feed a turbine with compressed air. Closed volumetric receivers are more complex since the heated fluid is pressurized. The secondary concentrator focuses sunlight in a pressure vessel, enclosed within a quartz window. This structure helps the convection and thus the overall efficiency Fig. 8 [29].



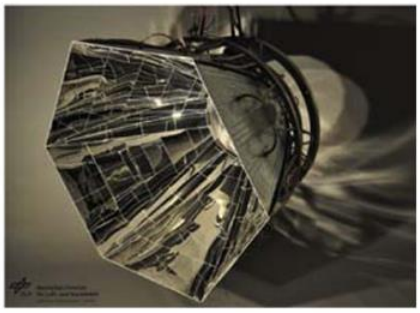


Fig. 8 Scheme of the pressurized volumetric receiver (PVR) (left); secondary concentrator (right). Copyright: DLR

2.1.2.3 Plate or Channel

A peculiarity of this type of receiver is the high thermal efficiency which can reach 90% [30,31]. This result is achieved by exploiting the full potential of the surface between the receiver and the heated transfer fluid with mini or micro-channels Fig. 9. Furthermore, it is possible to increase the number of micro-channel modules for bigger capacities [32]. Unfortunately, mass production is very difficult and expensive due to the–complexity of manufacturing and the specifications required. Coupling supercritical Rankine cycle with either supercritical CO_2 or

H_2O provides more energy to the grid but the cost of these plants are much higher mainly due to thermal storage (packed bed thermocline) and the pipes net [33].

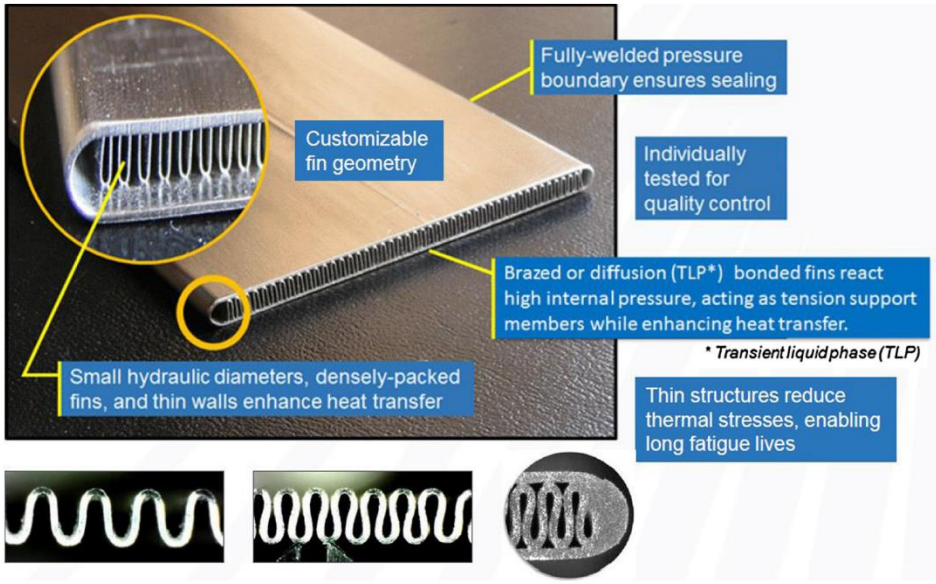


Fig. 9 Finned-plate receiver design with numerous mini-channels for fluid flow [34]

2.1.2.4 Heat Pipe

The structure of these receivers is very simple. They are composed of an upper header, some receiver tubes covered by reflectors and a lower header Fig. 10. Each reflector focuses sunlight onto its section tube which contains the intermediate fluid. Once it is heated up, the fluid starts evaporating and it moves into the receiver tube where the working fluid is condensed Fig. 11. During the entire process it is suggested to keep the receiver tube warm, typically with electrical heating, to increase the daily operating time and to reduce any possible freezing of the working fluid. The upper header is used as a collector and it allows the condensation to pass into the next receiver tube. It is possible to increase the outer surface temperature and the heat flux density by repeating cycles; the drawback is pressure drop [35].

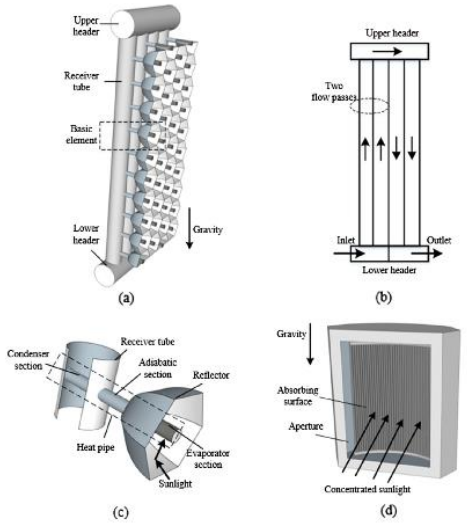


Fig. 10 Schematic of a heat pipe solar central receiver: (a) receiver panel; (b) two dimensional receiver panel; (c) basic element; (d) receiver structure (heat pipes and reflectors are not shown)

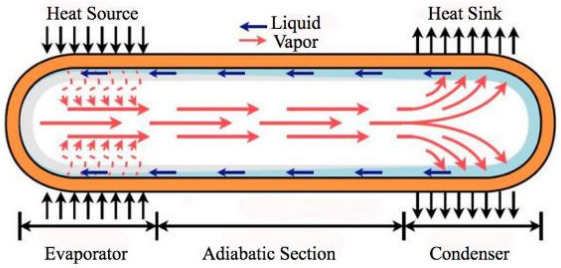


Fig. 11 High L/D Heat Pipe System Sodium-stainless steel heat pipes with very high L/D used to produce an extended array. These heat pipes act as the prime solar capture and thermal transport medium [36]

2.1.2.5 Solid Particles

Solid particle receivers are the most promising among all receivers. The heat is transferred through particles in three different mechanisms: downflow, which exploits gravity as a mean of transport Fig. 12; upflow, which generally uses air pumps Fig. 13; horizontal flow, which operates with series of multi-stage fluidized beds Fig. 14. The main advantages of these type of receivers is the capability of reaching high temperatures (over 1000°C), the high stability of the system, the ability to exploit particles themselves as thermal energy storage, the low cost of the material of the particles, and the greater efficiency of the Rankine or Brayton cycle for energy production. On the other hand, the particles flux does not create a perfect heat distribution. There are different ways to try to adjust this shortcoming, for example, Λ–shaped mesh structures Fig. 15, Porous structures Fig. 16, Hexagonal heat transfer tubes Fig. 17 or inclined plates Fig. 18, however all require greater manufacturing and maintenance costs. Moreover, the use of air pumps entails–additional engineering challenges and working with high temperatures leads to thermal stress more frequently [37].

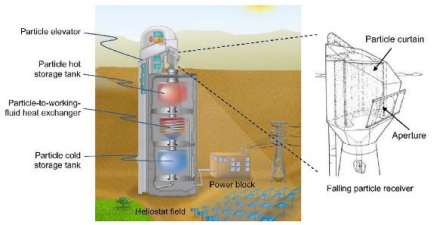


Fig. 12 Schematic of a free-falling solid particle receiver.

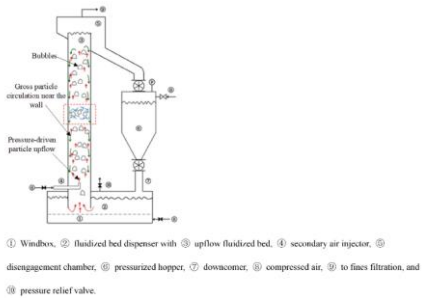


Fig. 13 Schematic of a laboratory-scale upflow fluidized particle-in-tube receiver.

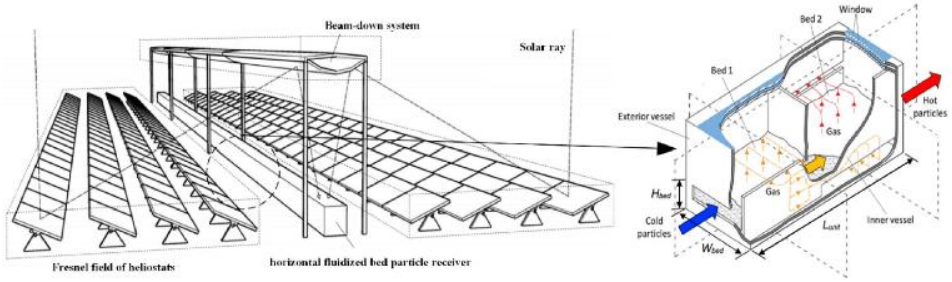


Fig. 14 Layout of the optical system and schematic diagram of a horizontal fluidized bed particle receiver.

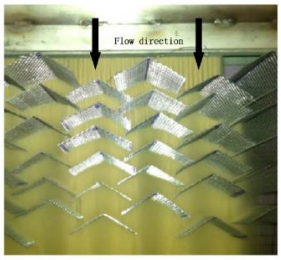


Fig. 15 Particles flowing over a staggered array of Λ-shape structures.

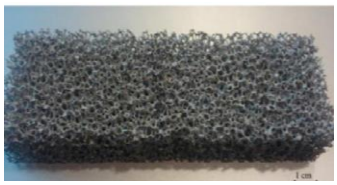


Fig. 16 Metallic porous blocks used in experiments on the cavity of a solid particle receiver.

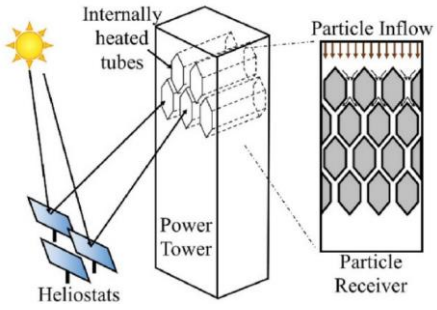


Fig 17 Schematic of an enclosed particle receiver with hexagonal heat transfer tubes.

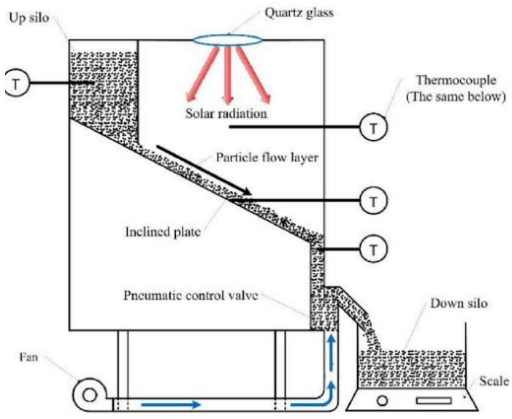


Fig 18 Schematic of an inclined plate solid particle receiver

2.1.3 Thermal Energy Storage (TES)

The adoption of TES Fig. 19 in the CSP plant plays and will play one the most important role in addressing the duck curve Fig. 20. This graph shows both the benefits of renewable energies, such as PV and Wind, during the daytime and the huge energy demand from dusk onwards [38]. Trying to charge these batteries with the common renewables or grid is very inefficient. On the contrary, CSP plants are very attractive as they are competitive and reliable. Currently, there are three different types of TES: Sensible heat storage (SHS), Latent heat storage (LHS), and Thermochemical storage. The trade-off, in regards to the technology implementation, starts from the sixth criteria: i. capacity of energy stored; ii. power intended as how fast is charging and discharging phases; iii. efficiency; iv. storage period, which expresses for how long energy can be stored and lasts; v. charge and discharge time and vi. cost referred to €/kW or €/kWh, which is the most weighted [39]. Nowadays most plants use Sensible Heat Storage due to their low cost of materials (with the exclusion of liquid metals or oils), high-temperature resistance and stability, even if they suffer in the discharging phase. Latent heat storage offers high energy density and isothermal charging/discharging, these are not commonly employed because of the heat transfer fluid which has low thermal conductivity and may be flammable (Phase Change Materials case), difficult to transport (highly lipophilic), or corrosive [40]. Thermochemical are the most promising in future applications since they have the highest thermal energy storage density, long power duration and low heat loss; however, this storage is still in the laboratory stage because more studies are required [41].

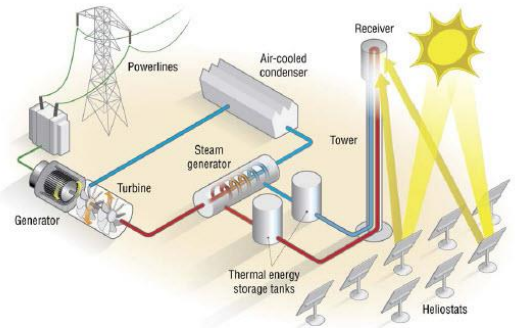


Fig. 19 Thermal Energy Storage tanks

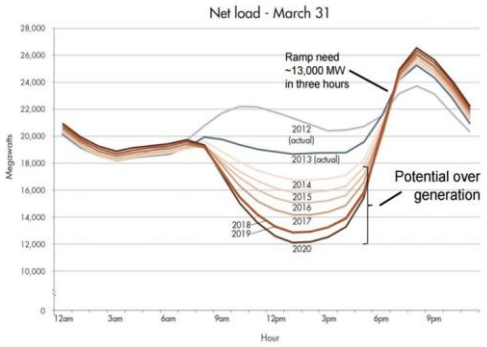


Fig. 20 The Duck Curve Graphic: California ISO and J. Lazar [42]

2.1.4 Solar Concentrator

2.1.4.1 Solar Parabolic Dish

SPD uses a bowl-shaped heliostat to focus sunlight onto the receiver Fig. 21, Fig. 22. The diameter of the SPDes varies from $5 \div 10\ m$ and the surface area is $40 \div 120\ m^2$. The shiny surface of the SPD is constructed of silver or aluminium which is coated with glass or plastic. The main advantage of plants that use the Stirling engine is that the curved mirrors used in this system always point directly towards the sun which ensures the highest efficiency. Moreover, every dish is "stand-alone" and it has not any constraints on the morphology of the terrain; in this context, it can be easily implemented in an isolated location. Unfortunately, manufacturing costs are high [43].

Fig. 21 Maricopa – Peoria, Arizona US Stirling SunCatcher with "Heat Engine" Technology. https://www.buildinggreen.com/news-article/stirling-suncatcher-heat-engine-technology

Fig. 22 The Maricopa Solar Plant . It is a part of SRP's sustainable renewable energy portfolio of providing 15% of retail energy by 2025. https://www.buildinggreen.com/news-article/stirling-suncatcher-heat-engine-technology

2.1.4.2 Parabolic Trough Collectors

In PTC plants large mirrors shaped as gutters are used to reflect sunlight onto a receiver Fig. 23, Fig. 24. The harvesting area is huge and typically includes several hundred troughs which are placed in parallel rows along the north-south axis. This configuration allows the tracking of the sun with a single-axis movement. To enhance the absorption and to minimize heat losses, all troughs are colourized. The energy production begins with the passage of heated fluid in the receiver tube. To maximize the efficiency of the system, this fluid must have a high absorption coefficient and the focal point of the trough must be placed correctly. The super-hot medium heats water via a heat exchanger, in turn, water becomes steam which feeds the turbine [43].

Fig. 23 Mojave Solar Project solar power plant Parabolic Trough Collectors, (San Bernardino County, California, US.) https://www.graywolfindustrial.com/project/mojave-solar-project-mirror-project/

Fig. 24 Mojave Solar Project solar power plant https://www.graywolfindustrial.com/project/mojave-solar-project-mirror-project/

2.1.4.3 Solar Power Tower

Even if the SPT plant requires the largest working area, they provide the highest achievable temperature of all the heat transfer medium. The structure is composed of many heliostats that point to the top of a tower where the receiver is placed Fig.25, Fig. 26. Nowadays there is not a unique choice of heat transfer medium. In the US many SPTs are running with molten nitrate salt because of its better heat transfer and stocking. In Europe, they are working with air because of its high temperature and there are no additional costs. In both plants, water plays the most important role in producing steam and this factor is crucial for all SPT plants in arid and desert regions where the lack of water is a significant issue [43].

Fig. 25 Plastic model of the Mohammed bin Rashid Al Maktoum Solar Park in Dubai https://www.dewa.gov.ae/en/about-us/strategic-initiatives/mbr-solar-park/facts-about-the-solar-park

Fig. 26 Noor Power Plant in Morocco Agadir district-Ouarzazate. In the foreground Noor 1 https://www.ecohz.com/renewable-energy-solutions/powerplants/noor-solar-power-in-morocco/

2.1.4.4 Linear Fresnel Reflectors

LFR plant is very similar to PTC because, in the same way, it exploits arrays of mirrors to focus light onto a tube. In this case, mirrors are linear and follow the same mechanism of the Fresnel lens. All Fresnel reflectors are automatically directed and able to track the sun Fig. 27, Fig. 28. The main advantage, in contrast to PTC plants, is the lower cost of mirrors [43]. The drawback is reduced efficiency because of a higher influence of the incidence angle and the cosine factor [44].

Fig. 27 eLLO Solar Thermal ProjectLlo, Pyrénées Orientales France eLLO Solar Thermal Project. Photo from ALMECO GROUP. https:// www.almecogroup.com

Fig. 28 Puerto Errado 2 Thermosolar Power Plant Spain https://www.ohlindustrial.com/en/projects/30-mw-puerto-errado-2-thermosolar-plant-murcia/#

2.2 Artificial Neural Network

Artificial Neural Networks (ANNs) [45] are an Artificial Intelligence application, developed from human brain pattern. ANNs have been successfully applied in many different fields such as medicine, industry, stock markets, biology and electronic systems. During the last few decades, the increased use of sensors and tools able to gather information (data) from the environment has demonstrated the great advantages and importance of ANNs. These networks are capable of solving both linear and non-linear approaches, overcoming classical statical methods which are strictly linked to the assumption of repeatability of past data. This outcome can be achieved because ANNs are data-driven, self-adaptive methods that do not require any prior assumptions. They 'learn' from samples and capture "unknown" correlations among the data. Since the world is made of many non-linear relations it is complex and hard to describe; for this purpose, ANNs are modelled as a black-box. The structure of Feed-forward networks is composed of an input layer, one or more hidden layers, and an output. All layers are made up of the simplest blocks, "neurons". Every neuron is directly linked to all the neurons of the following layer.

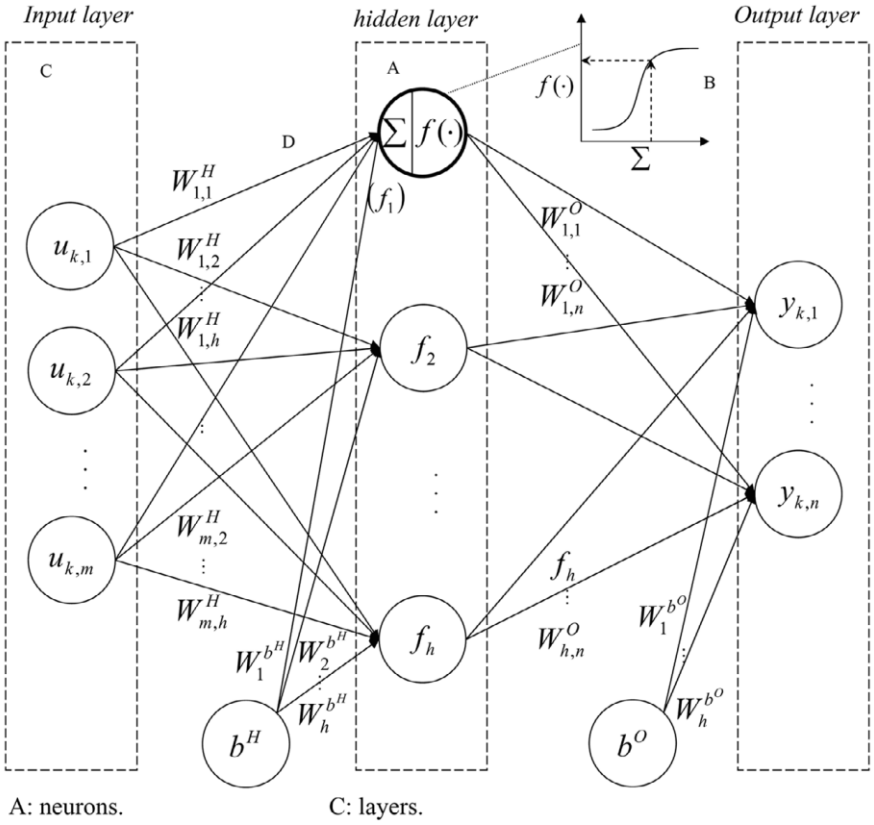


Fig. 29 A feedforward neural nework (FNN) with 1-hidden-layer multiple-input-multiple-output (MIMO) framework [46].

All available data are divided into: a training set, used to train the ANN and devoted to create a model that can identify weights and biases; a validation test that optimizes the internal parameters of the model by minimizing validation error; a testing set which is used for the evaluation the accuracy of the data forecasted. The first critical point needed for the correct configuration of an ANN it is the determination of the input and output number of neurons, which are given during the formulation of the problem, and the hidden neurons, which are more complex. Moreover, one single hidden layer is sufficient to tackle the nonlinear functions, while the usage of more layers may provide more accuracy. The most common way to determine these factors is through trial and error. There is not a clear and unique methodology, mainly because every problem has its own set of attributes and correlations. In literature, it is common to find the same problems with different models because of the high variability of the world and the hidden nonlinear relations between parameters. The second critical point is the activation function which is the transfer function that determines the relationship between two adjacent layers and its aim is to introduce the degree of nonlinearity [47]. To ensure stability these functions are bounded, monotonically increasing and differentiable. The third critical point is the training algorithm. ANNs can also be defined as an unconstrained nonlinear minimization problem whose objective function is, generally,

'Mean Square Error'. For this purpose, weights and biases are iteratively modified to minimize overall error. Although, it is hard to find the global minimum which optimizes the objective function because of the non-convex nature of problems Fig. 30.

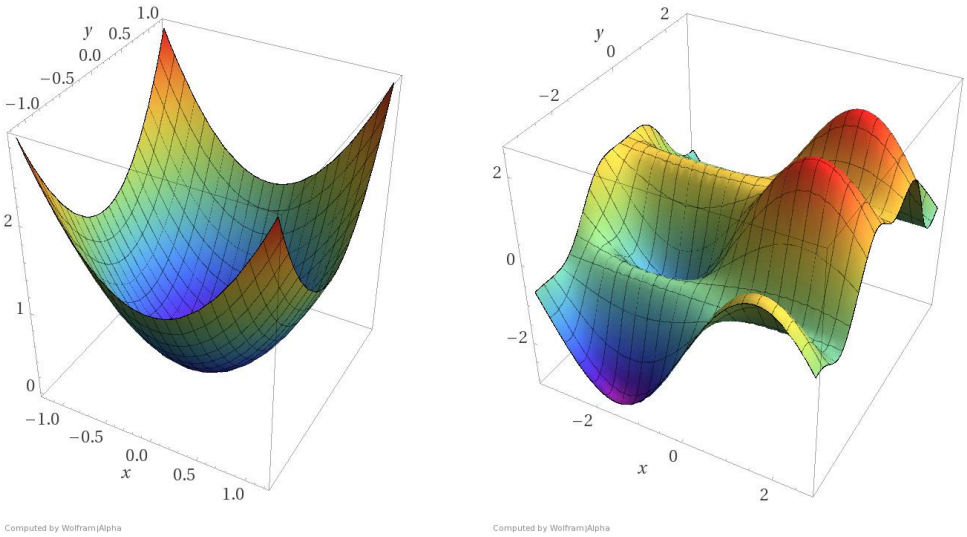


Fig. 30 Left: a convex function. Right: a non-convex function. It is much easier to find the bottom of the surface in the convex function than the non-convex surface. (Source: Reza Zadeh)

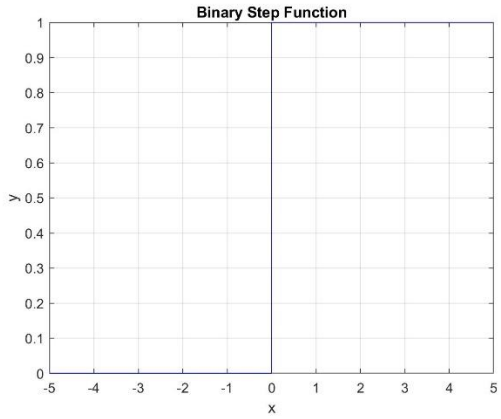
Unfortunately, there is not an algorithm capable of finding the optimal result that corresponds to the global minimum, however, there are some alternate avenues available allowing the possibility to analyze different local minima.

The fourth critical point is the normalization of data; the choice of interval in which inputs are normalized depends largely on the activation function, $[0,1]$ for logistic and $[-1,1]$ for hyperbolic. The normalization in a generic interval $[a,b]$ is given by $x = (b - a)\frac{(x-min(x))}{max(x)-min(x)} + a$.

The last critical point is the performance measure, whose most important parameter is the prediction accuracy.

2.2.1 Activation function

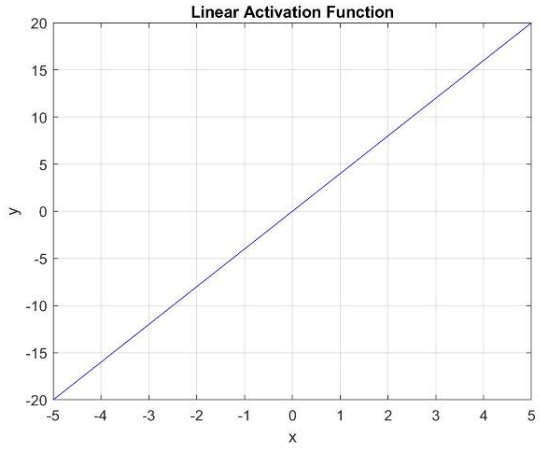
2.2.1.1 Binary Step Function:



$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

2.2.1.2 Linear Activation Function:

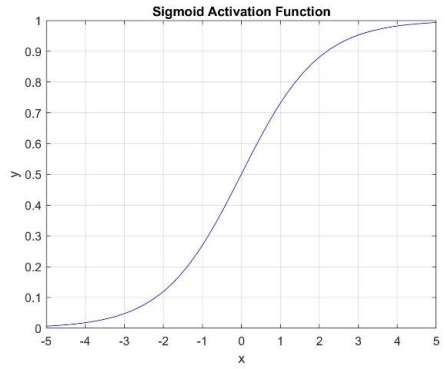
The only advantage with respect to Binary one is that the linear function has a derivative different from 0. Although this it does not improve the non-linearity of the network



$$f(x) = ax$$

2.2.1.3 Sigmoid Activation Function

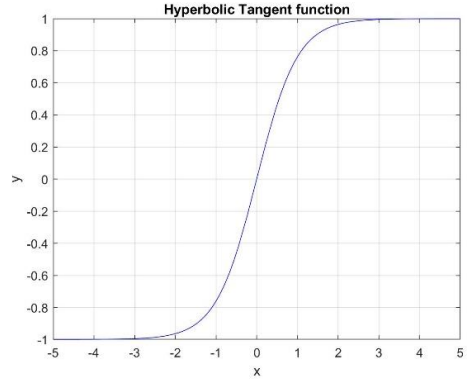
Is the most widely used function; it has a smooth S shape and derivative is: $S(x)' = 1 - S(x)$



$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1} = 1 - f(-x)$$

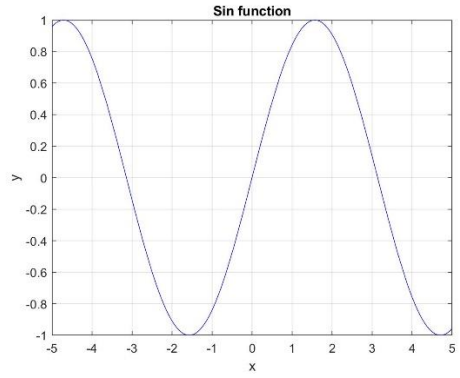
2.2.1.4 Hyperbolic (tanh) function

In contrast to Sigmoid this function is bounded in [-1,1]; it is an odd function, convex for $x < 0$ and concave for $x > 0$.

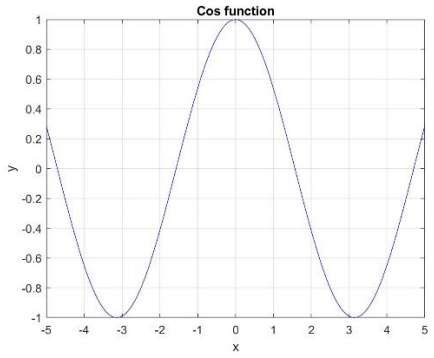


$$f(x) = \tanh x = \frac{\sinh x}{\cosh x} = \\ = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

2.2.1.5 The sine or cosine function:



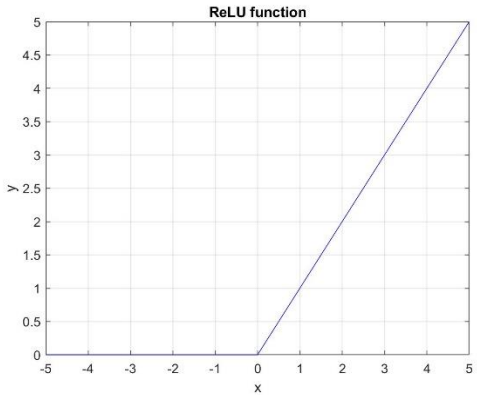
$$f(x) = \sin(x)$$



$$f(x) = \cos(x)$$

2.2.1.6 ReLU Function

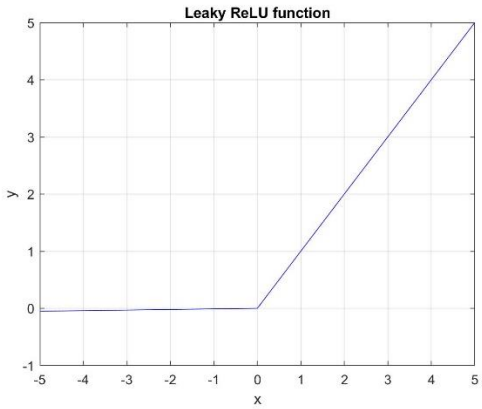
The Rectified Linear Unit is famous due to its property of activating neurons at different times and they are deactivated only when the output of linear transformation is zero.



$$f(x) = x^{+} = max(0, x)$$

2.2.1.7 Leaky ReLU Function

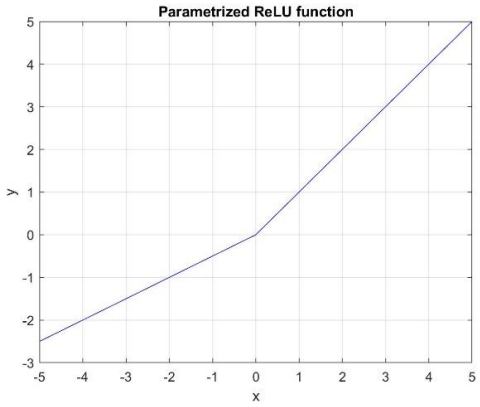
It is similar to ReLU one but the negative part has a slope close to 0



$$f(x) = \begin{cases} x, x \geq 0 \\ 0.01x, x < 0 \end{cases}$$

2.2.1.8 Parametrized ReLU Function

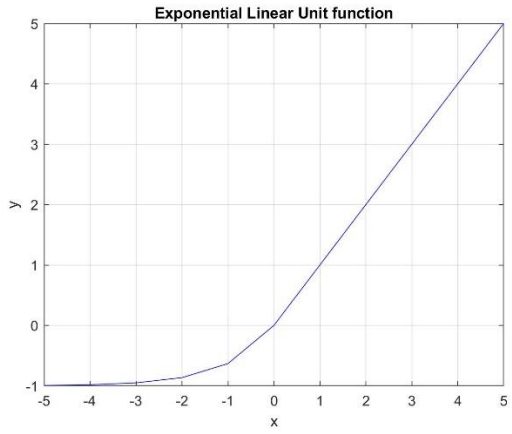
Like the Leaky function, the negative part has a slope used for a faster and far better convergence from which the network can 'learn'. When a=0.01 the function is Leaky



$$f(x) = \begin{cases} x, x \geq 0 \\ -ax, x < 0 \end{cases}$$

2.2.1.9 Exponential Linear Unit

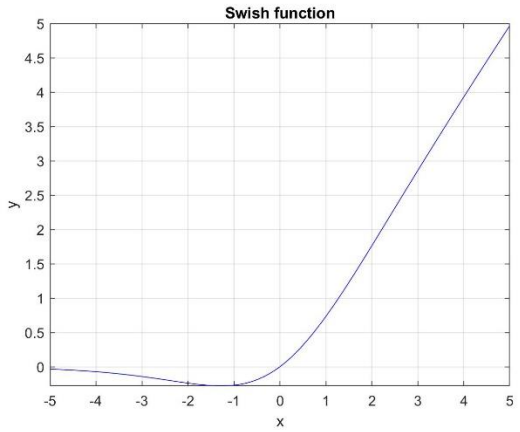
It is a variant of the ReLU function because in this case the negative part is given by $a(e^x - 1)$.



$$f(x) = \begin{cases} x, x \geq 0 \\ a(e^x - 1), x < 0 \end{cases}$$

2.2.1.10 Swish Function

The Swish function was discovered by GOOGLE researchers, and, in some cases it outperforms the ReLU function.



$$f(x) = x\text{sigmoid}\,(x) = \frac{x}{(1+e^{-x})}$$

2.2.1.11 Softmax Activation Function

The Softmax activation function is a combination of multiple Sigmoid and it is generally used for multiclass classification problems

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K$$

2.2.2 Backpropagation Learning Algorithms

Backpropagation is a class of neural networks composed of two steps Fig. 31. The first is focused on feed forward, the second is devoted to learn from the first, calculating and backpropagating the error [48].

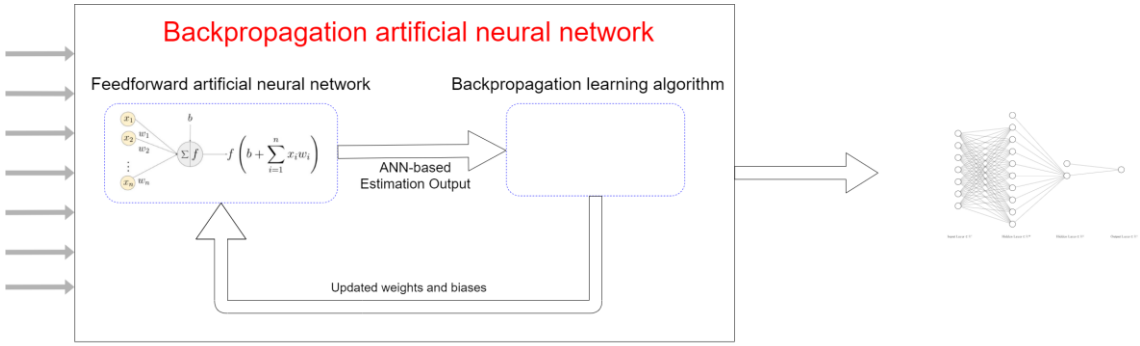


Fig. 31 Backpropagation artificial neural network schematic

For a generic FFANN the output a can be calculated as $a = f(n)$ Fig. 32. Where $f(\cdot)$ is the activation function, and n is net input: $n = \sum_{i=1}^{R} w_j p_j + b = \mathrm{Wp} + b$

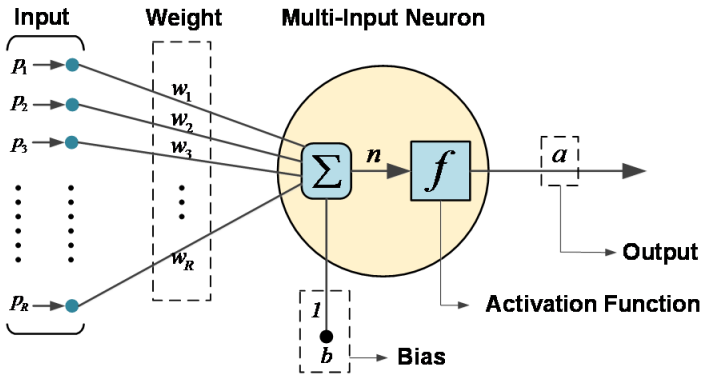


Fig 32 FFANN schematic

The scope of the learning method is to train the network by studying paired set of input-output $\{(\mathbf{p}_1, \mathbf{t}_1), (\mathbf{p}_2, \mathbf{t}_2), \dots, (\mathbf{p}_Q, \mathbf{t}_Q)\}$ with p network input and t the corresponding target output, for this reason backpropagation is called the "supervised learning method". The typical function used for evaluating the algorithm accuracy is the mean square error which is quantified as follow:

$$F(\mathrm{w}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})] \tag{1}$$

Where w is the weights vector. The steepest descendent, Fig. 33, the algorithm is given by:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m} \tag{2}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m} \tag{3}$$

Where $\hat{F}(w) = (t(k)-a(k))^T(t(k)-a(k)) = e^T(k)e(k)$ and α is the learning rate (step size). Exploiting the chain rule it is possible to decompose:

$$\frac{\partial\hat{F}}{\partial w_{i,j}^m} = \frac{\partial\hat{F}}{\partial n_i^m}\cdot\frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{4}$$

$$\frac{\partial\hat{F}}{\partial b_i^m} = \frac{\partial\hat{F}}{\partial n_i^m}\cdot\frac{\partial n_i^m}{\partial b_i^m} \tag{5}$$

Now it is possible to define the sensitivity s_i^m of $\hat{F}$ concerning the i^{th} element of the net input at layer m:

$$s_i^m \equiv \frac{\partial\hat{F}}{\partial n_i^m} \tag{6}$$

Thanks to Eq(6), Eq(4) and Eq(5) can be rewritten as:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{7}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{8}$$

The weight and biases can be calculated by substituting:

$$W^m(k+1) = W^m(k) - \alpha s^m (a^{m-1})^T \tag{9}$$

$$b^m(k+1) = b^m(k) - \alpha s^m \tag{10}$$

Where:

$$s^m \equiv \frac{\partial \hat{F}}{\partial n^m} = \left[\frac{\partial \hat{F}}{\partial n_1^m}, \frac{\partial \hat{F}}{\partial n_2^m}, \dots, \frac{\partial \hat{F}}{\partial n_{s^m}^m}\right]^T \tag{11}$$

Now it necessary to find the recurrence relationship of the sensitivities. This can be done through again applying the chain rule:

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^{m+1}} \frac{\partial n_i^{m+1}}{\partial n_i^m} \tag{12}$$

Where:

$$\frac{\partial n_i^{m+1}}{\partial n_i^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m\left(n_j^m\right) = W^{m+1} \dot{F}^m(n^m) \tag{13}$$

And:

$$\dot{F}^m(n^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}^m(n_2^m) & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m\left(n_{s^m}^m\right) \end{bmatrix} \tag{14}$$

The relation between s^m and s^{m+1} is given by:

$$s^m = \frac{\partial \hat{F}}{\partial n^m} = \left(\frac{\partial n^{m+1}}{\partial n^m}\right)^T \frac{\partial \hat{F}}{\partial n^{m+1}} = \dot{F}^m(n^m)(W^{m+1})^T s^{m+1} \tag{15}$$

This recurrence relation is initialized at the final layer as:

$$s^M = -2\dot{F}^M(n^M)(t-a) \tag{16}$$

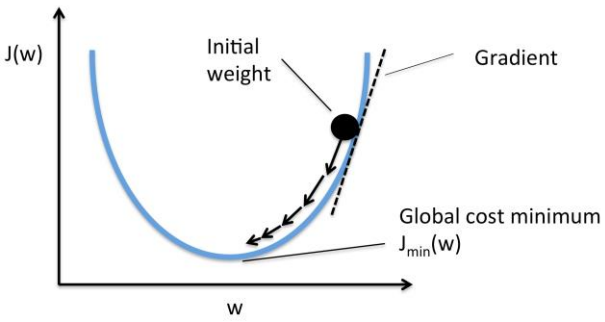


Fig. 33 Steepest gradient descendent by Sebastian Raschka

2.2.2.1 Levenberg-Marquardt (LM)

The Levenberg-Marquardt algorithm is derived from Newton's method which was designed for minimizing functions that are sums of squares of nonlinear functions [48,49]. Newton's method consists in optimizing:

$$w_{k+1} = w_k - A_k^{-1} g_k \tag{17}$$

$$A_k \equiv H(w)|_{w=w_k} \tag{18}$$

$$g_k \equiv G(w)|_{w=w_k} \tag{19}$$

where H is the Hessian matrix and G is the gradient. Assuming $F(w)$ as the sum of squares function:

$$F(w) = \sum_{i=1}^{N} v_i^2(w) = v^T(w)v(w) \tag{20}$$

The respective gradient and Hessian matrix are:

$$G(w) = 2J^T(w)v(w) \tag{21}$$

$$H(w) = 2J^T(w)J(w) + 2S(w) \tag{22}$$

With J(w) defined as Jacobian matrix and S(w):

$$J(w) = \begin{bmatrix} \frac{\partial v_1(w)}{\partial w_1} & \frac{\partial v_1(w)}{\partial w_2} & \cdots & \frac{\partial v_1(w)}{\partial w_n} \\ \frac{\partial v_2(w)}{\partial w_1} & \frac{\partial v_2(w)}{\partial w_2} & \cdots & \frac{\partial v_2(w)}{\partial w_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial v_N(w)}{\partial w_1} & \frac{\partial v_N(w)}{\partial w_2} & \cdots & \frac{\partial v_N(w)}{\partial w_n} \end{bmatrix} \tag{23}$$

$$S(w) = \sum_{i=1}^{N} v_i(w) H v_i(w) \tag{24}$$

Gauss-Newton method ends with:

$$\Delta w_k = -[J^T(w_k)J(w_k)]^{-1}J^T(w_k)v(w_k) \tag{25}$$

The problem of this solution is that the matrix may not be invertible but modifying it with an approximation to the Hessian matrix it is possible to ensure a (local) uniqueness. Among all solutions of F(x), it will pick the one with minimal norm.

$$G = H + \mu I \tag{26}$$

This leads to the Levenberg-Marquardt algorithm:

$$\Delta x_k = -[J^T(w_k)J(w_k) + \mu_k I]^{-1}J^T(w_k)v(w_k) \tag{27}$$

Exploiting the gradient direction and recomputing the approximated performance index, if a smaller value is yield, then μ_k is divided by a factor $\theta > 1$. On the contrary, if the value is not reduced then μ_k is multiplied by θ. In the LM algorithm, we compute the derivative of the error and not its square, as it happens in standard backpropagation. Using the Marquardt concept of sensitivities:

$$\tilde{s}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \tag{28}$$

Elements of Jacobian become:

$$J_{h,i} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m}\frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{s}_{i,h}^m a_{j,q}^{m-1} \tag{29}$$

Or in case of a bias:

$$J_{h,i} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m}\frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{s}_{i,h}^m \tag{30}$$

Once the inputs are applied to the network and found the corresponding output, the LMBP is initialized by:

$$\tilde{s}_q^M = -\dot{F}^M\left(n_q^M\right) \tag{31}$$

Similarly to the normal backpropagation approach, $\tilde{s}_q^m$ can be backpropagated thanks to Eq(15).

2.2.2.2 Bayesian Regularization (BR)

ANNs are a powerful tool for modelling nonlinear functions, but they can suffer from overfitting or overtraining. It is clear that once a model loses its predictability, it will incur in

validation and optimization problems. The main advantages of the Bayesian Regularization algorithm are the robustness and that the validation process is unnecessary [50,51]. Furthermore:

Their models are difficult to be overtrained due to an objective criterion that disables training; the result is that the validation set, used to detect any overtraining, is unnecessary.

Overfitting is settled with the calculation and training on the effective number of parameters which is lower than the number of weights. Essentially, Bayesian Regularization incorporates Occam's razor since it discriminates complex models. The more complex the system is the faster the number of parameters converges to a constant.

The algorithm is based on the Bayes' theorem, called the 'inverse probability law', and the Gauss-Newton approximation to the Hessian matrix which leads to Levenberg-Marquardt algorithm. For the adjustment of the solution Λ-matrix, diagonal and composed of λ elements, is added to H. The new objective function becomes:

$$F(w) = \sum_{i=1}^{N_D} (t_i - a_i)^2 + \lambda \sum_{j=1}^{N_w} w_j^2 \tag{33}$$

Where $0 \leq \lambda \leq 1$; N_D is the number of data, and N_w is the number of weights. This equation can be rewritten in terms of hyperparameters α and β instead of λ:

$$F(w) = \beta \sum_{i=1}^{N_D} (t_i - a_i)^2 + \alpha \sum_{j=1}^{N_W} w_j^2 \tag{34}$$

Once initialized α and β the objective function is minimized by the weights w. Assuming that data and weights probability distributions are Gaussian, the a priori density, for a model M, can be written as :

$$P(\, w \mid \alpha, M\,) = \frac{1}{Z_W(\alpha)} e^{(-\alpha E_W)} \tag{35}$$

$with\ E_W = \sum_{j=1}^{N_W} w_j^2$ being the error of the weights and $Z_W(\alpha) = (\pi/\alpha)^{N/2}$. The likelihood function, which is the probability of the data occurring, given the weights is:

$$P(\, D \mid w, \beta, M\,) = \frac{1}{Z_D(\beta)} e^{(-\beta E_D)} \tag{36}$$

$with\ E_D = \sum_{i=1}^{N_D} [t_i - a_i]^2$ being the error of the data and $Z_D(\beta) = (\pi/\beta)^{n/2}$ The result of substituing the bayesian interference for the weights, inverse probability law, is:

$$P(w|D|\alpha, \beta, M) = \frac{P(D \mid w, \beta, M) P(w \mid \alpha, M)}{P(D \mid \alpha, \beta, M)} = \frac{1}{Z_F(\alpha, \beta)} e^{[-F(w)]} \tag{37}$$

Where $P(D \mid \alpha, \beta, M)$ is a normalization factor, which guarantees that the total probability is 1.

At this point, it's possible to expand $F(w)$ with Taylor about the most probable value of weights w_{MP}:

$$F(w) \approx F(w_{MP}) + \frac{1}{2}(w - w_{MP})^T G(w - w_{MP}) \tag{38}$$

Considering G as the Hessian matrix of the total error function $F(\mathrm{w_{MP}})$ and D the hessian of the data alone:

$$\mathrm{G} = \nabla^2 \mathrm{F}(\mathrm{w_{MP}}) = \beta \nabla^2 E_D(\mathrm{w_{MP}}) + \alpha \mathrm{I} = \beta \mathrm{D} + \alpha \mathrm{I} \tag{39}$$

Substituing 38 and 39 in 37 the distribution of w is close to:

$$P(\,\mathrm{w} \mid D, \alpha, \beta, M\,) \cong \frac{1}{Z_F} \mathrm{e}^{[-F(\mathrm{w_{MP}})]} - \frac{1}{2}(\Delta \mathrm{w}^{\mathrm{T}} \mathrm{G} \Delta \mathrm{w}) \tag{40}$$

where $\Delta \mathrm{w} = \mathrm{w} - \mathrm{w}_{MP}$ and Z_F is a normalizing function.

The optimal weights should maximize the posterior probability $P(\mathrm{w}|D|\alpha, \beta, M)$. That is equivalent to minimize $P(D \mid \alpha, \beta, M)$.

$$\begin{aligned} P(D \mid \alpha, \beta, M) &= \frac{P(\,D \mid \mathrm{w}, \beta, M\,) P(\,\mathrm{w} \mid \alpha, M\,)}{P(\,\mathrm{w} \mid D, \alpha, \beta, H\,)} = \\ &= \frac{\left[\frac{1}{Z_D(\beta)} \mathrm{e}^{(-\beta E_D)}\right]\left[\frac{1}{Z_W(\alpha)} \mathrm{e}^{(-\alpha E_W)}\right]}{\frac{1}{Z_F(\alpha, \beta)} \mathrm{e}^{(-F(\mathrm{w}))}} = \\ &= \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \cdot \frac{\mathrm{e}^{(-\beta E_D - \alpha E_W)}}{\mathrm{e}^{(-\mathrm{F(w)})}} = \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \end{aligned} \tag{41}$$

Even if $Z_F(\alpha, \beta)$ is unknown it can be estimated with Taylor, around $\mathrm{w_{MP}}$ and expanding $F(\mathrm{w})$:

$$Z_F \approx (2\pi)^{N/2} \sqrt{(det((H^{MP})^{-1}))} e^{(-F(w_{MP}))} \tag{42}$$

Replacing Z_F in $P(D \mid \alpha, \beta, H)$ it is possible to solve the optimal values for α and β at $\mathrm{w_{MP}}$. Posing the derivative equals to 0 yields to:

$$\alpha_{MP} = \frac{\gamma}{2E_W(w_{MP})} \tag{43}$$

$$\beta_{MP} = \frac{n - \gamma}{2E_D(w_{MP})} \tag{44}$$

Where $\gamma = \sum_{i=1}^{N_W} \frac{\lambda_i}{\lambda_i + \alpha} = N_P - \alpha trace\,(H^{-1})$ is named as the effective number of parameters and N_P is the total number of parameters in the network. In particular, it quantifies

how many parameters are involved in the reduction of the error function, it can span from 0 to N_P.

2.2.2.3 Resilient Propagation (RPROP)

RPROP algorithm is based on an efficient learning methodology that directly adapts the weights via the local gradient information; in particular, the updates are made only from the signs of partial derivatives, without considering the magnitude [52]. The regulation is made by modifying the weights in opposition to the derivative direction until a local minimum is found in Fig. 34.

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)} & \text{, if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)} & \text{, if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)} & \text{, else} \end{cases} \quad (45)$$

where $0 < \eta^- < 1 < \eta^+$, $\Delta_{ij}^{(t)}$ is the individual update-value, $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i}\frac{\partial s_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}}$ is developed by chain rule where w_{ij} is the weight from neuron j to neuron i, s_i is the output, and net_i is the weighted sum of the inputs of neuron i. If the partial derivative (of the corresponding w_{ij}) changes signs, it means that the update was too large and that it jumped over a local minimum so the update Δ_{ij} is decreased by η^-. On the contrary, if the derivative does not change sign, the updated value must be increased by η^+; in this way, the convergence is accelerated. Every update-value is linked to the weight-update as follows:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & \text{, if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)} & \text{, if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0 & \text{, else} \end{cases} \quad (46)$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \quad (47)$$

When partial derivate changes signs, the minimum is missed, and the new weight-update is restored:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \quad (48)$$

Due to the backtracking, this behaviour is supposed to occur again. To address this issue, it must be set:

$\frac{\partial E}{\partial w_{ij}}^{(t-1)} := 0$ in the updated-values Eq.(47).

The iniziatiliazion of $\Delta_{ij}^{(t)}$, for update-values, is not critical, Martin Riedmiller and Heinrich Braun [52] observed that fast convergence is reached either for big or small values. They suggest the use of η^-=0.5 on the consideration of halving the update-value and η^+ large derived from the trade-off between fast growth of the update-value and learning process disturbance; they found out that η^+=1.2 was a good choice.

The main advantages of this algorithm are the simplicity and high computational efficiency both for time and storage consumption. Furthermore, for common gradient descendent the slope of the activation function (i.e sigmoid) becomes a limiting condition for all weights far from the output layer they are less modified and learn slower. For RPROP, the dependence on the sign and not on the amplitude of the derivative allows an equal chance to grow and learn of the entire network.

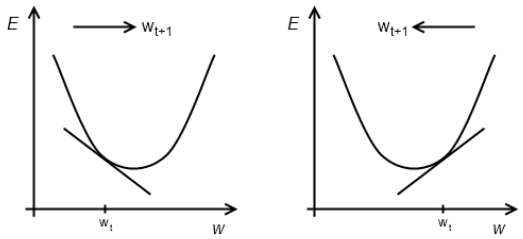


Fig. 34 RPROP behavior [53]

2.2.2.4 Scaled Conjugate Gradient (SCG)

The idea of SCG is to combine the Levenberg-Marquardt algorithm, the classical conjugate gradient approach and second-order estimation used by Hestenes [54, 55]. Using a scaled conjugate gradient the Hessian matrix is always positive for all the iterations. The effect is a quicker convergence concerning other conjugate gradient methods.

The error function of the Taylor series is defined as:

$$E(\tilde{w} + \Delta\tilde{w}) = E(\tilde{w}) + E'(\tilde{w})^T \Delta\tilde{w} + \Delta\tilde{w}^T E''(\tilde{w}) \Delta\tilde{w} + \cdots \quad (49)$$

Supposing the notations $A = E'(\tilde{w})$ and $H = E''(\tilde{w})$; $\tilde{w}_k$ as weight vector in k^{th} iteration. The solution of the quadratic difference equation is $\Delta\tilde{w} = H^{-1}A$, which corresponds to Newton's method outcome. If the Hessian matrix is invertible there is a convergence in one step, but there are some critical issues:

1) When there are many weights to be found, the Hessian calculation is complex and time-consuming.
2) The Hessian matrix may be non-positive.

3) The convergence happens iff the error equation is perfectly quadratic. And generally is not due to the Taylor higher-order terms.

Considering a non zero conjugate weight vectors $p = (p_1, p_2, p_3, \dots, p_N)$ in R^N and are H conjugate, then:

$$\tilde{p}_k^T H \tilde{p}_i = 0 \quad \text{for all } k \text{ and } i \text{ except } k = i$$
$$\tilde{p}_k^T H \tilde{p}_i > 0 \quad \text{for } k = i$$

Defining:

$$\sigma_k = \frac{\sigma}{|\tilde{p}_k|} \tag{50}$$

$$\tilde{s}_k = \frac{E'(\widetilde{w}_k + \sigma_k \tilde{p}_k) - E'(\widetilde{w}_k)}{\sigma_k} \tag{51}$$

$$\delta_k = \tilde{p}_k^T \tilde{s}_k \tag{52}$$

Where $0 < \sigma_k \ll 1$, and $\tilde{s}_k$ is standard Conjugate Gradient case. Hestene introduces a non-symmetric approximation:

$$s_k = \frac{E'(\widetilde{w_k} + \sigma_k \tilde{p}_k) - E'(\widetilde{w_k})}{\sigma_k} + \lambda_k \tilde{p}_k \tag{53}$$

where parameter λ can be considered as a dumper, used in case of negative H. Each iteration δ_k sign is checked, when $\delta_k \leq 0$ λ_k is increased and vice versa. The objective is to find a set of weight vector for which H become positive definite very near to zero. The equations are:

$$\widetilde{w}_{k+1} = \widetilde{w}_k + \alpha_k \tilde{p}_k \tag{54}$$

$$\tilde{p}_{k+1} = \tilde{r}_k + \beta_k \tilde{p}_k \tag{55}$$

With $r_k = -E'(w_k)$, and α_k and β_k are calculated iteratively by:

$$\alpha_k = \frac{\tilde{p}_k^T \tilde{r}_k}{\tilde{p}_k^T H \tilde{p}_k} = \frac{\tilde{p}_k^T \tilde{r}_k}{\delta_k} \tag{55}$$

$$\beta_k = \frac{\langle \tilde{r}_{k+1}, \tilde{r}_{k+1} \rangle - \langle \tilde{r}_{k+1}, \tilde{r}_k \rangle}{\tilde{p}_k^T H \tilde{p}_k} \tag{56}$$

For example, in the case $\delta_k \leq 0$, $\lambda_k$must be increased by $\bar{\lambda}_k$ so that $\tilde{s}_k$ and δ_k are updated:

$$\tilde{\tilde{s}}_k = \tilde{s}_k + (\bar{\lambda}_k - \lambda_k)\tilde{p}_k \tag{57}$$

$$\widetilde{\delta_k} = \tilde{p}_k^T \tilde{\tilde{s}}_k \tag{58}$$

Substituting $\tilde{\tilde{s}}_k$ in $\widetilde{\delta_k}$ it must be satisfied the condition that $\widetilde{\delta_k} > 0$:

$$\widetilde{\delta_k} = \delta_k + (\bar{\lambda}_k - \lambda_k)|\tilde{p}_k|^2 > 0 \tag{59}$$

$$\Rightarrow \bar{\lambda}_k > \lambda_k - \frac{\delta_k}{|\tilde{p}_k|^2} \tag{60}$$

A good solution is to choose $\bar{\lambda}_k = 2(\lambda_k - \frac{\delta_k}{|\tilde{p}_k|^2})$. The resulting $\widetilde{\delta_k}$ is:

$$\widetilde{\delta_k} = \delta_k + (\bar{\lambda}_k - \lambda_k)|\tilde{p}_k|^2 = \delta_k + \left(2\lambda_k - 2\frac{\delta_k}{|\tilde{p}_k|^2} - \lambda_k\right)|\tilde{p}_k|^2 = \\ = -\delta_k + \lambda_k|\tilde{p}_k|^2 > 0 \tag{61}$$

The step size α is:

$$\alpha_k = \frac{\mu_k}{\delta_k} = \frac{\mu_k}{\tilde{p}_k^T\tilde{s}_k + \lambda_k|\tilde{p}_k|^2} \tag{62}$$

$$\mu_k = \tilde{p}_k^T\tilde{r}_k \tag{63}$$

This last passage highlights the inverse dependence of the step size and λ_k; the choice of lambda is important because the rate of convergence may be slow. For this purpose rho is introduced, which measures the deviation of the second approximation of the error curve and the original one:

$$\rho_k = \frac{E(\tilde{w}_k) - E(\bar{w}_k + \alpha_k\tilde{p}_k)}{E(\tilde{w}_k) - E_{qw}(\alpha_k\tilde{p}_k)} = \frac{2\delta_k[E(\bar{w}_k) - E(\bar{w}_k + \alpha_k\tilde{p}_k)]}{\mu_k^2} \tag{64}$$

$$\text{if } \rho_k > 0.75, \text{ then } \lambda_k = \frac{1}{4}\lambda_k$$

$$\text{if } \rho_k < 0.25, \text{ then } \lambda_k = \lambda_k + \frac{\delta_k(1 - \Delta_k)}{|\tilde{p}_k|^2}.$$

The case of $\rho_k = 1$ means best approximation. Møller showed that if delta is kept small it won't adversely affect overall performance [55].

2.2.3 Performance measures

2.2.3.1 Root Mean Squared Error (RMSE)

Describes a measure of the average spread of errors:

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(f_n - r_n)^2}$$

With f_n the forecast value of the observed r_n. $RMSE \cong 0$ means that the forecasting model is very close to the reality.

2.2.3.2 Mean Absolute Error (MAE)

Considers only the absolute value of the error whose smaller values indicate better forecasts.

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|f_n - r_n|$$

2.2.3.3 Mean Bias Error (MBE)

Compares the variance of the errors to the variance of the data. A positive bias corresponds to an overestimation of the dataset.

$$MBE = \frac{1}{N}\sum_{n=1}^{N}(f_n - r_n)$$

2.2.3.4 Mean Absolute Percentage Error (MAPE)

Başar et al, have evaluated that $\text{MAPE} \leq 10\%$ means high prediction accuracy, $10\% \leq MAPE \leq 20\%$ means good prediction, $20\% \leq MAPE \leq 50\%$ means reasonable prediction and a $MAPE \geq 50\%$ means inaccurate forecasting [56].

$$MAPE = \frac{100\%}{N}\sum_{n=1}^{N}\left|\frac{f_n - r_n}{r_n}\right|$$

2.2.3.5 Correlation Factor (R)

$$R = \frac{\frac{1}{N}\sum_{n=1}^{N}(f_n - \bar{f})(r_n - \bar{r})}{\sigma_f \sigma_r}$$

Where σ_f and σ_r are the standard deviations of f and r, respectively.

2.2.3.6 R-squared

Is the square of the correlation between predicted and actual data. It ranges from 0 to 1; $R^2 = 1$ means high correlation, on the contrary, 0 means non-correlated data.

$$R^2 = \frac{\left(\sum_{i=1}^{N}(f_n - \bar{f})(r_n - \bar{r})\right)^2}{\sum_{n=1}^{N}(f_n - \bar{f})^2 \times \sum_{n=1}^{N}(r_n - \bar{r})^2}$$

2.2.3.7 Taylor diagram

The Taylor diagram is used for graphical identifications of predicting models [57]. The analysis starts considering the RMSE:

$$E = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(f_n - r_n)^2}$$

as two components. The overall bias:

$$\bar{E} = \bar{f} - \bar{r}$$

And the centred pattern RMS difference:

$$E' = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[(f_n - \bar{f}) - (r_n - \bar{r})\right]^2}$$

All three linked with:

$$E^2 = \bar{E}^2 + E'^2$$

For the comparisons of different patterns are considered $R, E', \sigma_f, and\ \sigma_r$ which are linked by:

$$E'^2 = \sigma_f^2 + \sigma_r^2 - 2\sigma_f\sigma_r R$$

This is the definition of the cosine law Fig. 35:

$$c^2 = a^2 + b^2 - 2ab\cos\,\phi$$

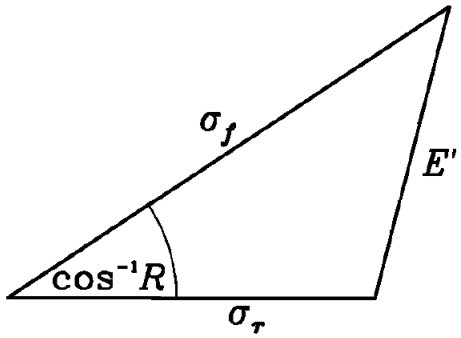


Fig. 35 Geometric relationship between the correlation coefficient R, the centered pattern RMS error E', and the standard deviations σ_f and σ_r of the test and reference fields, respectively

Thanks to this relation it is possible to quantify the degree of similarity between a reference model and test ones. The closer is the test the better is the prediction. The diagram is

composed of two orthogonal axes, which express the standard deviation, and the correlation coefficient expressed in azimuthal position Fig.36.

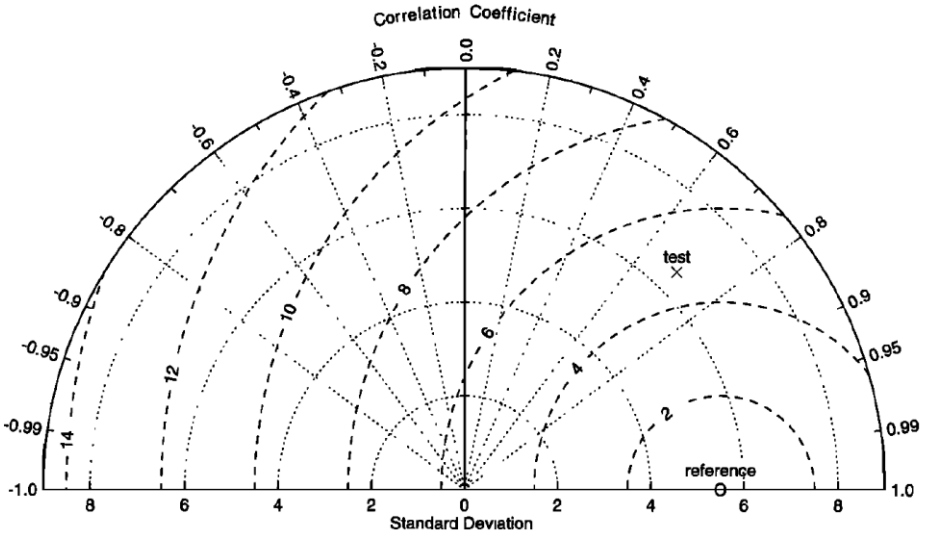


Fig. 36. The radial distance from the origin is proportional to the standard deviation of a pattern. The centred RMS difference between the test and reference field is proportional to their distance apart (in the same units as the standard deviation). The correlation between the two fields is given by the azimuthal position of the test field

2.3 ANNs for the environment

In the last few decades, there has been an-increase in the development of ANNs and growth in specialized employment in the sector of weather forecasting. Hsieh, W.H and B. Tang, 1998, studied various ANN prediction models for the meteorology and oceanography fields and they concluded that this technique was effective and useful [58]. In the same year W. DAWSON & ROBERT WILBY investigated a rainfall-runoff model for *Rivers Amber and Mole.* The ability of ANNs to learn to cope effectively with missing data had a big impact on their work, making conventional methods outdated [59]. K. C. Luk et al, 2001, studied and compared the application of three types of ANNs for rainfall forecasts; the researchers observed that the '*15-min rainfall time'* series have a short term memory and that networks with a lower lag were more accurate [60]. Chattopadhyay and Chattopadhyay, 2008, studied the number of layers of a BPANN for the best prediction of monsoon rainfall in India; the results showed that a hidden layer with eleven nodes was more accurate in respect to the other layers tested [61]. Singh et al., 2013, have investigated the potential of BPANNs for the Indian summer monsoon rainfall with respect to statistical models. They concluded that ANNs are simpler, more robust, and the learning phase is very fast; according to them using the technology of ANNs can be useful for predicting future droughts and floods [62].

Tsong-Lin Lee, 2004, compared BPN and the conventional harmonic method for studying tidal patterns. He discovered that ANNs could predict a yearly tidal level with only 15 days of observational data [63]. In 2005, Chaudhuri et al. studied an efficient FFNN with one hidden layer that was used for estimating the maximum surface temperature and relative humidity in Calcutta. The analysis aimed to predict thunderstorm 24hrs in advance and to provide a reconstruction model of the missing data [64]. Jon Vandegriff et al. developed a system to predict the arrival time of interplanetary (IP) shock to Earth with ANNs. Forecasting capabilities were promising and much better in respect to existing methods [65]. Bustami Rosmina et al., 2007, faced the absence of precipitation readings with BPANN in his estimation of both precipitation and water levels of Bedup River and the corresponding region. The analysis showed that the ANN yielded positive results in terms of accuracy; 96.4% and 85.3% respectively [66]. Mohsen Hayati, and Zahra Mohebi have used the multi-layer perceptron network for the design of short-term temperature forecasting (STTF) systems for Kermanshah city in Iran. The net resulted to have good performances and reasonable prediction accuracy for one day ahead [67]. Haghizadeh et al., 2010, have proposed an ANN model for the prediction of sediment yield since measuring concentrated sediment from a river or sediment transport equations is expensive and complex. The outcome underlined the capability of ANN to couple with nonlinear problems to produce a good estimation of the sediment yield [68]. Pan et al., 2011, have experimented with a FFNN to improve rainfall statistic forecasting models after typhoon Morakot (2009) produced a rainfall record of 2361mm in 2days. The new method produced a robust improvement of the typhoon rainfall climatology model (TRCM) [46]. Mohan Raju et al., 2011, have compared ANNs with Levenberg Marquardt's algorithm and Linear Multiple Regression Models for predicting the weekly spring discharge in a district of Uttarakhand. Taking into account the lower error rate, the researchers concluded that ANN models performed better [69]. In 2012, Sawaitul et al., have discussed the Back Propagation Algorithm in weather forecasting whose data was recorded by a satellite and sensors. They observed that ANNs are an effective tool for the prediction and classification of thunderstorms, moreover, increasing the input makes estimations even more accurate [70]. In 2014, Amit Kumar Yadav and S.S. Chandel have reviewed different studies on solar radiation prediction with ANN models. The researchers pointed out the importance of determining the radiation level for solar system design and power generation, moreover this prediction is valuable for both researchers and power plant installers to determine solar radiation in sites without meteorological stations. Many different input parameters can be considered in the design of the ANN model, in particular, sunlight

hours and air temperature showed a correlation coefficient of 97.65% [71]. In 2018, Feng et al., have estimated the Monthly-Mean Daily Global Solar Radiation using: MODIS Atmospheric Products [72]. The study was based on clouds, aerosols and precipitable water-vapour which feed a back propagation neural network using the Levenberg-Marquardt algorithm. The model was highly accurate in terms of correlation and errors, and also exhibited sound stability in respect to the multiple linear regression (MLR) model, and remotely-sensed radiation products by Cloud and the Earth's Radiation Energy System (CERES) [73]. Archana Sarkara and Prashant Pandey have modelled Yamuna river water quality with a Feed forward back propagation neural network that studied dissolved oxygen concentrations. The network performed well in short time base input data, 10-daily or daily data, without any assumptions on the range of flow discharge, temperature, biochemical oxygen demand and dissolved oxygen [74]. In 2019, Maleki et al. have developed an ANN for predicting the air pollution in Ahvaz; They used six pollutants (O3, NO2, PM10, PM2.5, SO2, and CO) for quantifying the reliability of the model and the Air Quality Index (AQI) and Air Quality Health Index (AQHI) for its applicability. The outcome proposes that the ANN could be a good alternative to air pollution spatial interpolation and it could eliminate the numerous monitoring stations around the city [75]. In 2020, Ballestrín et al. have modelled solar extinction, caused by scattering and absorption phenomena in between solar tower receiver and heliostats, using Levenberg-Marquardt's algorithm back propagation neural network. The model, fed with aerosol particles and water vapour as input, produced a good result in terms of accuracy with a Mean Bias Deviation of 0.03% and a Normalize Root Mean Square Error of 5.45% [76]. Mosaffaei et al. have developed an ANN for studying the soil and plant degradation of Sorkheh Hesar National Park in Iran. The necessity to protect and preserve natural areas is crucial for ecosystems: soil degradation reduces water storage capacity and germination in the soil, which ultimately leads to the destruction of plants. The results showed that their model responded well to criteria implemented such as distance from the road, slope and the geographical aspect [77].

3 Materials and Methods

3.1 Solar Parabolic Dish

The Solar Parabolic Dish Fig. 37 is composed of an aluminium paraboloid that is completely coated by a polymeric film, characterized by a high reflection efficiency. The plant comprises of an automatic solar tracking system with two independent, axes, azimuth and elevation, for calibrating its position concerning the sun to maximize solar radiation with the best angle of incidence; the entire software is developed and implemented by El.Ma. Srl. The dish changes its orientation in real-time, and it functions by calculating the hours, the date, latitude and longitude, with maximum theoretical accuracy (>0.015°). It is possible to steer two different dishes by controlling one axis meanwhile the other is kept fixed, e.g. elevation constant - azimuth variable and vice versa. All the specifics are reported in Table 1.

Fig. 37 El.Ma. Electronic Machining S.r.l. Solar Parabolic Dish (SPD)

Caratteristiche tecniche generali

- Temperatura min. di esercizio: 0° C
- Temperatura max di esercizio: + 35°C
- Umidità relativa (non condensata): 95%
- Altitudine max: 3000 m
- Livello di pressione acustica: vedere par. 2.7.1
- Vibrazioni: vedere par. 2.7.2
- Dimensioni di ingombro massimo: 3200 x 2800 x H3500 mm

Impianto elettrico

- Tensione di alimentazione: 220 V
- Frequenza: 50 Hz
- Potenza totale installata: vedere schema elettrico
- Potenza totale assorbita: vedere schema elettrico

Table 1 SPD specifics

3.2 Thermocouple Type B

Thermocouples are one of the simplest sensors. The circuit is made up of at least two junctions: one, hot, used for the measurements, and the second, cold, used as a reference. A voltage, based on the Seeback effect, is generated once a difference in temperature between the two is detected. Thermocouples of type B are made of platinum and rhodium alloy; the positive material is composed of 70% Pt and 30% Rh, meanwhile the negative material has 94% Pt and 6% Rh. The accuracy class 2 is 0.5% for temperatures greater than 800°C and the working range spans from 50°C to 1820 °C; the transfer function is nonlinear in the working interval and it is constantly under 50°C. Data are sampled every minute on different days from December 2019 to March 2020

3.3 Weather Station

It measures seven meteorological conditions every 15 minutes. The station is composed of an hygrometer for humidity $[\%]$, a thermometer for temperature $[°C]$, a barometer for atmospheric pressure $[mbar]$, an anemometer for wind velocity $[\frac{m}{s}]$, a wind vane for wind direction [°], a pyranometer for global radiation $[\frac{W}{m^2}]$, and a rain gauge for rain $[mm]$. For accuracy issues temperature and humidity measurements are kept free from direct solar radiation. Personal weather station involves a digital console that provides an excel readouts of the data being collected. These files are interfaced to a personal computer where data are displayed, stored, and used.

3.4 ANNs configuration

The BPANN used to achieve the thesis objective is reported in Fig. 38. The first stage is devoted to the Feedforward artificial neural network which is fed with the weather station atmospheric conditions. This network is composed of one input layer with seven neurons, one or two hidden layers with a variable number of neurons, and one output layer with one output neuron. The second stage is constituted by a learning algorithm that is intended to update weights and biases. The final result is a BPANN based on SPD temperature Fig. 38.

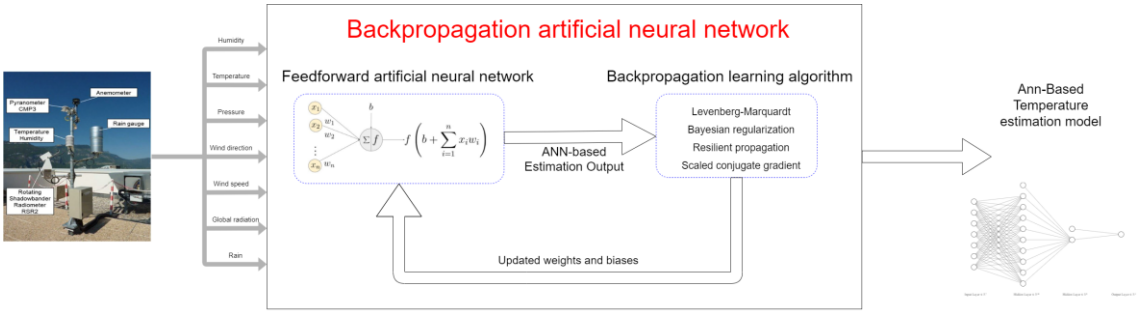


Fig. 38 BPANN schematic

The study of this thesis is conducted via MATLAB 2020b with the Deep Learning Toolbox. In this environment data gathered from sensors are compiled into excel files. Any user can implement their analysis by generating an algorithm once the user filled two proper folders with all data collections. For this study, Meteo Data and SPD Data are used as folders. The code begins by scanning both folders for ".xlsx" file extension and creates the list of data, respectively. Fig. 39, Fig. 40. Furthermore, a first sifting was implemented for the data of the tower according to thermocouple type B. For this purpose, every single file is scanned to look for all temperature values in between 50°C and 1800 °C. All the other temperatures are not considered. In the event, a file contains temperatures outside of the working range it is skipped, and the final structure of the dish data list is reshaped Fig. 40.

```
%% Loading Meteo
Base = 'C:\Users\leont\OneDrive\Desktop\SCRIPT TESI Marzo\Dati Meteo';
List = dir(fullfile(Base, '*.xlsx'));
Result = cell(1, numel(List));
datetime.setDefaultFormats('default','dd/MM/yyyy HH:mm:ss')
for k = 1:numel(List)
    File   = fullfile(Base, List(k).name);
    MeteoResult = readtable(File,"VariableNamingRule",'preserve');
    MeteoDate(k)= MeteoResult.Data(1)
    StartingTimeMeteo(k)=MeteoResult.UTCTime(1);
    EndingTimeMeteo(k)=MeteoResult.UTCTime(end);
    MeteoResult=table2timetable(MeteoResult(:,3:10));
    MeteoDailyFile{k}=MeteoResult;
end
```

Fig. 39 Meteo data importing

```
%% Loading Tower
Base = 'C:\Users\leont\OneDrive\Desktop\SCRIPT TESI Marzo\Dati Tower';
List = dir(fullfile(Base, '*.xlsx'));
Result = cell(1, numel(List));
Nanflag=0;
for k = 1:numel(List)
    File   = fullfile(Base, List(k).name);
    TowerResult = readtable(File,"VariableNamingRule",'preserve');
    TowerDate(k)=TowerResult.Date(1)
    StartingTimeTower(k)=TowerResult.UTCTime(1);
    EndingTimeTower(k)=TowerResult.UTCTime(end);
    TowerDailyFile{k}=TowerResult;
    p=1;
    for l=1:height(TowerDailyFile{k})
        if ((TowerResult.Treatt_0(l)>50)&(TowerResult.Treatt_0(l)<1800))
            NewTowerResult(p,:)=TowerResult(l,:);
            p=p+1;
        end
    end
    if (p~=1)
        NewTowerDailyFile{k-Nanflag}=table2timetable(NewTowerResult(:,3:12));
        clear NewTowerResult
    else
        Nanflag=Nanflag+1;
    end
end
```

Fig. 40 Dish data importing and sifting

Once all the data are correctly imported and filtered, a '*for loop*' searches for the dates with accessible data. Fig.41. The contextualization of data, in terms of time, initiates with highlighting the first and the last day of collection. In this interval, '*for loops*' scan and count which days both meteo and tower data are available and which days are not. For greater comprehension, the result is plotted in a graph Fig.42 in which '1' corresponds to the availability of data and '0' with the absence of data. Meteo is denoted by a blue circle and a tower with a red cross.

```
%% Date Measurements
if (MeteoDailyFile{1}.Time(1)<TowerDailyFile{1}.Time(1))
    StartingDay=MeteoDailyFile{1}.Time(1);
    MeteoFlag=1;
else
    StartingDay=TowerDailyFile{1}.Time(1);
end
if (MeteoDailyFile{end}.Time(1)<TowerDailyFile{end}.Time(1))
    EndingDay=MeteoDailyFile{end}.Time(1);
else
    EndingDay=TowerDailyFile{end}.Time(1);
end
DaysCounter=round(days(EndingDay-StartingDay));
MeteoDayMeasurements(DaysCounter)=0;
TowerDayMeasurements(DaysCounter)=0;
if MeteoFlag==1
    MeteoDayMeasurements(1)=1;
else
    TowerDayMeasurement(1)=1;
end
for i=1:DaysCounter
    for j=1:length(MeteoDate)
        if datetime(MeteoDate(j))==StartingDay+i
            MeteoDayMeasurements(i+1)=1;
        end
    end
    for h=1:length(TowerDate)
        if datetime(TowerDate(h))==StartingDay+i
            TowerDayMeasurements(i+1)=1;
        end
    end

end
figure
plot(StartingDay:caldays(1):EndingDay,MeteoDayMeasurements,'ob');hold on;grid on
plot(StartingDay:caldays(1):EndingDay,TowerDayMeasurements,'xr');
```

Fig. 41 Accessible data analysis

Fig. 42 Date plot of accessible data

Now it is necessary to create two newer lists that contain only the '1' set of pairs meteo-tower. To achieve this objective a nested loop, capable of comparing the date of acquisition was used, along with a simple counter, which increases if the previous condition holds Fig.43.

```
%% Data consistency
n=1;
for i=1:NO_METEODAYS
    for j=1:NO_TOWERDAYS
        if MeteoDate(i)==TowerDate(j)
            M_DATA{n}=MeteoDailyFile{i};
            M_END(n)=EndingTimeMeteo(i);
            T_DATA{n}=NewTowerDailyFile{j};
            T_END(n)=EndingTimeTower(j);
            T_START(n)=StartingTimeTower(j);
            n=n+1;
        end
    end
end
```

Fig. 43 Common date structures creation

The overall sum of meteo data (sampled every fifteen minutes), tower data (sampled every minute), and thermocouple filter brings to a tiny collection of data that can be used to feed ANN. For this purpose, meteo data have been stretched. For each variable, every quarter of an hour is divided into an interval of fifteen, equally spaced points Fig. 44. The outcome is a zoomed out variable with up to 1425 points. Fig. 45 reports the relative humidity on the first date. For obvious reasons, the last quarter of an hour, from 23:45 to 00:00, has not been considered in the subdivision.

```
%% Creating Data for Meteo (extending)
for i=1:n-1
    NORMALIZED_VAR=M_DATA{i}.Variables;
    NM=zeros(15*(size(NORMALIZED_VAR,1)-1),size(NORMALIZED_VAR,2));
    UTCTimeMeteoExtended=zeros(size(NORMALIZED_VAR,1),1);
    cnt_minutes=1;
    for k=1:size(NORMALIZED_VAR,1)-1
        for j=1:size(NORMALIZED_VAR,2)
            C(:,j)=linspace(NORMALIZED_VAR(k,j),NORMALIZED_VAR(k+1,j),15);
        end
        NM([cnt_minutes:15*k],:)=C;
        cnt_minutes=cnt_minutes+15;
    end
    UTCTimeMeteoExtended=linspace(0,(cnt_minutes-1)/60/24,cnt_minutes-1)';
    EXTENDED_METEO_DATA{i}=[UTCTimeMeteoExtended NM];
end
```

Fig. 44 Meteo data equally spaced with one minute

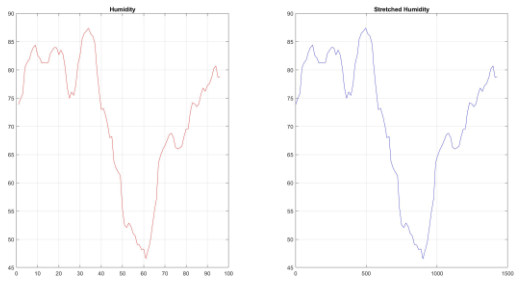


Fig. 45 Humidity example

At this point the user has two structures of files characterized by the same date, the same step size of one minute; it is noted, however, for reasons already stated, that there are a higher number of meteo samples than tower samples. Based on this information an algorithm is developed to establish the interval of time in which the couple meteo-tower is in working condition. Every day, the research begins with the identification of the "ending" time for both the meteo and the tower; subsequently, it is necessary to find the starting time of the tower through another loop Fig. 46. Every minute of the meteo is compared with starting time of the tower until their difference is lower than $3e-04$ (highlighted in red). This value comes from a first calculation of a minute's weight in a day $(24*60)^{-1} = 1440^{-1} = 6.9444e-04$, and then it is decreased for tolerance Fig. 47. To increase the reliability and robustness of the '*if statement*' two cases are analyzed: the first one corresponds to the "regular" case, in which all the tower times are correctly paired with meteo times Fig. 48 (meteo times number higher than tower times number); the second case, as stated, is implemented for security reasons. In this instance, the ending time point does not come from the dish (tower) but instead comes from the meteo Fig. 49. There was not any evidence of this occurring in this study.

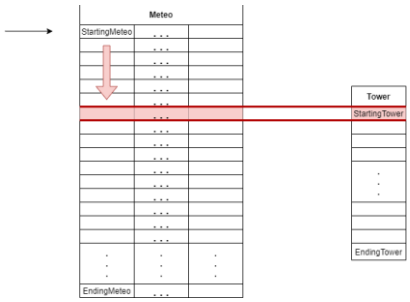


Fig. 46 Seeking of starting times

```
%% Preparing Data for NET (consistency of Tower and Meteo data per minute)
for i=1:length(T_DATA)
    DataMeteoExtended=EXTENDED_METEO_DATA{i}(:,2:end);
    UTCTimeMeteoExtended=UTCTimeMeteoExtended(:,1);
    EndingSamples=min(UTCTimeMeteoExtended(end),T_DATA{i}.UTCTime(end));
    for j=1:height(EXTENDED_METEO_DATA{i})
        if (abs(UTCTimeMeteoExtended(j)-T_START(i))<3e-04)
            if (ismember(EndingSamples,T_DATA{i}.UTCTime))
                    readyMETEOdata=zeros(size(T_DATA{i},1),size(EXTENDED_METEO_DATA{i},2));
                    readyMETEOdata=DataMeteoExtended(j:height(T_DATA{i})+j-1,:);
                    readyTOWERdata=T_DATA{i}.Treatt_0;
                    LOLTOWERConfiguration=T_DATA{i}.Configuration;
            else
                    readyMETEOdata=zeros(size(EXTENDED_METEO_DATA{i},1)-j+1,size(EXTENDED_METEO_DATA{i},2));
                    readyMETEOdata=DataMeteoExtended(j:end,:);
                    readyTOWERdata=T_DATA{i}.Treatt_0(1:length(DataMeteoExtended)-j+1);
                    LOLTOWERConfiguration=T_DATA{i}.Configuration(1:length(DataMeteoExtended)-j+1);
            end
        end
    end
    netDATAMETEO{i}=readyMETEOdata;
    netDATATOWER{i}=readyTOWERdata;
end
```

Fig. 47 Condition of correct starting

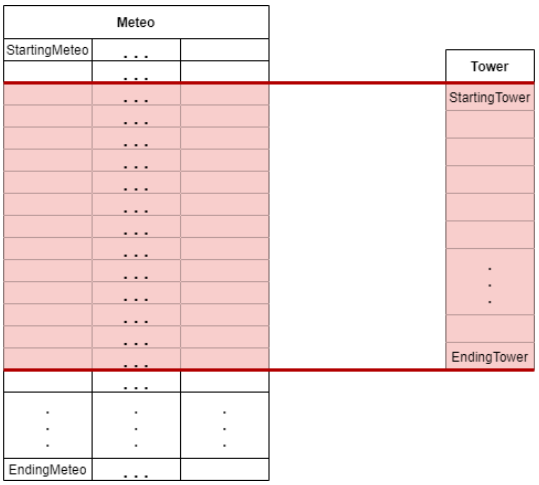


Fig. 48 Dish ending contained in meteo times

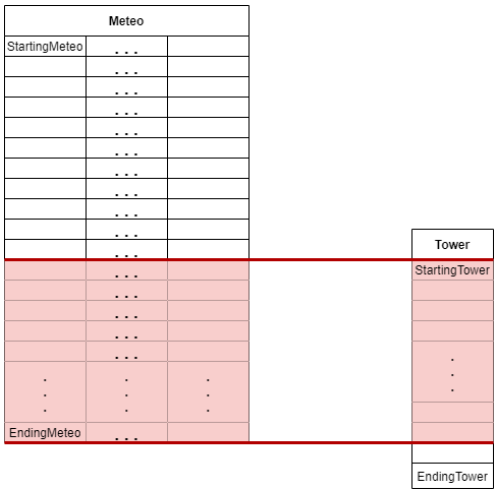


Fig. 49 Dish ending not contained in meteo times

Before training the ANNs all data are merged Fig. 50 and then normalized in [-1,1]:

$$x = 2\frac{(x - min(x))}{max(x) - min(x)} - 1$$

```
%% Data net
xt=netDATAMETEO{1}';
yt=netDATATOWER{1}';
 for i=2:length(netDATAMETEO)
    xt=catsamples(xt,netDATAMETEO{i}');
    yt=catsamples(yt,netDATATOWER{i}');
 end
% NORMALIZATION OF ALL INPUT IN [-1,1]
 for j=1:7
    xt(j,:)=2*((xt(j,:)-min(xt(j,:)))./(max(xt(j,:))-min(xt(j,:))))-1;
 end
```

Fig. 50 Data normalization in [-1,1]

The first BPANN in consideration is the one trained with Levenberg-Marquardt Fig. 51, Fig. 52. Data are divided in three data sets: training (70% = 15516 samples), validation (15% = 3325 samples), and test (15% = 3325 samples). In the first cycle, the indexes division is made randomly, and successive cycles are uploaded with the global variables *tInd, vInd, teInd*. The first activation function is a hyperbolic tangent that occurs from the input to the hidden layer, the second function is a pure line that operates from the hidden layer to the output layer Fig. 53.

```
%% Levenberg-Marquardt with 1 hidden layer
NUMBERS='';
 for i=1:N
hiddenLayer1Size=i;
net=fitnet(hiddenLayer1Size);
if i==1
net.divideParam.trainRatio= 70/100; %used to learn
net.divideParam.valRatio= 15/100; %used to optimize the internal parameters of the model that minimize error of validation
net.divideParam.testRatio= 15/100; %used as test because we don't want to reuse the validation data
end
if i~=1
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
end
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainlm(net,xt,yt); %tr=info
if i==1
    tInd=tr.trainInd; % training data indices
    vInd= tr.valInd; % validation data indices
    teInd= tr.testInd;  % testing data indices
end
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainLevenberg_Marquardt1L(i)=mean(yTrain-yTrainTrue);
MAEtrainLevenberg_Marquardt1L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainLevenberg_Marquardt1L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue));
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
```

```
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainLevenberg_Marquardt1L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainLevenberg_Marquardt1L(i)=sqrt(mean((yTrain-yTrainTrue).^2))
centeredtrainRMSE1L(i)=sqrt(mean((yTrain-mean(yTrain)-yTrainTrue+mean(yTrainTrue)).^2))
OverallBiastrainRMSE1L(i)=mean(yTrain)-mean(yTrainTrue);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalLevenberg_Marquardt1L(i)=mean(yVal-yValTrue);
MAEvalLevenberg_Marquardt1L(i)=mean(abs(yVal-yValTrue));
MAPEvalLevenberg_Marquardt1L(i)=mean(abs((yVal-yValTrue)/yValTrue));
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalLevenberg_Marquardt1L(i)=Rval(i)^2;
RMSEvalLevenberg_Marquardt1L(i)=sqrt(mean((yVal-yValTrue).^2))
centeredvalRMSE1L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2))
OverallBiasvalRMSE1L(i)=mean(yVal)-mean(yValTrue);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestLevenberg_Marquardt1L(i)=mean(yTest-yTestTrue);
MAEtestLevenberg_Marquardt1L(i)=mean(abs(yTest-yTestTrue));
MAPEtestLevenberg_Marquardt1L(i)=mean(abs((yTest-yTestTrue)/yTestTrue));
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestLevenberg_Marquardt1L(i)=Rtest(i)^2;
RMSEtestLevenberg_Marquardt1L(i)=sqrt(mean((yTest-yTestTrue).^2))
centeredtestRMSELM1L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2))
OverallBiastestRMSELM1L(i)=mean(yTest)-mean(yTestTrue);
 end
```

Fig. 51 LM 1 HL Matlab code

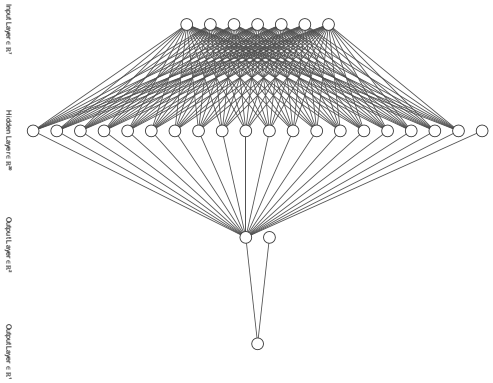


Fig. 52 LM 1 HL architecture.

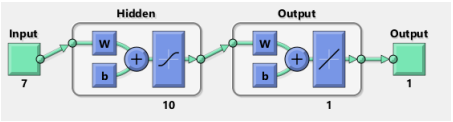


Fig. 53 LM 1 HL Matlab architecture

The training process stops when any of these conditions occur:

- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min_grad.
- mu exceeds mu_max.

- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

Once the training phase has finished and so a model is found, RMSE, MAE, MAPE, MBE, standard deviations, R, R^2, time, and epochs are calculated. These last two stopping conditions are left by *default* set to *inf* and *1000.* The first comparative analysis is carried out with the Taylor diagram.

The second BPANN is equivalent to the first but with two hidden layers Fig. 54; the second hidden layer is composed of three times the number of neurons of the first layer. Fig. 55. The activation functions are hyperbolic, hyperbolic and pure line, respectively Fig. 56.

```
%% Levenberg-Marquardt with 2 hidden layer
NUMBERS='';
for i=1:N
hiddenLayer1Size=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
net.layers{3}.transferFcn='purelin';
[net,tr]=trainlm(net,xt,yt); %tr=info
```

Fig. 54 LM 2 HL Matlab code

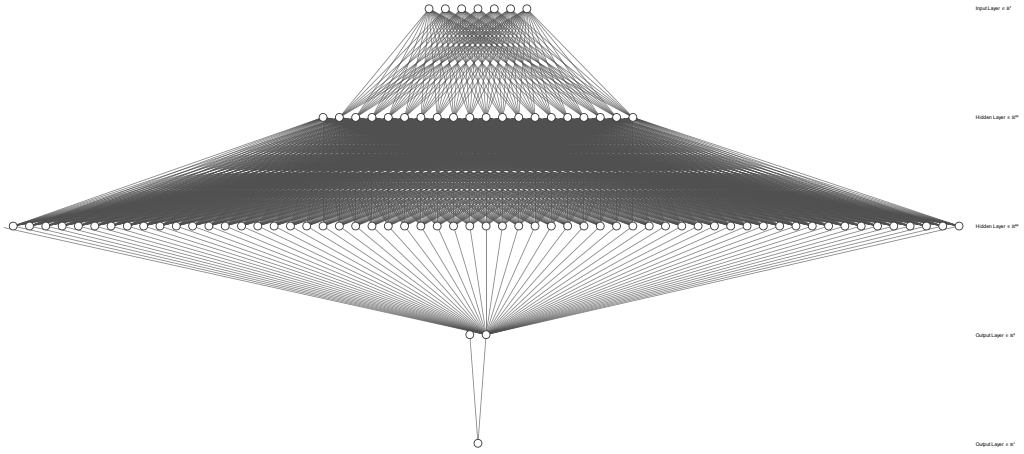


Fig. 55 LM 2 HL architecture

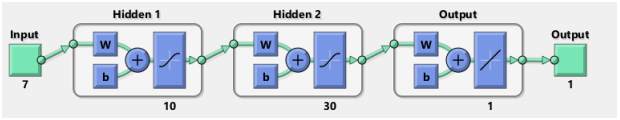


Fig. 56 LM 2 HL Matlab architecture

The third BPANN is trained with Bayesian regularization Fig. 57. Due to its capacity of non-validation, data was divided into 70% = 15516 samples for training and 30% = 6650 for testing. The activation functions used were hyperbolic and pureline Fig. 58.

```
%% Bayesian regularization  1 HL
NUMBERS='';
for i=1:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.testInd= sort([teInd vInd]);  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainbr(net,xt,yt); %tr=info
```

Fig. 57 BR 1 HL Matlab code

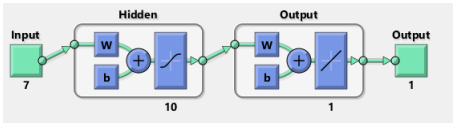


Fig. 58 BR 1 HL Matlab architecture

Training stops when any of these conditions occur:

- The maximum number of epochs is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min_grad.
- mu exceeds mu_max.

The fourth BPANN is equivalent to the previous one but with two hidden layers Fig. 59; the second hidden layer is composed of three times the number of neurons of the first hidden layer. The activation functions are hyperbolic, hyperbolic, and pure line, respectively Fig. 60.

```
%% Bayesian regularization with 2 Hidden
NUMBERS='';
for i=1:N
hiddenLayer1Size=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.testInd= sort([teInd vInd]);  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
net.layers{3}.transferFcn='purelin';
[net,tr]=trainbr(net,xt,yt); %tr=info
```

Fig. 59 BR 2 HL Matlab code

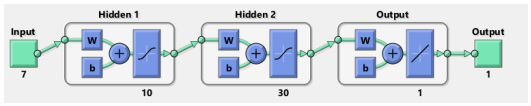


Fig. 60 BR 2 HL Matlab architecture

The fifth BPANN is trained with Resilient backpropagation Fig. 61. In this case, data division is equivalent to LM: 70%, 15%, 15%. The activation functions used were sigmoid, and pure line Fig. 62.

```
%% Resilient Backpropagation 1 HL
NUMBERS='';
for i=1:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='logsig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainrp(net,xt,yt); %tr=info
```

Fig. 61 RPROP 1 HL Matlab code

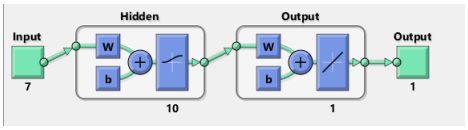


Fig. 62 RPROP 1 HL Matlab architecture

Training stops when any of these conditions occur:

- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min_grad.
- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

The sixth BPANN is equivalent to the previous one with two hidden layers Fig. 63; the second hidden layer is composed of three times the number of neurons than the first hidden layer. The activation functions are sigmoid, sigmoid, and pure line, respectively Fig. 64.

```
%% Resilient Backpropagation 2HL
NUMBERS='';
 for i=1:N
hiddenLayerSize=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='logsig';
net.layers{2}.transferFcn='logsig';
net.layers{3}.transferFcn='purelin';

[net,tr]=trainrp(net,xt,yt); %tr=info
```

Fig. 63 RPROP 2 HL Matlab code

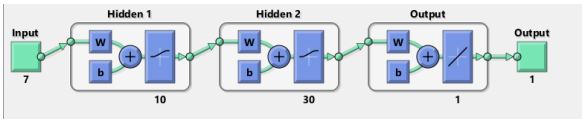


Fig. 64 RPROP 2 HL Matlab architecture

The seventh BPANN uses a Scaled conjugate gradient Fig. 65. Similarly to LM and RPROP, the data is divided into 70%, 15%, and 15%. The activation functions are hyperbolic, and pureline Fig. 66.

```
  %% Scaled Conjugate Gradient 1HL
NUMBERS='';
  for i=1:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainscg(net,xt,yt); %tr=info
```

Fig. 65 SCG 1 HL Matlab code

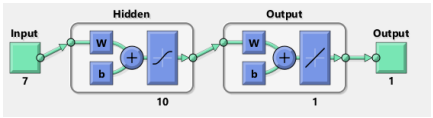


Fig. 66 SCG 1 HL Matlab architecture

Training stops when any of these conditions occur:

- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.

- The performance gradient falls below min_grad.
- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

The BPANN is equivalent to the previous with two hidden layers Fig. 67; the second hidden layer is composed of three times the number of neurons than the first hidden layer. The activation functions are hyperbolic, hyperbolic, and pure line, respectively. Fig. 68.

```
  %% Scaled Conjugate Gradient 2L
NUMBERS='';
  for i=1:N
hiddenLayerSize=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
net.layers{3}.transferFcn='purelin';
[net,tr]=trainscg(net,xt,yt); %tr=info
```

Fig. 67SCG 2 HL Matlab code

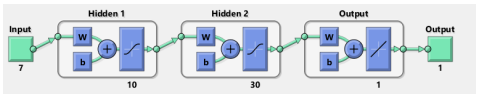


Fig. 68 SCG 2 HL Matlab architecture

The '*Taylor diagram'* code is intended for creating circular graphs, according to the cosine law. Typical inputs are the standard deviation vector, the correlation coefficient vector and the rmse vector; the first element of each vector is referred to as the *observed* model, hence correlation and RMSE are both set to *1* and *0*, respectively. Before starting the code, the relation underlined in red is verified Fig. 69.

```
% TAYLORDIAG Plot a Taylor Diagram
%
% [hp ht axl] = taylordiag(STDs,RMSs,CORs,['option',value])
%
% Plot a Taylor diagram from statistics of different series.
%
% INPUTS:
%   STDs: Standard deviations
%   RMSs: Centered Root Mean Square Difference
%   CORs: Correlation
%
%   Each of these inputs are one dimensional with same length. First
%   indice corresponds to the reference serie for the diagram. For exemple
%   STDs(1) is the standard deviation of the reference serie and STDs(2:N)
%   are the standard deviations of the other series.
%
%   Note that by definition the following relation must be true for all series i:
%   RMSs(i) - sqrt(STDs(i).^2 + STDs(1)^2 - 2*STDs(i)*STDs(1).*CORs(i)) = 0
%   This relation is checked and if not verified an error message is sent. Please see
%   Taylor's JGR article for more informations about this.
%   You can use the ALLSTATS function to avoid this to happen, I guess ;-). You can get
%   it somewhere from: http://codes.guillaumemaze.org/matlab
%
% OUTPUTS:
%   hp: returns handles of plotted points
%   ht: returns handles of the text legend of points
%   axl: returns a structure of handles of axis labels
```

Fig. 69 Cosine law condition

If it is correct then outputs are generated. The first output vector returns handle of plotted points; the second output vector returns the handles of the legend points of the alphabetic text; the third output vector returns a structure of handles of axis labels. In this work, both the number of inputs and outputs have been modified to guarantee a higher loops variability. In particular, the non-modified, '*Taylor diagram'* suffers from a finite number of alphabet

characters, even though the author incorporated in the code the capacity to differentiate between capital and lower case characters. To address this issue, a fourth input was added, called '*NUMBERS',* which stores the *'for loop'* index Fig. 70, and so the number of neurons in every cycle as a character. Fig. 71.

```
% Taylor Diagram
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] = taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
```

Fig. 70 Forth input implementation

```
NUMBERS=split([NUMBERS, num2str(i)]);
```

Fig. 71 'NUMBERS' definition

The coupling of '*for loops'* and '*taylordiag'* allows to locate the respective model with greater ease and ensures increased reliability. Three new outputs have been implemented to enhance utility: color1, color2 and color3. They are RGB parameters that are automatically and randomly generated inside '*taylordiag'* for every architecture. The model is *observed,* for the study*,* has been set to 'red' ([1, 0, 0]) for every analysis Fig. 72. The configuration of all parameters allows the final plot to be enriched with a clear legend.

```
% FINALY PLOT THE POINTS:
    hold on
    for ii = 1 : length(STDs)
        if j==1
            colorsel{j}(ii,:)=[rand,rand,rand];
            colorsel{j}(1,:)=[1 0 0];
            pp(ii)=plot(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)));
            set(pp(ii),'marker','.','markersize',20);
            set(pp(ii),'color',colorsel{j}(ii,:));
%       if length(STDs)<=26
            tt(ii)=text(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)),NUMBERS(ii),'color','r');
%       elseif length(STDs)<=26*2
%           tt(ii)=text(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)),lower(ALPHABET(ii)),'color','r');
%       else
%           error('sorry I don''t how to handle more than 52 points labels !');
%         end
```

Fig. 72 RGB initialization

Two more inputs, related to colour memorization, were added at a later date: *j* and *color* Fig. 73. In the first cycle they are set to *1* and *0*, this allows the *taylordiag* code to recognise that it is running the first out of the three loops (training, validation and testing) and that it has to initialize the random generation of colours. For successive cycles, changing '*j*' and setting *color* with the combination generated previously, grant all three graphs of the same algorithm to have the same shade Fig. 74. Moreover, the new output will no longer yield six values but contain only three. Fig. 75.

```
% Taylor Diagram
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] = taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] = taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color);
end
```

Fig. 73 Fifth and sixth input implementation

```
% FINALY PLOT THE POINTS:
    hold on
    for ii = 1 : length(STDs)
        if j==1
            colorsel{j}(ii,:)=[rand,rand,rand];
            colorsel{j}(1,:)=[1 0 0];
            pp(ii)=plot(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)));
            set(pp(ii),'marker','.','markersize',20);
            set(pp(ii),'color',colorsel{j}(ii,:));
%       if length(STDs)<=26
            tt(ii)=text(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)),NUMBERS(ii),'color','r');
%       elseif length(STDs)<=26*2
%           tt(ii)=text(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)),lower(ALPHABET(ii)),'color','r');
%       else
%           error('sorry I don''t how to handle more than 52 points labels !');
%        end
        else
            colorsel(ii,:)=color(ii,:);
            colorsel(1,:)=[1 0 0];
            pp(ii)=plot(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)));
            set(pp(ii),'marker','.','markersize',20);
            set(pp(ii),'color',colorsel(ii,:));
            tt(ii)=text(rho(ii)*cos(theta(ii)),rho(ii)*sin(theta(ii)),NUMBERS(ii),'color','r');
        end
    end
```

Fig. 74 Modified Taylor diagram color selection

```
% Taylor Diagram
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] = taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] = taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color);
end
```

Fig. 75 Output update for validation and testing phases

The last part of the code is reserved to plot all the previously calculated error parameters.

The entire code is used in this thesis is found in the Appendix chapter.

4 Results

The internal architecture comparison of each BPANN learning methods is highlighted on twenty-two figures from Fig 76 to Fig. 97.

The performance of BPANN, with one hidden layer, trained by Levenberg-Marquardt, are described in Fig. 76 to 78 and Table 2. All three Taylor diagrams highlight how increasing the overall number of neurons in the hidden layer is positive for the forecast model. The best result was found with 7-19-1-1 architecture which has $\mathrm{RMSEt} \cong 73.538;\ \mathrm{MAEt} \cong 54.530;\ \mathrm{MBEt} \cong -0.005;\ \mathrm{MAPEt} \cong 5.651\%;\ \mathrm{Rt} \cong 0.864;\ \mathrm{R^2t} \cong 0.746;\ \mathrm{RMSEval} \cong 74.820;\ \mathrm{MAEval} \cong 55.558;\ \mathrm{MBEval} \cong 2.158;\ \mathrm{MAPEval} \cong 5.154\%;\ \mathrm{Rval} \cong 0.860;\ \mathrm{R^2val} \cong 0.739;\ \mathrm{RMSEte} \cong 75.345;\ \mathrm{MAEte} \cong 55.761;\ \mathrm{MBEte} \cong -0.282;\ \mathrm{MAPEte} \cong 6.009\%;\ \mathrm{Rte} \cong 0.856;\ \mathrm{R^2te} \cong 0.733$, where '*t*' stands for training, '*val*' for validation and '*te*' for testing. Models 16, 17, 18, and 20 are very close to 7-19-1-1 and valid as well; considering issues regarding complexity, architecture 7-16-1-1 may be preferred.

Moreover, the addition of one more hidden layer can further enhance the performance in respect to the previous case and the results are reported in Fig. 79 to 81 and Table 3. This new configuration shows a "constant" improvement for all models. From 1 to 10 this is evident and the step is big, for models 11 to 20 scaling up is more difficult and the trend indicates that it is converging. Configurations 7-18-54-1-1, 7-19-57-1-1 and 7-20-60-1-1 are very promising in regards to forecasting capability with $\mathrm{RMSEt} \cong 20;\ \mathrm{MAEt} \cong 14;\ \mathrm{MAPEt} \cong 0.5\%;\ \mathrm{Rt} \cong 0.99;\ \mathrm{R^2t} \cong 0.98;\ \mathrm{RMSEval} \cong 25;\ \mathrm{MAEval} \cong 16;\ \mathrm{MAPEval} \cong 0.5\%;\ \mathrm{Rval} \cong 0.98;\ \mathrm{R^2val} \cong 0.97;\ \mathrm{RMSEte} \cong 25;\ \mathrm{MAEte} \cong 16;\ \mathrm{MAPEte} \cong 0.6\%;\ \mathrm{Rte} \cong 0.98;\ \mathrm{R^2te} \cong 0.97$. As well as before, for complexity reason architecture 7-18-54-1-1 is considered the best forecasting model.

Training and testing performances of BPANN with one hidden layer, with Bayesian regularization, are represented in Fig 82, 83 and Table 4. Similarly to the first analysis, the number of neurons increases assists in the overall performances. Looking at the diagram it can be seen that 7-16-1-1, 7-19-1-1 and 7-20-1-1 are close to each other; this last architecture reports the best performances, with $\mathrm{RMSEt} \cong 72.565;\ \mathrm{MAEt} \cong 53.185;\ \mathrm{MBEt} \cong 0;\ \mathrm{MAPEt} \cong 5.506\%;\ \mathrm{Rt} \cong 0.868;\ \mathrm{R^2t} \cong 0.753;\ \mathrm{RMSEte} \cong 73.564;\ \mathrm{MAEte} \cong 53.683;\ \mathrm{MBEte} \cong 1.535;\ \mathrm{MAPEte} \cong 5.214\%;\ \mathrm{Rte} \cong 0.864;\ \mathrm{R^2te} \cong 0.746$.

BPANN training and testing, with two hidden layers, trained with Bayesian regularization, are found in Fig. 84, 85 and Table 5. Increasing the number of layers, with this learning method, has enhanced the forecasting ability, in fact, the cluster of architectures in the Taylor diagram is stretched more and it tends toward the *Observed model*. The best performing architecture among BR 2 HL is 7-19-57-1-1 which is characterized by $\text{RMSEt} \cong 14.487$; $\text{MAEt} \cong 9.882$; $\text{MBEt} \cong 0$; $\text{MAPEt} \cong 0.220\%$; $\text{Rt} \cong 0.995$; $\text{R}^2\text{t} \cong 0.990$; $\text{RMSEte} \cong 18.034$; $\text{MAEte} \cong 11.845$; $\text{MBEte} \cong -0.232$; $\text{MAPEte} \cong 0.337\%$; $\text{Rte} \cong 0.992$; $\text{R}^2\text{te} \cong 0.985$.

The BPANN performances, with one hidden layer, trained by Resilient Propagation are reported on Fig. 86 to 88 and Table 6. Compared to previous models, this algorithm does not improve "regularly" with the number of neurons, in fact, the Taylor diagrams show a high model density. Configuration 7-16-1-1 is the best performer, with $\text{RMSEt} \cong 94.565$; $\text{MAEt} \cong 73.858$; $\text{MBEt} \cong -0.234$; $\text{MAPEt} \cong 9.635\%$; $\text{Rt} \cong 0.762$; $\text{R}^2\text{t} \cong 0.581$; $\text{RMSEval} \cong 95.389$; $\text{MAEval} \cong 74.571$; $\text{MBEval} \cong 2.604$; $\text{MAPEval} \cong 9.052\%$; $\text{Rval} \cong 0.759$; $\text{R}^2\text{val} \cong 0.576$; $\text{RMSEte} \cong 96.056$; $\text{MAEte} \cong 75.498$; $\text{MAPEte} \cong 9.658\%$; $\text{Rte} \cong 0.752$; $\text{R}^2\text{te} \cong 0.565$.

The BPANN behaviours of two hidden layers, trained by Resilient Propagation, are highlighted from Fig. 89 to 91 and Table 7. All the architectures have similar and poor characteristics in respect to previous multi hidden layered architectures. In fact, RMSEt is high, spanning from 98 of 7-1-3-1-1 to 74.5 of 7-18-54-1-1, MAE from 74 to 54, MAPE from 10% to 6%, R from 0.74 to 0.86, R^2 from 0.55 to 0.74; the same pattern occurs in both validation and testing where RMSEval is in between 98 of 7-1-3-1-1 and 75 of 7-18-54-1-1, MAE from 74 to 55, MAPE from 9.4% to 5.5%, R from 0.74 to 0.86, R^2 from 0.55 to 0.73, and RMSEte in between 98 of 7-1-3-1-1 and 76 of 7-18-54-1-1, MAE from 75 to 55, MAPE from 10% to 6.4%, R from 0.74 to 0.85, R^2 from 0.54 to 0.73.

Training, validation and testing performances of BPANN with one hidden layer, with Scaled conjugate gradient, are reported from Fig. 92 to 94 and Table 8. SCG performances are far from the *Observed model* and the best architecture is represented by 7-8-1-1 with $\text{RMSEt} \cong 106.913$; $\text{MAEt} \cong 84.852$; $\text{MBEt} \cong -0.548$; $\text{MAPEt} \cong 12.145\%$; $\text{Rt} \cong 0.681$; $\text{R}^2\text{t} \cong 0.464$; $\text{RMSEval} \cong 106.880$; $\text{MAEval} \cong 84.916$; $\text{MBEval} \cong 0.434$; $\text{MAPEval} \cong 12.055\%$; $\text{Rval} \cong 0.684$; $\text{R}^2\text{val} \cong 0.468$; $\text{RMSEte} \cong 106.837$; $\text{MAEte} \cong$

$85.103; \text{MBEte} \cong -2.188;\ \text{MAPEte} \cong 12.708\%;\ \text{Rte} \cong 0.68;\ \text{R}^2\text{te} \cong 0.462$. All other configurations are concentrated around the circle of $\text{RMSE} = 115$.

The BPANN behaviours, with two hidden layers, trained by Scaled conjugate gradient are reported in Fig. 95 to 97 and Table 10. Taylor diagrams are characterized by many "misbehaviors" in fact 8, 11, 16, 18 architectures have the worst performances among all. Moreover, 2 and 5 architectures are scaled up from their relative single hidden layer configurations. The best model, in terms of performances, is represented by 7-19-57-1-1 whose $\text{RMSEt} \cong 69.330;\ \text{MAEt} \cong 51.489; \text{MBEt} \cong -0.413;\ \text{MAPEt} \cong 4.955\%;\ \text{Rt} \cong 0.880;\ \text{R}^2\text{t} \cong 0.775;\ \text{RMSEval} \cong 69.463;\ \text{MAEval} \cong 52.023; \text{MBEval} \cong 0.664;\ \text{MAPEval} \cong 4.608\%;\ \text{Rval} \cong 0.881;\ \text{R}^2\text{val} \cong 0.775;\ \text{RMSEte} \cong 71.122;\ \text{MAEte} \cong 52.914; \text{MBEte} \cong 0.291;\ \text{MAPEte} \cong 5.056\%;\ \text{Rte} \cong 0.873;\ \text{R}^2\text{te} \cong 0.762.$ However, the best architecture that might be chosen is 7-5-15-1-1 because it has similar results, but less complex, as showed by $\text{RMSEt} = 73.836;\ \text{MAEt} \cong 53.371; \text{MBEt} \cong -0.299;\ \text{MAPEt} \cong 5.812\%;\ \text{Rt} \cong 0.863;\ \text{R}^2\text{t} \cong 0.744;\ \text{RMSEval} \cong 75.708;\ \text{MAEval} \cong 55.017; \text{MBEval} \cong 1.598;\ \text{MAPEval} \cong 5.458\%;\ \text{Rval} \cong 0.856;\ \text{R}^2\text{val} \cong 0.733;\ \text{RMSEte} \cong 76.509;\ \text{MAEte} \cong 55.169;\ \text{MBEte} \cong -0.386;\ \text{MAPEte} \cong 6.047\%;\ \text{Rte} \cong 0.851;\ \text{R}^2\text{te} \cong 0.724.$

Figure 98 to 115 are intended to classify and compare all the previously analyzed architectures; the set of images are divided into three subsections, one dedicated to training, one for validation, and the last section is for the testing phase. The sequence of parameters remains consistent for each of this subsection. The training subsection spans from Fig. 98 to Fig. 103. The first image presents the trend of RMSEt, on the y-axis, versus the number of neurons in the first hidden layer, on the x-axis. Typically, all architectures with a single hidden layer have poor results. SCG 2 HL and RP 2 HL are competitive only with few neurons (two). In later stage, BR 2 HL revealed the best performance till seventeen/eighteen neurons where LM 2 HL has an $\text{RMSEt} \cong 20.837$ and $\text{RMSEt} \cong 19.775$ closer to the Bayesian results: $\text{RMSEt} \cong 20.180$ and $\text{RMSEt} = 18.730$. The graph concludes with the overcoming of BR 2 HL, whose best result is $\mathbf{RMSEt \cong 14.487}$ for 7-19-57-1-1 configuration.

Fig. 99 presents MAEt, which almost has the same shape of RMSEt. BR 2 HL 7-19-57-1-1 architecture has the best performance with $\mathbf{MAEt \cong 9.882}$.

Fig. 100 shows MAPEt, which follows the same pattern of previous figures. BR 2 HL 7-19-57-1-1 provides a $\mathbf{MAPEt} \cong \mathbf{0.220}\%$.

Fig. 101 reports MBEt values. Given the nature of this parameter, it is evident that RPROP 1 HL, RPROP 2 HL, SCG 1 HL, and SCG 2 HL biases have been overestimated, since values are lower than 0 [78]. For all the other architectures BPANN neither underestimates nor overestimates the prediction [79].

Fig. 102 and Fig. 103 can be analyzed together since they correspond to Rt and R^2t plots. Similarly, concerning Fig. 98, SCG 2 HL and RPROP 2 HL both have acceptable results either for one, two or three neurons. For subsequent architectures, BR 2 HL and LM 2 HL are the most valid models. The best result is BR 2 HL 7-19-57-1-1 with $\boldsymbol{Rt} \cong \mathbf{0.995}$ and $\boldsymbol{R^2t} \cong \mathbf{0.990}$, respectively.

The validation subsection spans from Fig. 104 to Fig. 109. Due to the nature of BR, it has not been considered in this study. RMSEval, MAEval, and MAPEval, Fig.104 Fig. 105 Fig. 106, follow the same behaviour of the training subsection. LM 2 HL models' performances dominate when the number of neurons is greater than six. The best values of RMSEval and MAEval were achieved with 7-18-54-1-1 $\mathbf{RMSEval} \cong \mathbf{24.044}$, $\mathbf{MAEval} \cong \mathbf{15.438}$, while the best MAPEval comes from 7-20-60-1-1 $\mathbf{MAPEval} \cong \mathbf{0.470}\%$.

MBEval, Fig. 107, shows that the majority of the models found in the validation phase underestimated the-prediction since MBE values were greater than 0.

In Fig. 108 and Fig. 109 Rval and R^2val are reciprocal. The best values were $\mathbf{Rval} \cong \mathbf{0.986}$ and $\mathbf{R^2val} \cong \mathbf{0.973}$, obtained with the architecture 7-9-27-1-1

The last subsection goes from Fig. 110 to Fig. 115. The first two of these illustrations, which represent RMSEte, MAEte, Fig. 110 Fig. 111, have the same shape of the corresponding values both in training and validation. The best value is provided by BR 2 HL with 7-19-57-1-1 and the configuration $\mathbf{RMSEte} \cong \mathbf{18.034}$, $\mathbf{MAEte} \cong \mathbf{11.845}$.

Fig. 112 plots MAPEte and highlights that the architecture LM 2 HL 7-17-51-1-1 has MAPEte $\cong 0.570\%$ which is slightly greater than the corresponding BR 2 HL model, whose MAPEte $\cong 0.583\%$. Despite this countertrend, the best architecture in terms of MAPEte is BR 2 HL 7-16-48-1-1 with $\mathbf{MAPEte} \cong \mathbf{0.294}\%$.

Fig. 113 illustrates MBEt's performance. In this case, most architectures overestimated the prediction. Nevertheless, there are many models which have a MBEte close to 0.

The last two figures, Fig. 114 and Fig. 115, show that the best Rte and R^2te belong to BR 2 HL with the architecture 7-19-57-1-1 $\mathbf{Rte \cong 0.992}$ and $\mathbf{R^2te \cong 0.985}$.

These results highlight the better overall performance of BR 2 HL over the other learning algorithms, and among its architecture, the best found is the 7-19-57-1-1 one. However, these analyses are not weighted in terms of time and epochs required to achieve that goal. Since single-layered architectures perform poorly, time and epochs analyses have not been considered. Fig. 116 and Table 10 denote the time, in seconds, of all architecture. In the case of BR 2 HL, it is evident that the almost exponential dependence between the number of neurons and the time required for processing; this analysis ends with 1500 seconds for the architecture 7-20-60-1-1. In contrast, the LM 2 HL "time function" can be divided into two steps: the first-step is from 7-1-3-1-1 to 7-16-48-1-1. It required a maximum of 84.544 seconds, and the second step is from 7-17-51-1-1-until the end and reaches a maximum at the eighteenth model with 277.037 seconds. In terms of time, SCG is the best performer. Fig. 117 and Table 11 illustrate each epochs of these model architectures, respectively. There are some concerns regarding BR, and similarly RPROP. In only six out of twenty cases the algorithm has not stopped because of epochs constraint, which is set by default to 1000. The fact that these six are in the first ten architectures means that in all the successive architectures the model could be even more accurate; increasing the number of epochs can, and might, increase the time required to process. Since the last models provided a very good result the study was not extended; moreover, as it can be seen from Fig. 84, Fig. 85, and from Table 5 the rate of improvement is inversely proportional to the complexity. While SCG has stopped only once, LM has never stopped due to epochs constraint for any architecture. The outcome highlighted how architectures with two hidden layers can overcome their respective one layered architecture, and among the algorithms, LM and BR are the most promising. Despite the training velocity of LM 7-18-1-1, the best model to consider is BR 7-19-57-1-1 thanks to its accuracy $RMSEt \cong 14.487;\ MAEt \cong 9.882;\ MBEt \cong 0;\ MAPEt \cong 0.220\%;\ Rt \cong 0.995;\ R^2t \cong 0.990;\ RMSEte \cong 18.034;\ MAEte \cong 11.845;\ MBEte \cong -0.232;\ MAPEte \cong 0.337\%;\ Rte \cong 0.992;\ R^2te \cong 0.985$. Fig 118 to Fig 121, show the regression, the performance, and the plots of both the training and the test subset of this architecture, respectively.

4.1 Taylor diagrams

4.1.1 LM 1 HL

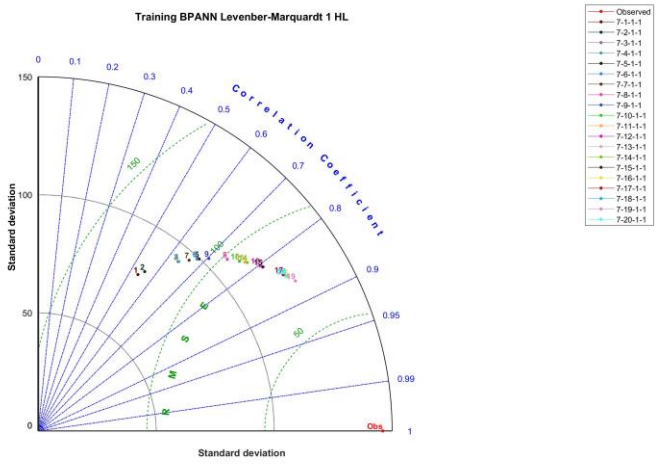


Fig. 76 LM 1 HL training phase

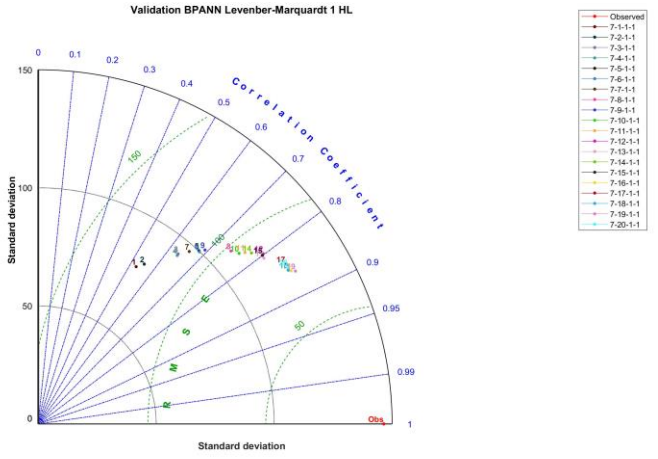


Fig. 77 LM 1 HL validation phase

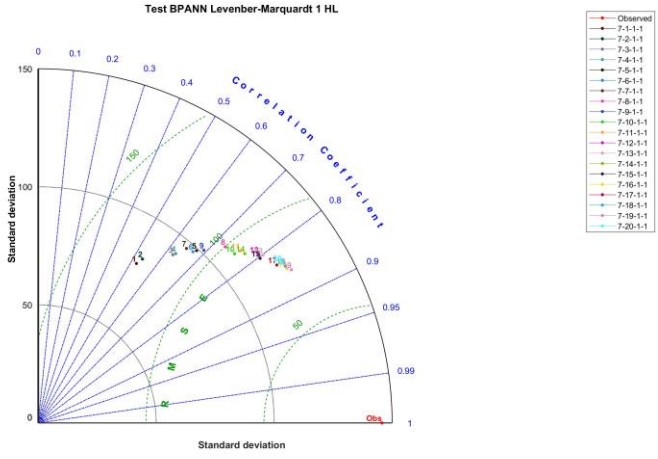


Fig. 78 LM 1 HL testing phase

4.1.2 LM 2 HL

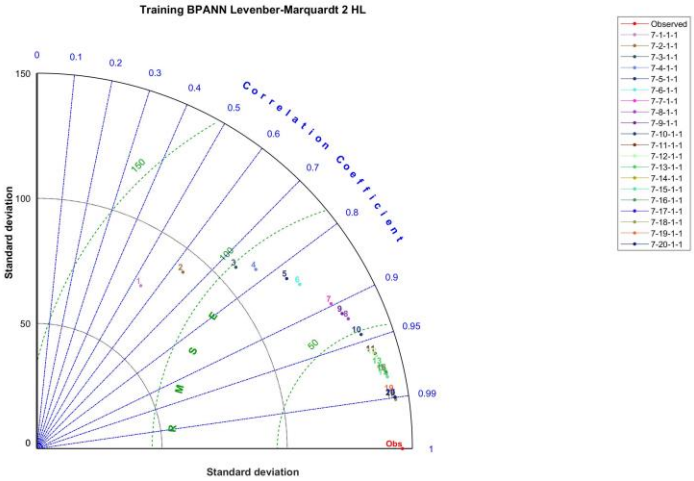


Fig. 79 LM 2 HL training phase

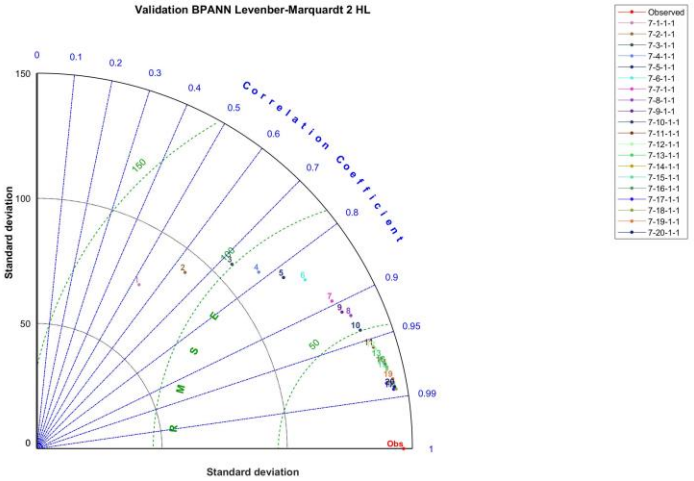


Fig. 80 LM 2 HL validation phase

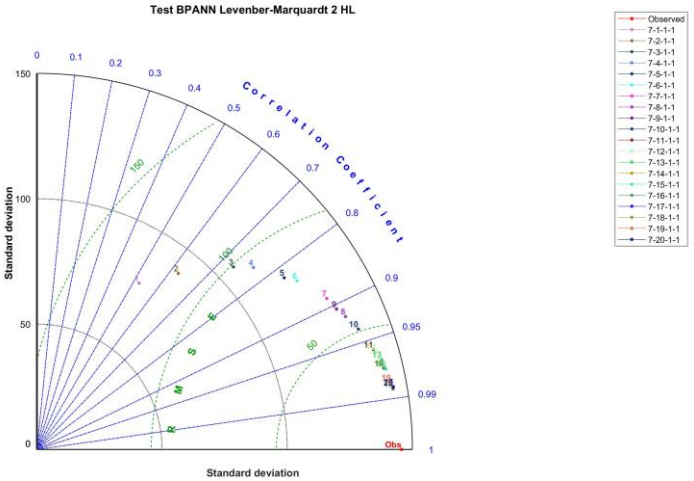


Fig. 81 LM 2 HL testing phase

4.1.3 BR 1 HL

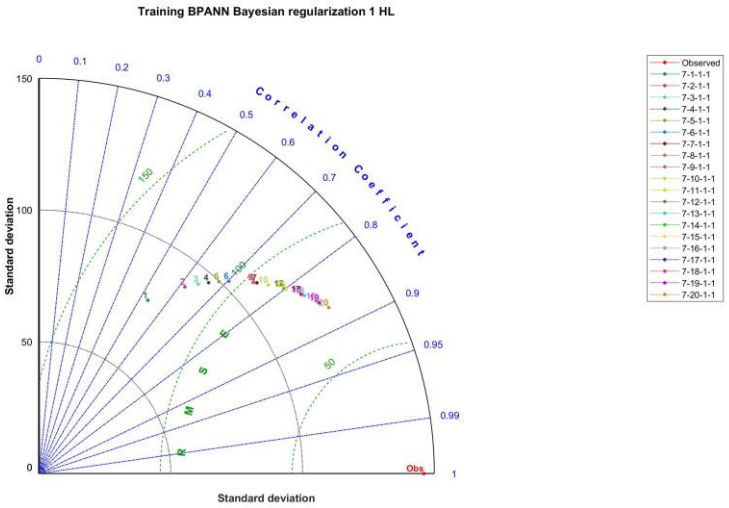


Fig. 82 BR 1 HL training phase

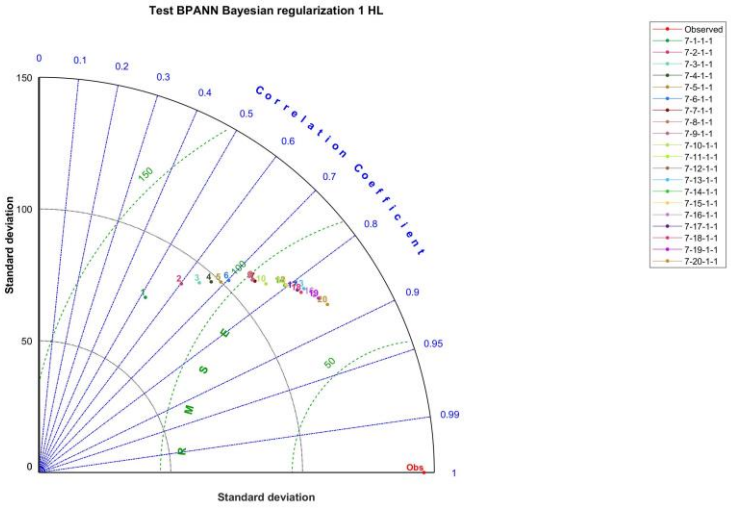


Fig. 83 BR 1 HL testing phase

4.1.4 BR 2 HL

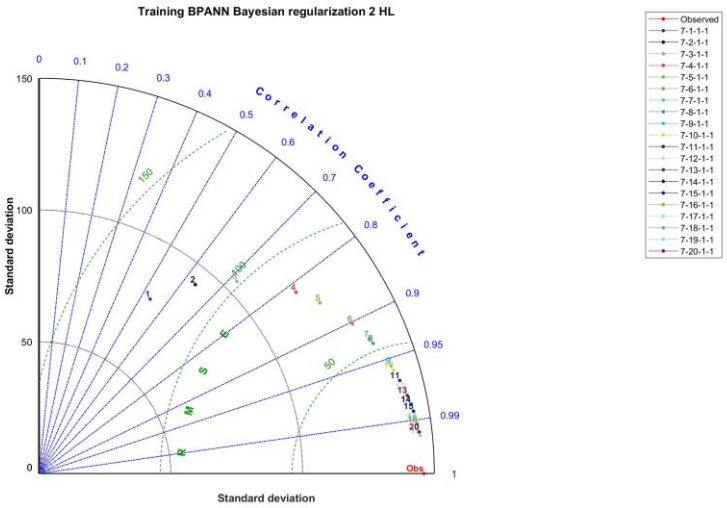


Fig. 84 BR 2 HL training phase

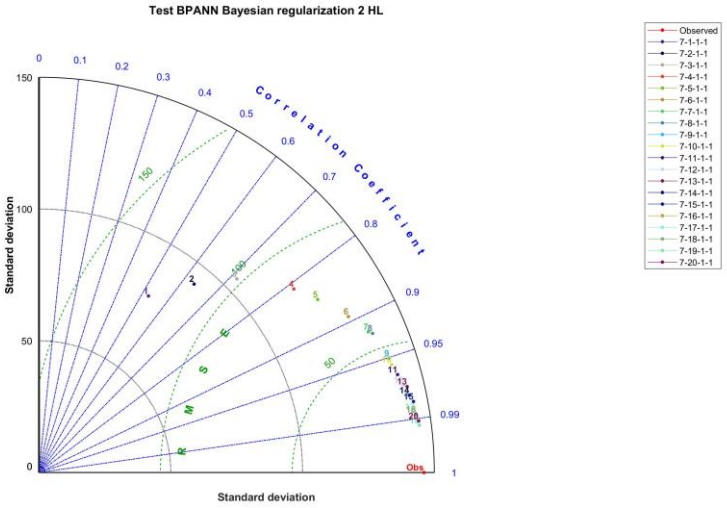


Fig. 85 BR 2 HL testing phase

4.1.5 RPROP 1 HL

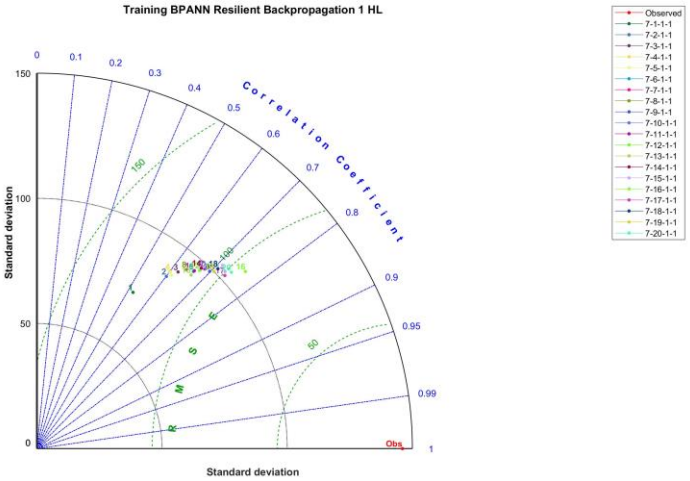


Fig. 86 RPROP 1 HL training phase

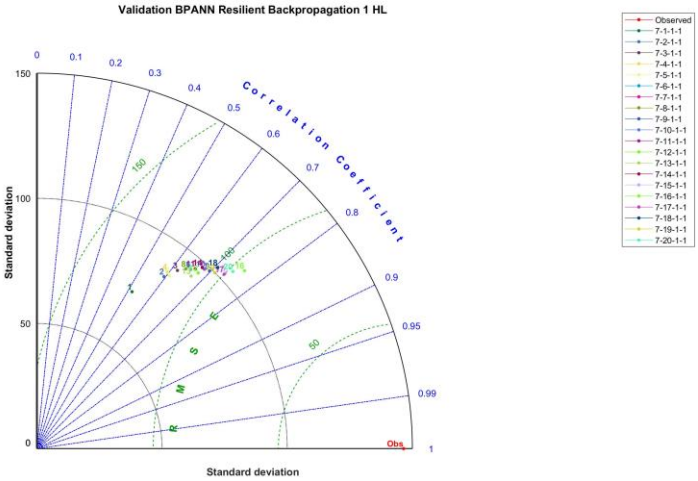


Fig. 87 RPROP 1 HL validation phase

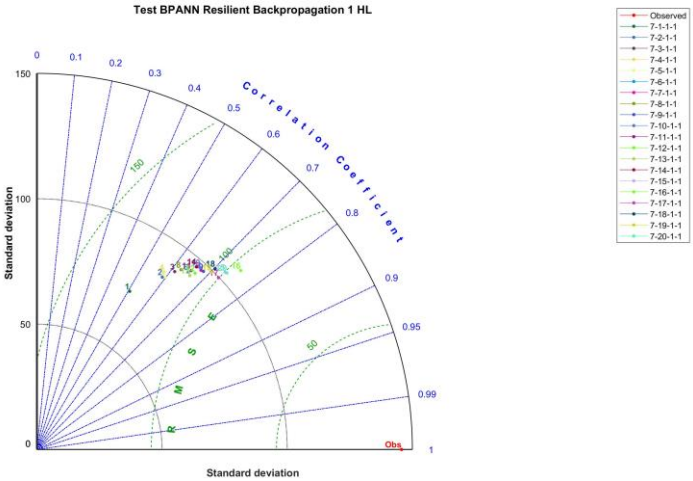


Fig. 88 RPROP 1 HL testing phase

4.1.6 RPROP 2 HL

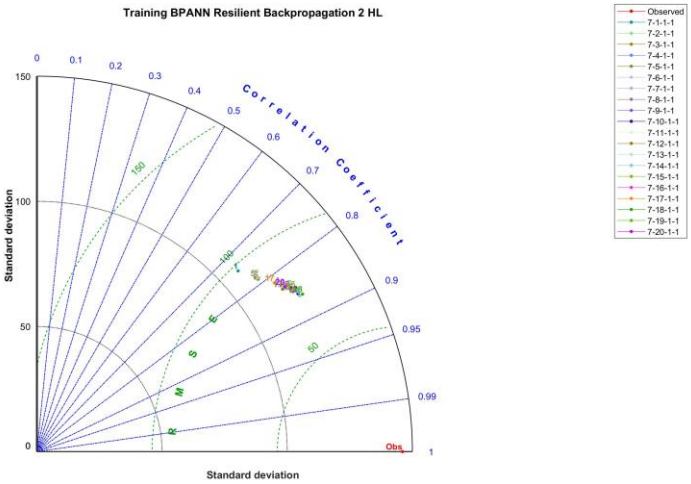


Fig. 89 RPROP 2 HL training phase

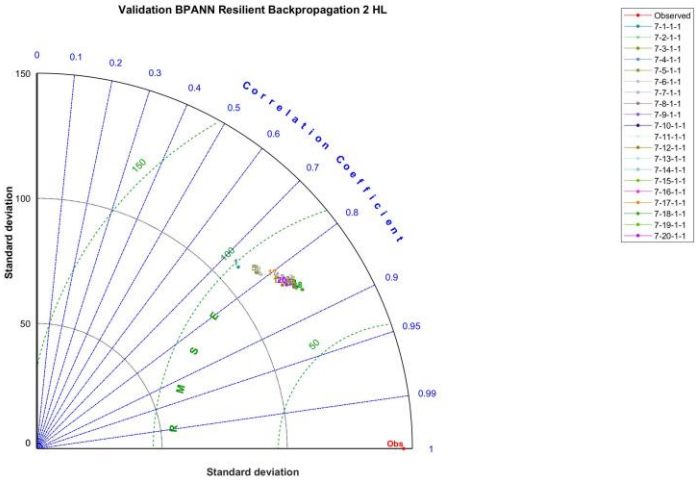


Fig. 90 RPROP 2 HL validation phase

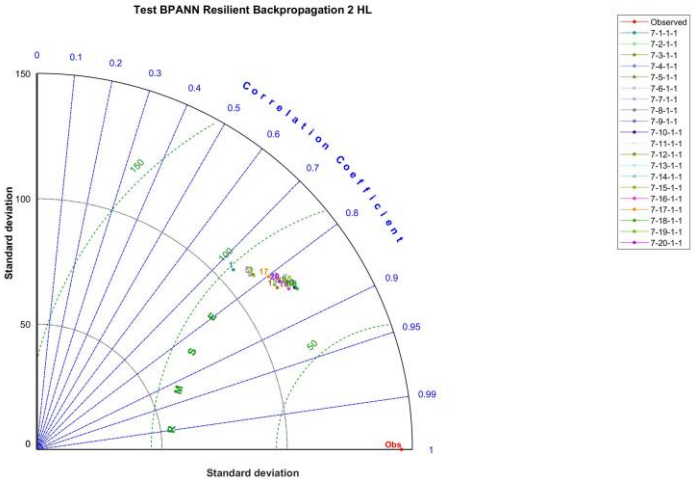


Fig. 91 RPROP 2 HL testing phase

4.1.7 SCG 1 HL

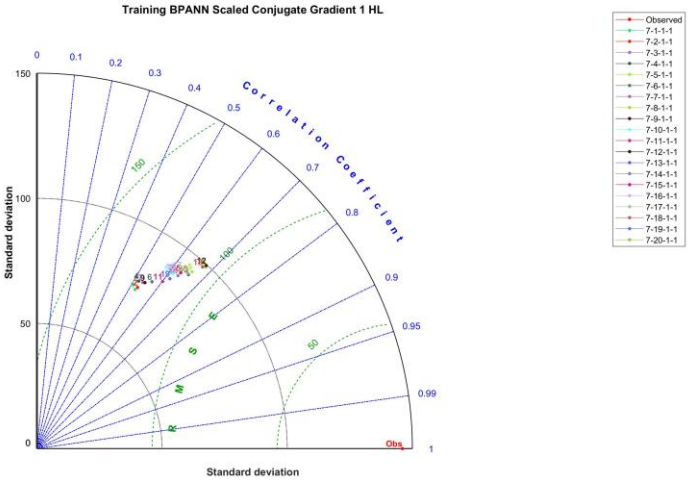


Fig. 92 SCG 1 HL training phase

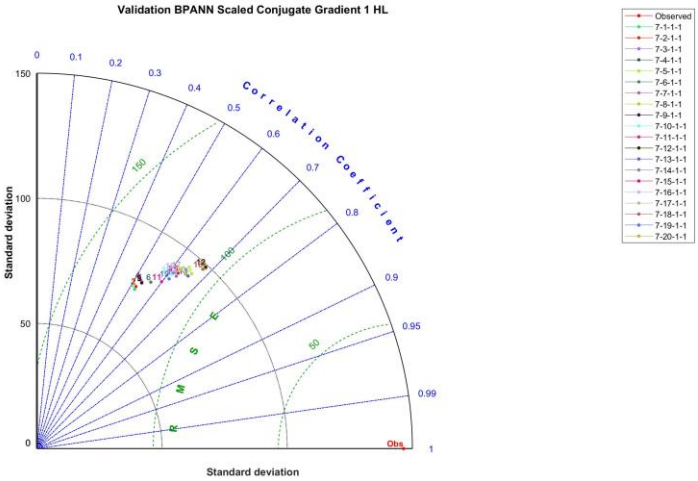


Fig. 93 SCG 1 HL validation phase

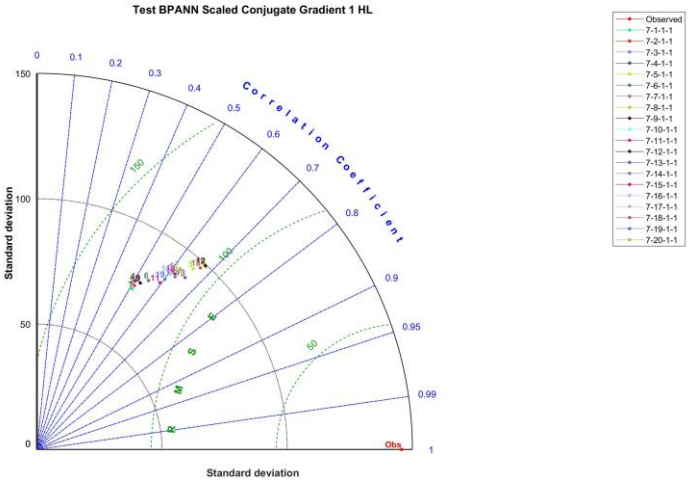


Fig. 94 SCG 1 HL testing phase

4.1.8 SCG 2 HL

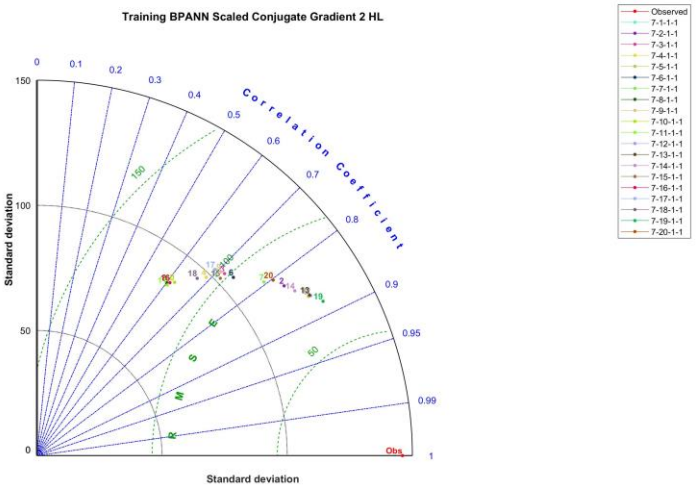


Fig. 95 SCG 2 HL training phase

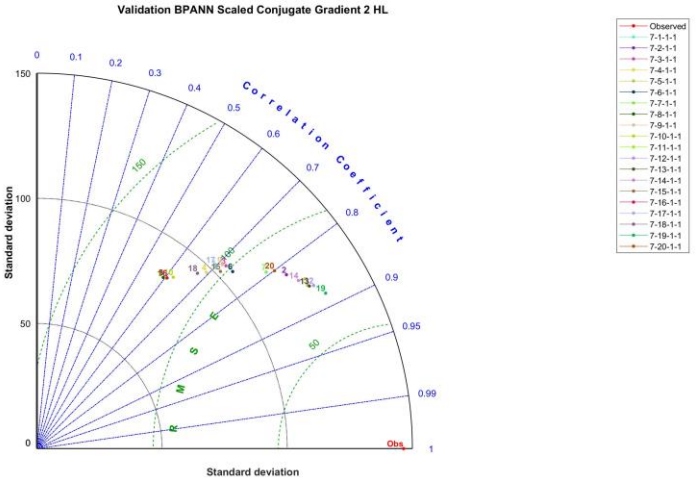


Fig. 96 SCG 2 HL validation phase

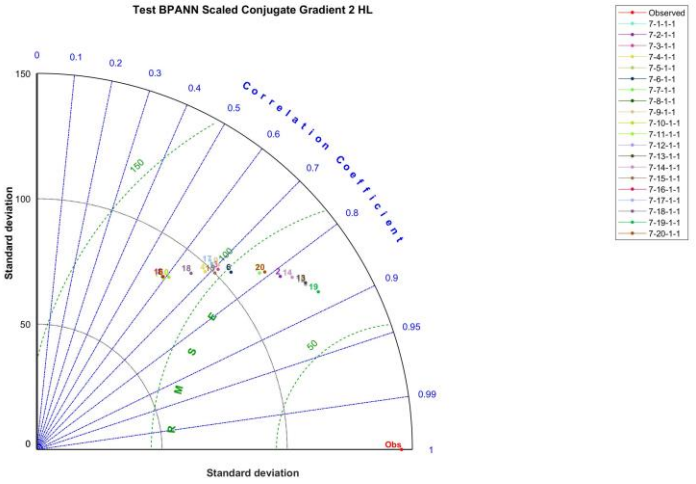


Fig. 97 SCG 2 HL testing phase

4.2 Error diagrams

4.2.1 Training

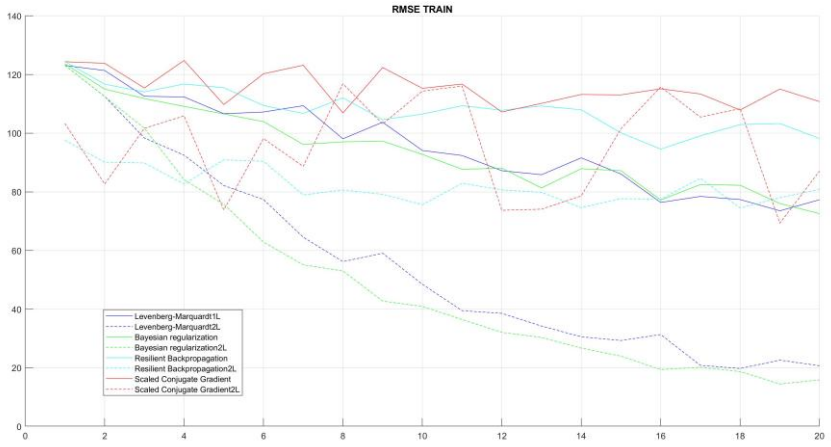


Fig. 98 RMSE training phase

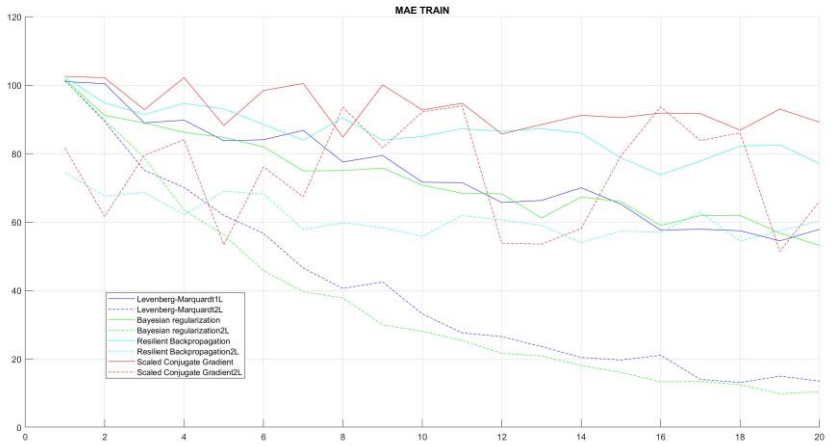


Fig. 99 MAE training phase

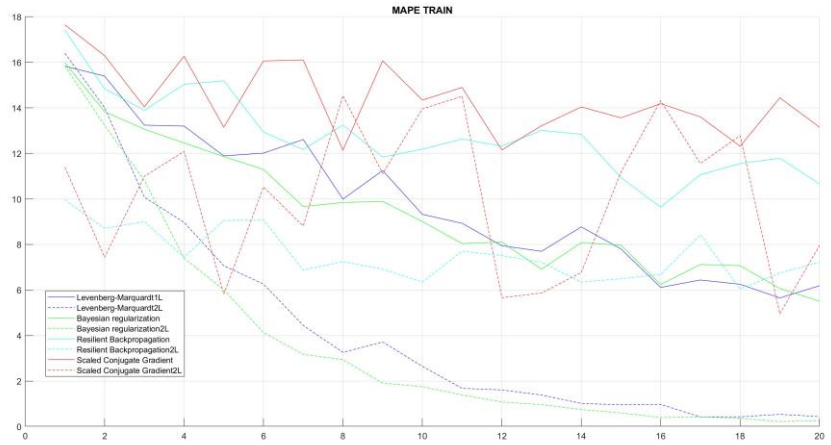


Fig. 100 MAPE training phase

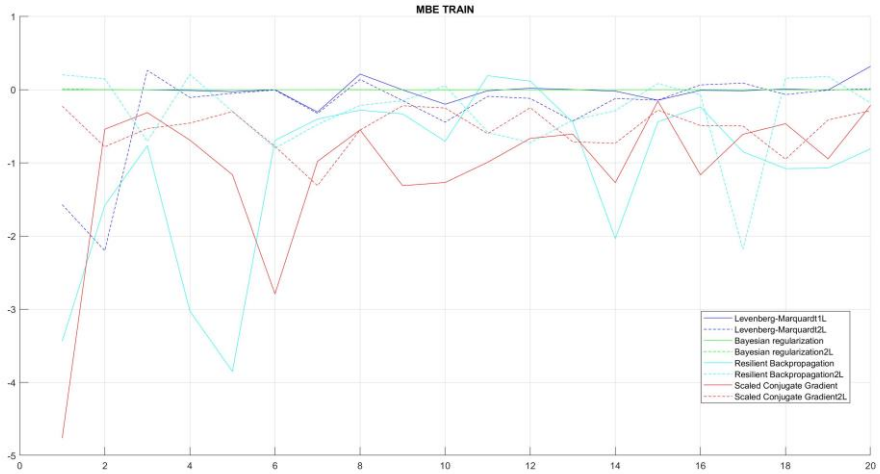


Fig. 101 MBE training phase

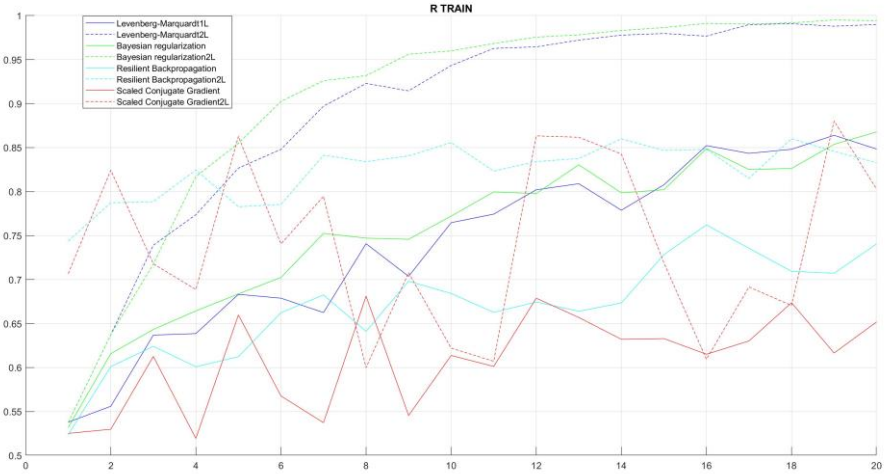


Fig. 102 R training phase

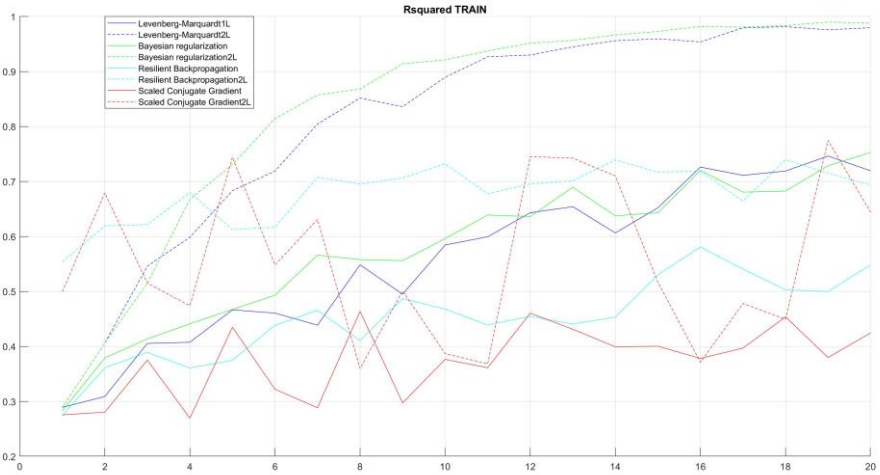


Fig. 103 R^2 training phase

4.2.2 Validation

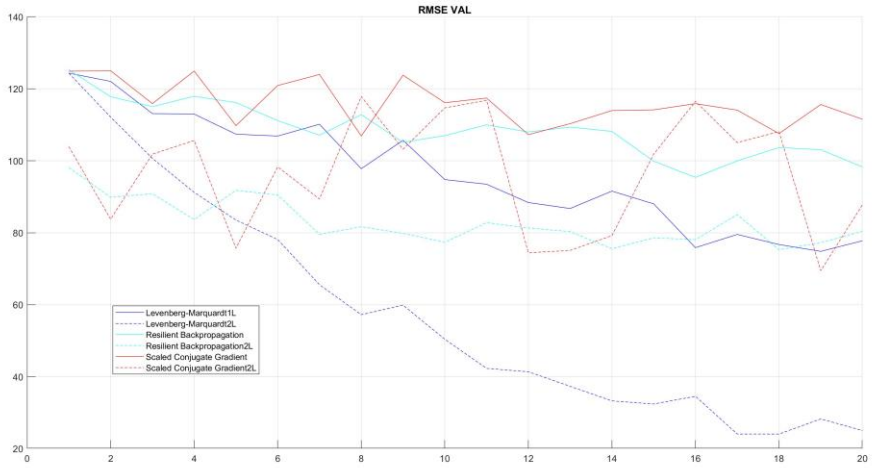


Fig. 104 RMSE validation phase

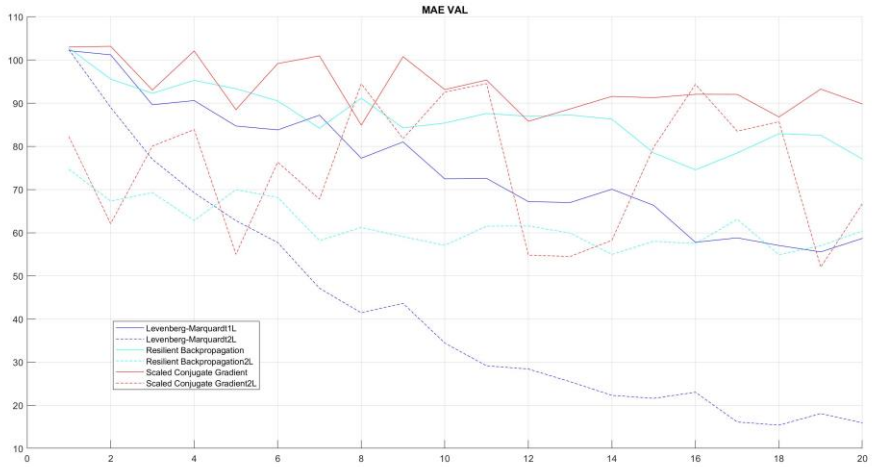


Fig. 105 MAE validation phase

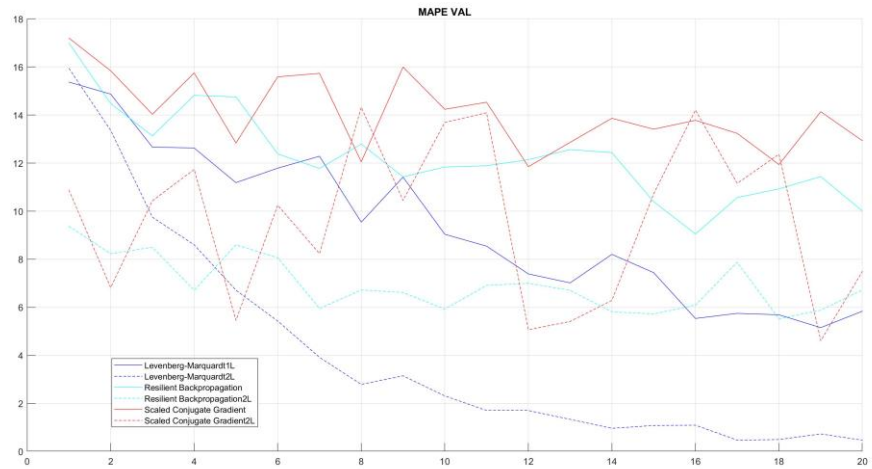


Fig. 106 MAPE validation phase

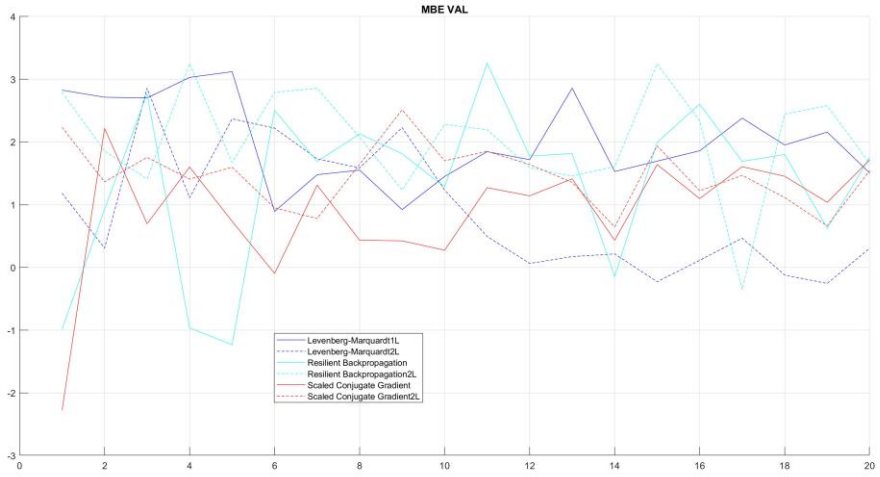


Fig. 107 MBE validation phase

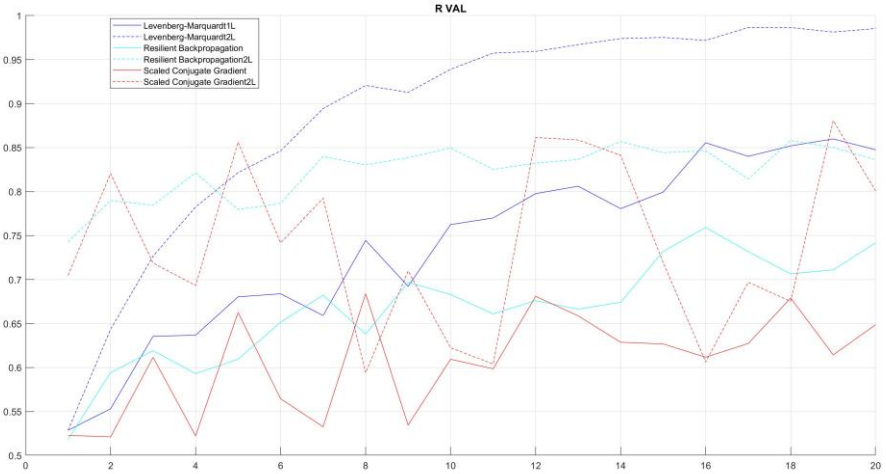


Fig. 108 R validation phase

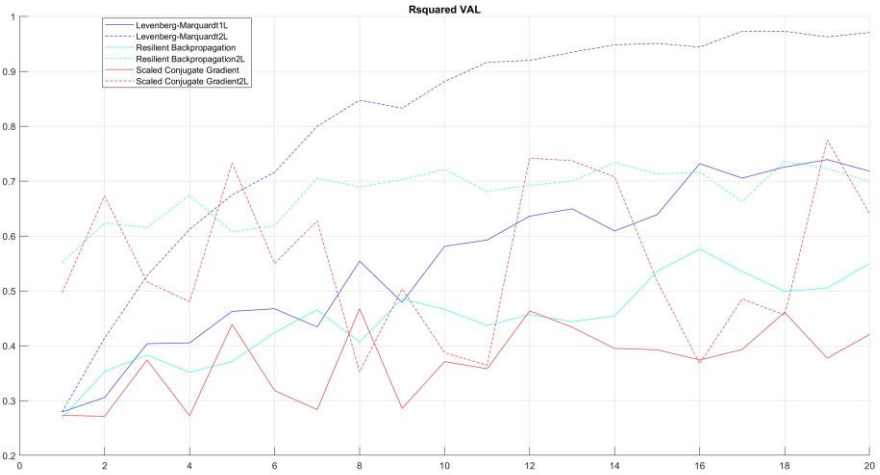


Fig. 109 R^2 validation phase

4.2.3 Testing

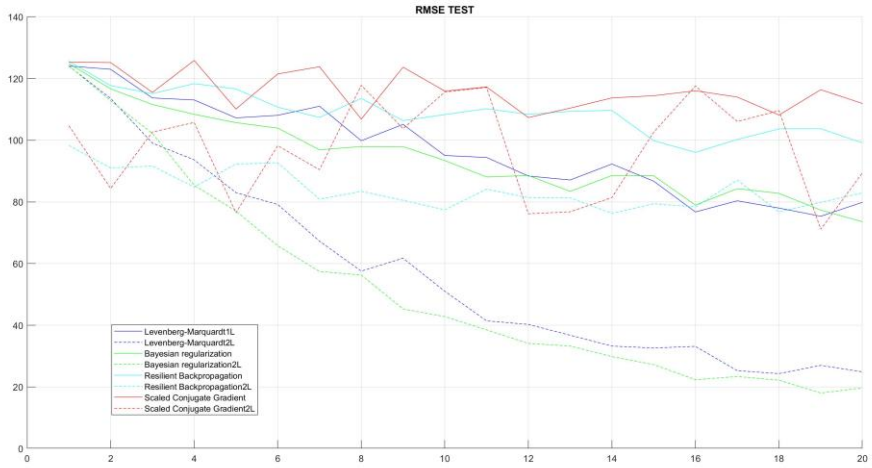


Fig. 110 RMSE testing phase

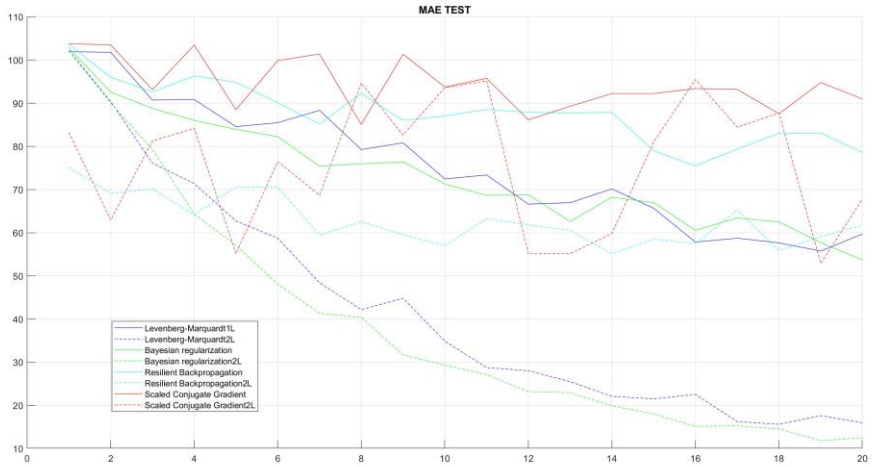


Fig. 111 MAE testing phase

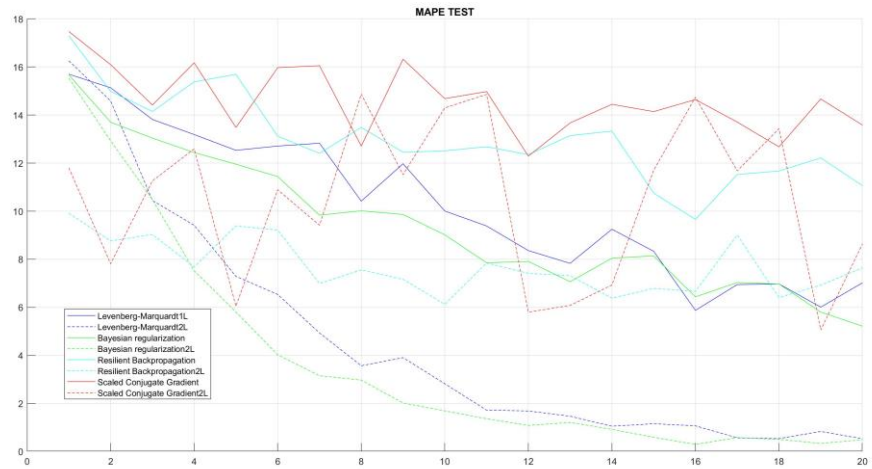


Fig. 112 MAPE testing phase

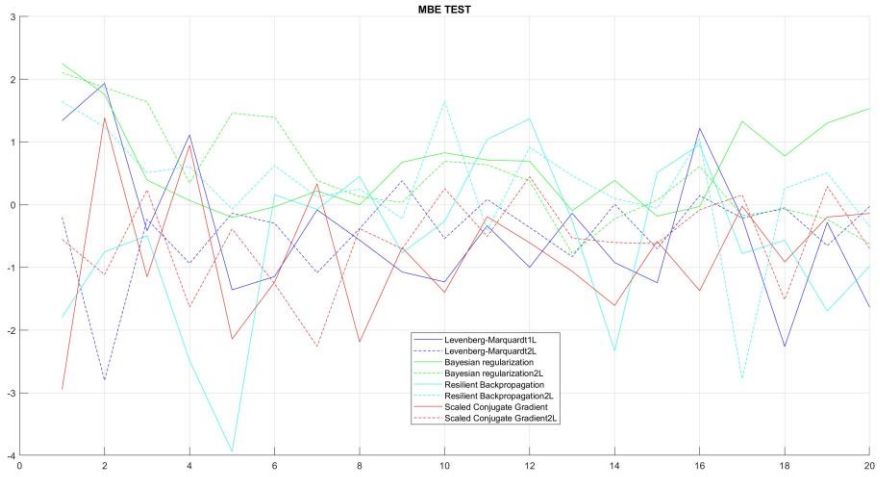


Fig. 113 MBE testing phase

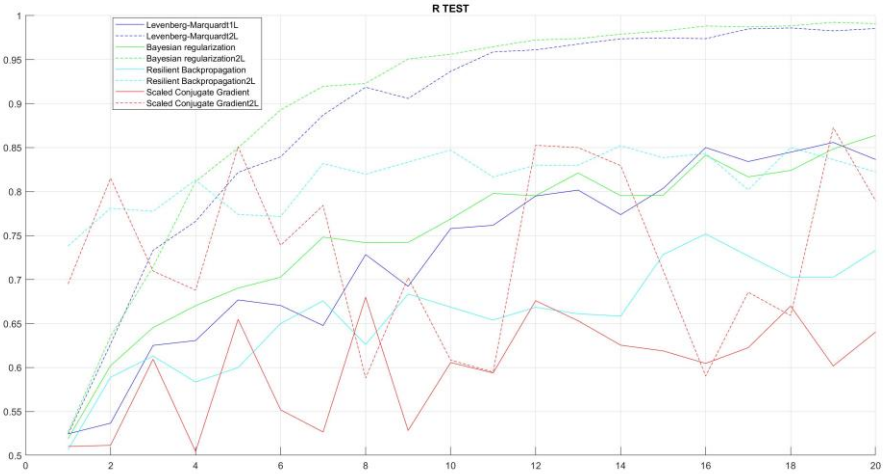


Fig. 114 R testing phase

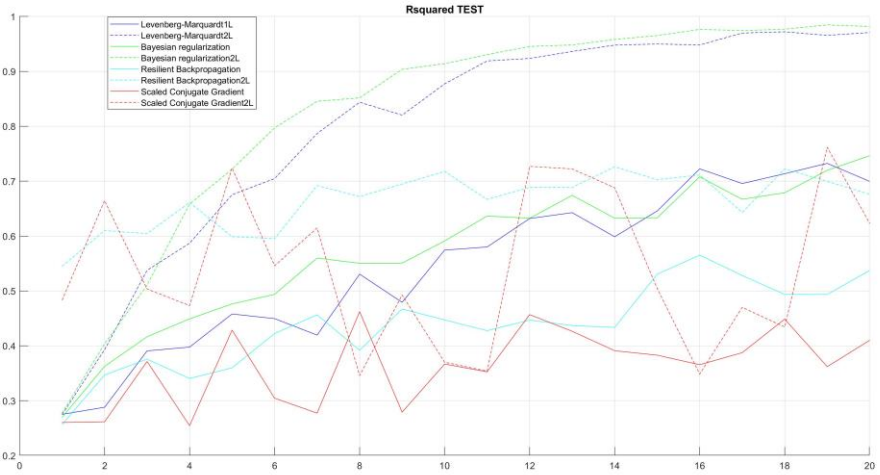


Fig. 115 R^2 testing phase

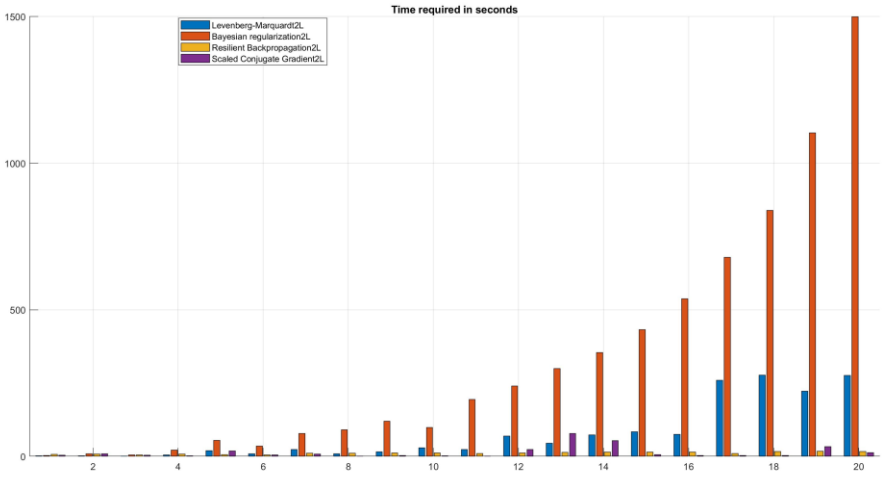


Fig. 116 Time of 2 HL architectures

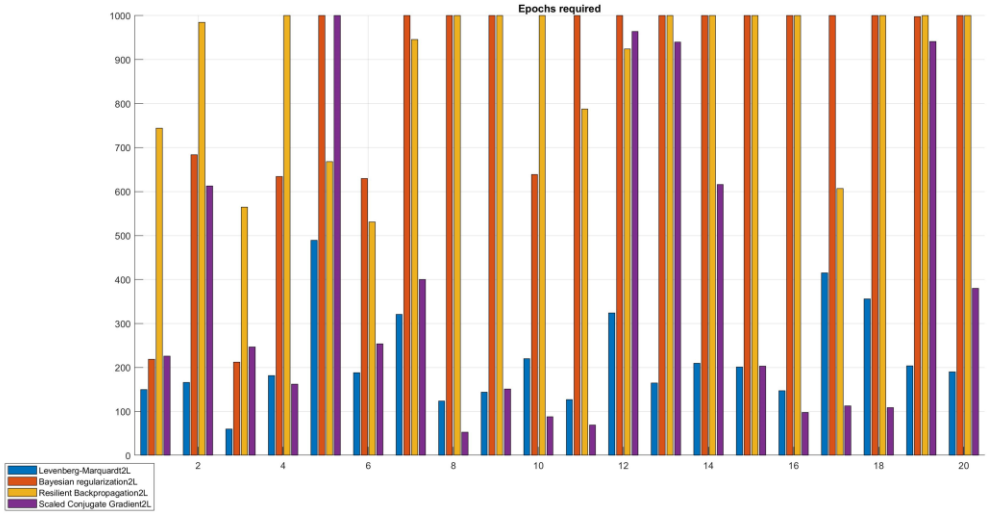


Fig. 117 Epochs of 2 HL architectures

Levenberg-Marquardt 1 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	123.105	101.091	0.000	15.833	0.538	0.289	124.358	102.168	2.826	15.373	0.529	0.280	124.057	101.922	1.340	15.700	0.525	0.275
2	121.382	100.479	0.001	15.395	0.556	0.309	122.056	101.207	2.714	14.874	0.553	0.306	122.993	101.794	1.935	15.136	0.537	0.288
3	112.584	89.027	-0.001	13.243	0.637	0.406	113.114	89.641	2.702	12.666	0.636	0.404	113.699	90.728	-0.416	13.817	0.625	0.391
4	112.376	89.832	-0.010	13.200	0.639	0.408	112.982	90.607	3.030	12.630	0.637	0.405	113.057	90.858	1.113	13.188	0.631	0.398
5	106.637	83.809	-0.027	11.890	0.683	0.467	107.393	84.725	3.121	11.188	0.680	0.463	107.246	84.576	-1.357	12.533	0.677	0.458
6	107.233	84.096	**0.000**	12.014	0.679	0.461	106.878	83.815	**0.892**	11.790	0.684	0.468	108.077	85.491	-1.145	12.711	0.671	0.450
7	109.389	86.849	-0.304	12.610	0.662	0.439	110.160	87.232	1.478	12.287	0.659	0.434	111.012	88.353	-0.081	12.828	0.648	0.420
8	98.101	77.588	0.214	9.993	0.741	0.549	97.796	77.260	1.551	9.545	0.745	0.554	99.842	79.267	-0.565	10.418	0.728	0.531
9	103.756	79.506	-0.002	11.248	0.704	0.495	105.697	81.034	0.922	11.418	0.692	0.479	105.129	80.828	-1.072	11.975	0.692	0.479
10	94.109	71.691	-0.198	9.322	0.765	0.585	94.772	72.468	1.450	9.039	0.763	0.581	95.043	72.454	-1.231	10.007	0.758	0.574
11	92.399	71.551	-0.014	8.931	0.774	0.600	93.484	72.587	1.845	8.545	0.770	0.593	94.399	73.378	-0.335	9.385	0.762	0.580
12	87.220	65.749	0.020	7.945	0.802	0.643	88.388	67.210	1.718	7.390	0.798	0.636	88.390	66.623	-0.998	8.361	0.795	0.632
13	85.851	66.368	0.002	7.700	0.809	0.654	86.729	66.992	2.858	7.017	0.806	0.650	87.099	66.957	-0.135	7.829	0.802	0.643
14	91.601	70.066	-0.020	8.771	0.779	0.607	91.569	70.076	1.529	8.203	0.781	0.609	92.286	70.159	-0.922	9.249	0.774	0.599
15	86.103	65.268	-0.144	7.805	0.808	0.652	88.029	66.344	1.695	7.442	0.799	0.639	86.723	65.671	-1.244	8.333	0.804	0.646
16	76.390	57.694	-0.008	6.114	0.852	0.726	75.858	57.785	1.860	5.535	0.856	0.732	76.742	57.839	1.220	5.874	0.850	0.723
17	78.455	57.945	-0.014	6.440	0.843	0.711	79.524	58.813	2.380	5.749	0.840	0.706	80.355	58.741	-0.208	6.946	0.834	0.696
18	77.381	57.503	0.010	6.251	0.848	0.719	76.721	57.040	1.952	5.689	0.852	0.726	77.993	57.649	-2.261	6.971	0.845	0.714
19	**73.538**	**54.530**	-0.005	**5.651**	**0.864**	**0.746**	**74.820**	**55.558**	2.158	**5.154**	**0.860**	**0.739**	**75.345**	**55.761**	**-0.282**	**6.009**	**0.856**	**0.733**
20	77.324	57.928	0.320	6.188	0.848	0.720	77.770	58.703	1.505	5.841	0.847	0.718	79.877	59.722	-1.635	7.020	0.836	0.700

Table 2 LM 1 HL

Levenberg-Marquardt 2 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	123.166	101.445	-1.570	16.399	0.537	0.289	124.338	102.435	1.184	15.949	0.529	0.279	124.061	102.247	-0.199	16.259	0.524	0.275
2	112.552	89.411	-2.200	14.008	0.637	0.406	112.148	89.034	0.308	13.366	0.643	0.414	113.542	90.416	-2.800	14.589	0.627	0.393
3	98.381	75.165	0.265	10.076	0.739	0.546	100.673	76.998	2.858	9.749	0.727	0.528	99.055	76.106	-0.232	10.447	0.733	0.538
4	92.514	70.151	-0.107	8.970	0.774	0.599	91.220	69.267	1.110	8.600	0.782	0.612	93.635	71.399	-0.935	9.412	0.766	0.587
5	82.163	62.018	-0.048	7.067	0.827	0.683	83.527	62.802	2.369	6.712	0.822	0.675	82.994	62.729	-0.135	7.287	0.822	0.675
6	77.393	56.732	-0.005	6.260	0.848	0.719	78.112	57.726	2.221	5.429	0.846	0.716	79.175	58.723	-0.297	6.536	0.839	0.705
7	64.529	46.571	-0.324	4.443	0.897	0.805	65.547	47.137	1.732	3.918	0.894	0.800	67.278	48.418	-1.085	4.933	0.887	0.787
8	56.212	40.642	0.139	3.261	0.923	0.852	57.229	41.462	1.588	2.793	0.921	0.848	57.560	42.184	-0.381	3.568	0.919	0.844
9	59.081	42.504	-0.148	3.716	0.915	0.836	59.874	43.639	2.229	3.150	0.913	0.833	61.740	44.822	0.382	3.906	0.906	0.820
10	48.494	33.176	-0.444	2.645	0.943	0.890	50.395	34.449	1.235	2.312	0.939	0.882	51.023	34.913	-0.537	2.819	0.937	0.877
11	39.491	27.641	-0.090	1.680	0.963	0.927	42.295	29.177	0.494	1.708	0.957	0.917	41.423	28.776	0.086	1.721	0.959	0.919
12	38.570	26.551	-0.119	1.611	0.964	0.930	41.354	28.432	**0.064**	1.706	0.959	0.920	40.268	28.044	-0.362	1.687	0.961	0.924
13	34.236	23.648	-0.428	1.387	0.972	0.945	37.320	25.501	0.173	1.337	0.967	0.935	36.769	25.505	-0.828	1.468	0.968	0.936
14	30.586	20.460	-0.120	1.018	0.978	0.956	33.261	22.338	0.216	0.966	0.974	0.948	33.245	22.111	0.009	1.052	0.974	0.948
15	29.315	19.722	-0.142	0.953	0.980	0.960	32.400	21.639	-0.228	1.082	0.975	0.951	32.626	21.500	-0.703	1.155	0.975	0.950
16	31.353	21.097	0.066	0.977	0.977	0.954	34.540	23.056	0.114	1.095	0.972	0.944	33.157	22.566	0.146	1.068	0.974	0.948
17	20.837	14.065	0.090	0.423	0.990	0.980	24.060	16.165	0.466	0.474	**0.986**	**0.973**	25.282	16.251	-0.215	0.570	0.985	0.970
18	**19.775**	**13.155**	-0.067	**0.431**	**0.991**	**0.982**	**24.044**	**15.438**	-0.122	0.498	**0.986**	**0.973**	**24.315**	**15.641**	-0.049	**0.537**	**0.986**	**0.972**
19	22.618	14.999	**-0.006**	0.538	0.988	0.976	28.252	18.117	-0.255	0.732	0.981	0.963	26.992	17.622	-0.652	0.835	0.983	0.966
20	20.712	13.560	0.014	0.440	0.990	0.980	25.010	15.945	0.307	**0.470**	0.985	0.971	24.889	15.991	**-0.027**	0.530	0.985	0.971

Table 3 LM 2 HL

Bayesian Regularization 1 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	123.670	101.729	-0.001	15.981	0.532	0.283	0.000	0.000	0.000	0.000	0.000	0.000	124.875	102.796	2.251	15.669	0.519	0.270
2	115.063	91.203	**0.000**	13.832	0.616	0.379	0.000	0.000	0.000	0.000	0.000	0.000	116.645	92.602	1.760	13.704	0.602	0.363
3	111.807	88.999	**0.000**	13.063	0.643	0.414	0.000	0.000	0.000	0.000	0.000	0.000	111.579	88.811	0.393	13.041	0.645	0.417
4	109.142	86.240	0.001	12.459	0.664	0.441	0.000	0.000	0.000	0.000	0.000	0.000	108.411	86.011	**0.068**	12.440	0.670	0.449
5	106.564	84.779	**0.000**	11.865	0.684	0.467	0.000	0.000	0.000	0.000	0.000	0.000	105.698	83.942	-0.202	11.955	0.690	0.476
6	103.949	82.054	**0.000**	11.294	0.702	0.493	0.000	0.000	0.000	0.000	0.000	0.000	103.927	82.261	-0.033	11.441	0.703	0.494
7	96.180	75.001	**0.000**	9.665	0.752	0.566	0.000	0.000	0.000	0.000	0.000	0.000	96.902	75.407	0.220	9.835	0.748	0.560
8	97.067	75.082	**0.000**	9.844	0.747	0.558	0.000	0.000	0.000	0.000	0.000	0.000	97.950	75.955	-0.001	10.015	0.742	0.550
9	97.293	75.820	**0.000**	9.891	0.746	0.556	0.000	0.000	0.000	0.000	0.000	0.000	97.911	76.433	0.677	9.864	0.742	0.551
10	92.792	70.816	**0.000**	9.009	0.772	0.596	0.000	0.000	0.000	0.000	0.000	0.000	93.407	71.289	0.828	9.017	0.769	0.591
11	87.709	68.477	**0.000**	8.039	0.800	0.639	0.000	0.000	0.000	0.000	0.000	0.000	88.075	68.654	0.713	7.852	0.798	0.637
12	88.063	68.278	**0.000**	8.105	0.798	0.636	0.000	0.000	0.000	0.000	0.000	0.000	88.575	68.784	0.694	7.909	0.795	0.632
13	81.349	61.238	**0.000**	6.915	0.830	0.690	0.000	0.000	0.000	0.000	0.000	0.000	83.400	62.614	-0.096	7.063	0.821	0.674
14	87.899	67.349	**0.000**	8.080	0.799	0.638	0.000	0.000	0.000	0.000	0.000	0.000	88.555	68.214	0.387	8.043	0.795	0.633
15	87.216	66.014	**0.000**	7.967	0.802	0.643	0.000	0.000	0.000	0.000	0.000	0.000	88.470	66.981	-0.181	8.142	0.796	0.633
16	77.220	59.025	**0.000**	6.241	0.849	0.720	0.000	0.000	0.000	0.000	0.000	0.000	78.952	60.559	-0.028	6.437	0.841	0.708
17	82.507	61.944	**0.000**	7.120	0.825	0.681	0.000	0.000	0.000	0.000	0.000	0.000	84.303	63.407	1.329	7.029	0.817	0.667
18	82.242	61.980	**0.000**	7.070	0.826	0.683	0.000	0.000	0.000	0.000	0.000	0.000	82.770	62.504	0.777	6.971	0.824	0.679
19	76.047	56.836	**0.000**	6.065	0.854	0.729	0.000	0.000	0.000	0.000	0.000	0.000	77.341	57.706	1.303	5.802	0.848	0.720
20	***72.565***	***53.185***	**0.000**	***5.506***	**0.868**	**0.753**	0.000	0.000	0.000	0.000	0.000	0.000	**73.564**	**53.683**	1.535	***5.214***	**0.864**	**0.746**

Table 4 BR 1 HL

Bayesian Regularization 2 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	123.144	101.306	0.012	15.841	0.537	0.289	0.000	0.000	0.000	0.000	0.000	0.000	124.188	102.183	2.106	15.533	0.527	0.278
2	112.511	89.933	-0.001	13.231	0.637	0.406	0.000	0.000	0.000	0.000	0.000	0.000	112.815	90.041	1.868	12.929	0.635	0.404
3	101.711	78.802	**0.000**	10.819	0.718	0.515	0.000	0.000	0.000	0.000	0.000	0.000	102.200	79.314	1.639	10.489	0.715	0.511
4	84.170	63.503	**0.000**	7.403	0.817	0.668	0.000	0.000	0.000	0.000	0.000	0.000	85.392	64.368	0.348	7.521	0.811	0.658
5	75.783	56.319	**0.000**	6.009	0.855	0.731	0.000	0.000	0.000	0.000	0.000	0.000	77.039	57.081	1.460	5.804	0.850	0.722
6	62.883	45.851	**0.000**	4.133	0.903	0.815	0.000	0.000	0.000	0.000	0.000	0.000	65.790	48.117	1.394	4.026	0.893	0.797
7	55.124	39.681	**0.000**	3.175	0.926	0.858	0.000	0.000	0.000	0.000	0.000	0.000	57.421	41.325	0.386	3.154	0.920	0.846
8	53.017	37.845	0.002	2.939	0.932	0.868	0.000	0.000	0.000	0.000	0.000	0.000	56.287	40.405	0.129	2.973	0.923	0.852
9	42.768	30.006	**0.000**	1.912	0.956	0.914	0.000	0.000	0.000	0.000	0.000	0.000	45.295	31.716	0.035	2.020	0.951	0.904
10	40.934	28.116	-0.002	1.753	0.960	0.921	0.000	0.000	0.000	0.000	0.000	0.000	42.811	29.320	0.694	1.691	0.956	0.914
11	36.476	25.480	**0.000**	1.391	0.968	0.938	0.000	0.000	0.000	0.000	0.000	0.000	38.502	27.154	0.636	1.363	0.965	0.931
12	32.108	21.638	**0.000**	1.079	0.976	0.952	0.000	0.000	0.000	0.000	0.000	0.000	34.111	23.185	0.377	1.087	0.972	0.945
13	30.335	20.882	-0.002	0.965	0.978	0.957	0.000	0.000	0.000	0.000	0.000	0.000	33.263	22.963	-0.778	1.209	0.974	0.948
14	26.738	18.085	**0.000**	0.748	0.983	0.966	0.000	0.000	0.000	0.000	0.000	0.000	29.836	19.922	-0.224	0.922	0.979	0.958
15	23.979	16.092	**0.000**	0.602	0.986	0.973	0.000	0.000	0.000	0.000	0.000	0.000	27.247	18.012	**0.072**	0.593	0.982	0.965
16	19.419	13.332	**0.000**	0.395	0.991	0.982	0.000	0.000	0.000	0.000	0.000	0.000	22.352	15.152	0.612	**0.294**	0.988	0.977
17	20.180	13.456	**0.000**	0.426	0.990	0.981	0.000	0.000	0.000	0.000	0.000	0.000	23.378	15.271	-0.172	0.583	0.987	0.974
18	18.730	12.494	**0.000**	0.367	0.992	0.984	0.000	0.000	0.000	0.000	0.000	0.000	22.184	14.560	-0.074	0.503	0.988	0.977
19	**14.487**	**9.882**	**0.000**	**0.220**	**0.995**	**0.990**	0.000	0.000	0.000	0.000	0.000	0.000	**18.034**	**11.845**	-0.232	0.337	**0.992**	**0.985**
20	15.880	10.456	0.002	0.265	0.994	0.988	0.000	0.000	0.000	0.000	0.000	0.000	19.684	12.510	-0.623	0.489	0.991	0.982

Table 5 BR 2 HL

Resilient Propagation 1 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	124.487	102.310	-3.435	17.403	0.524	0.274	125.288	102.784	-0.986	16.999	0.518	0.269	125.606	103.664	-1.792	17.300	0.507	0.257
2	116.734	94.877	-1.582	14.840	0.601	0.361	117.813	95.586	0.927	14.483	0.594	0.353	117.727	96.018	-0.751	14.990	0.589	0.347
3	114.100	91.455	-0.765	13.894	0.624	0.390	115.050	92.300	2.771	13.131	0.619	0.383	115.094	92.606	-0.493	14.149	0.613	0.376
4	116.771	94.733	-3.024	15.035	0.601	0.361	117.933	95.300	-0.962	14.820	0.593	0.352	118.339	96.329	-2.485	15.383	0.584	0.341
5	115.528	93.112	-3.851	15.189	0.612	0.375	116.164	93.355	-1.232	14.756	0.609	0.371	116.623	94.863	-3.936	15.692	0.600	0.360
6	109.434	88.589	-0.692	12.935	0.662	0.439	111.161	90.541	2.507	12.386	0.651	0.424	110.717	90.098	0.159	13.120	0.650	0.422
7	106.740	83.973	-0.401	12.170	0.683	0.466	107.110	84.206	1.689	11.777	0.682	0.466	107.397	85.195	**-0.074**	12.408	0.676	0.457
8	112.071	90.495	-0.278	13.243	0.641	0.411	112.804	91.150	2.129	12.794	0.638	0.407	113.580	92.256	0.449	13.486	0.626	0.392
9	104.610	83.974	-0.331	11.841	0.698	0.487	105.115	84.294	1.812	11.429	0.697	0.486	106.357	86.021	-0.768	12.455	0.684	0.467
10	106.513	85.071	-0.705	12.185	0.684	0.468	107.001	85.405	1.285	11.838	0.683	0.467	108.317	87.027	-0.266	12.510	0.669	0.447
11	109.378	87.331	0.192	12.628	0.663	0.439	109.972	87.623	3.256	11.892	0.661	0.437	110.212	88.552	1.041	12.682	0.654	0.428
12	107.829	86.561	0.118	12.323	0.675	0.455	108.007	86.961	1.776	12.143	0.676	0.457	108.383	87.903	1.370	12.356	0.669	0.447
13	109.278	87.340	-0.446	13.008	0.664	0.441	109.355	87.271	1.815	12.564	0.666	0.444	109.372	87.723	-0.251	13.145	0.661	0.437
14	107.978	86.023	-2.035	12.840	0.673	0.453	108.177	86.310	**-0.147**	12.437	0.674	0.455	109.667	87.894	-2.329	13.333	0.658	0.434
15	100.115	78.814	-0.439	10.939	0.728	0.530	99.891	78.490	1.997	10.394	0.732	0.536	99.857	78.994	0.511	10.754	0.728	0.531
16	***94.565***	**73.858**	**-0.234**	***9.635***	**0.762**	**0.581**	***95.389***	**74.571**	2.604	**9.052**	**0.759**	**0.576**	***96.056***	**75.498**	0.959	**9.658**	**0.752**	***0.565***
17	99.085	77.880	-0.848	11.066	0.736	0.541	99.971	78.473	1.691	10.566	0.732	0.535	100.245	79.361	-0.777	11.526	0.727	0.528
18	102.946	82.251	-1.078	11.554	0.709	0.503	103.681	82.944	1.801	10.930	0.706	0.499	103.662	83.070	-0.567	11.674	0.703	0.494
19	103.311	82.618	-1.071	11.790	0.707	0.500	103.081	82.582	0.625	11.441	0.711	0.506	103.725	83.080	-1.696	12.216	0.703	0.494
20	98.186	77.134	-0.808	10.660	0.741	0.548	98.280	77.057	1.757	10.005	0.742	0.550	99.127	78.566	-0.978	11.064	0.733	0.538

Table 6 RPROP 1 HL

Resilient Propagation 2 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	97.599	74.544	0.204	9.959	0.744	0.553	98.110	74.733	2.791	9.378	0.743	0.552	98.283	75.122	1.645	9.915	0.738	0.545
2	90.098	67.621	0.148	8.707	0.787	0.620	89.862	67.319	1.856	8.230	0.790	0.624	91.012	69.142	1.242	8.762	0.781	0.610
3	89.860	68.716	-0.707	9.007	0.789	0.622	90.846	69.300	1.411	8.500	0.785	0.616	91.641	70.170	0.510	9.031	0.778	0.605
4	82.661	62.208	0.208	7.438	0.825	0.680	83.671	62.829	3.249	6.710	0.821	0.674	84.816	64.115	0.608	7.679	0.813	0.661
5	90.923	69.026	-0.295	9.054	0.783	0.613	91.773	69.982	1.675	8.600	0.780	0.608	92.295	70.624	-0.070	9.389	0.774	0.599
6	90.426	68.272	-0.799	9.075	0.785	0.617	90.484	68.169	2.793	8.066	0.787	0.619	92.683	70.567	0.622	9.213	0.772	0.595
7	78.954	57.879	-0.475	6.874	0.841	0.708	79.549	58.180	2.855	5.953	0.840	0.705	80.842	59.455	0.126	6.990	0.832	0.692
8	80.634	59.817	-0.214	7.246	0.834	0.696	81.679	61.224	2.076	6.720	0.830	0.689	83.442	62.563	0.252	7.557	0.820	0.672
9	79.155	58.406	-0.150	6.915	0.841	0.707	79.819	59.103	1.235	6.618	0.839	0.703	80.503	59.525	-0.226	7.165	0.834	0.695
10	75.642	55.858	0.055	6.351	0.856	0.732	77.345	57.080	2.280	5.926	0.849	0.722	77.429	57.028	1.648	6.126	0.847	0.718
11	82.941	61.944	-0.583	7.706	0.823	0.678	82.791	61.521	2.195	6.912	0.825	0.681	84.112	63.259	-0.413	7.845	0.817	0.667
12	80.656	60.664	-0.724	7.512	0.834	0.696	81.325	61.571	1.591	7.002	0.832	0.693	81.411	61.810	0.915	7.413	0.830	0.689
13	79.865	59.050	-0.415	7.222	0.838	0.701	80.310	59.925	1.457	6.705	0.837	0.700	81.341	60.557	0.463	7.321	0.830	0.689
14	74.617	**54.040**	-0.286	6.349	**0.860**	**0.740**	75.561	54.979	1.607	5.815	0.857	0.734	**76.246**	**55.138**	**0.097**	**6.381**	**0.852**	**0.726**
15	77.701	57.432	0.084	6.497	0.847	0.717	78.574	58.022	3.246	5.725	0.844	0.713	79.396	58.523	-0.056	6.786	0.838	0.703
16	77.458	57.049	**-0.070**	6.682	0.848	0.719	78.089	57.492	2.338	6.095	0.846	0.716	78.367	57.481	1.053	6.665	0.843	0.711
17	84.672	62.821	-2.183	8.424	0.815	0.664	85.035	63.100	**-0.348**	7.877	0.814	0.663	87.076	65.219	-2.774	9.005	0.802	0.643
18	**74.522**	54.492	0.157	**6.046**	**0.860**	**0.740**	**75.293**	**54.925**	2.445	**5.519**	**0.858**	**0.736**	76.759	55.900	0.260	6.410	0.850	0.723
19	78.016	57.412	0.181	6.745	0.846	0.715	77.233	56.978	2.578	5.881	0.850	0.723	79.870	59.118	0.508	6.923	0.836	0.700
20	80.878	60.403	-0.182	7.222	0.833	0.694	80.419	60.422	1.656	6.729	0.836	0.699	82.918	61.686	-0.346	7.641	0.822	0.676

Table 7 RPROP 2 HL

Scaled Conjugate Gradient 1 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	124.381	102.609	-4.762	17.649	0.525	0.276	124.884	103.012	-2.276	17.207	0.523	0.273	125.315	103.763	-2.945	17.476	0.510	0.260
2	123.858	102.209	-0.540	16.294	0.530	0.281	125.020	103.175	2.218	15.844	0.521	0.272	125.185	103.509	1.387	16.099	0.512	0.262
3	115.418	92.878	-0.313	14.044	0.613	0.375	115.889	93.050	0.696	14.033	0.612	0.374	115.494	93.135	-1.152	14.418	0.610	0.372
4	124.811	102.249	-0.689	16.269	0.519	0.270	124.937	102.095	1.600	15.752	0.522	0.273	125.852	103.462	0.946	16.174	0.505	0.255
5	109.773	88.262	-1.162	13.151	0.660	0.435	109.769	88.502	0.735	12.835	0.663	0.439	110.127	88.502	-2.140	13.485	0.655	0.429
6	120.264	98.500	-2.790	16.061	0.568	0.322	120.921	99.187	-0.095	15.590	0.564	0.319	121.481	99.870	-1.235	15.974	0.552	0.305
7	123.167	100.534	-0.979	16.106	0.537	0.289	123.975	100.962	1.311	15.737	0.533	0.284	123.837	101.404	0.335	16.055	0.527	0.277
8	**106.913**	**84.852**	-0.548	**12.145**	**0.681**	**0.464**	**106.880**	**84.916**	0.434	12.055	**0.684**	**0.468**	**106.837**	**85.103**	-2.188	12.708	**0.680**	**0.462**
9	122.401	100.132	-1.312	16.069	0.545	0.298	123.784	100.781	0.424	15.999	0.535	0.286	123.685	101.361	-0.685	16.324	0.528	0.279
10	115.306	92.784	-1.268	14.343	0.614	0.377	116.137	93.160	**0.274**	14.239	0.609	0.371	115.923	93.746	-1.396	14.684	0.606	0.367
11	116.769	94.767	-0.991	14.897	0.601	0.361	117.427	95.364	1.272	14.534	0.599	0.358	117.265	95.759	-0.191	14.977	0.594	0.353
12	107.238	85.777	-0.665	12.153	0.679	0.461	107.270	85.802	1.142	11.853	0.681	0.464	107.358	86.151	-0.606	**12.291**	0.676	0.457
13	110.171	88.541	-0.607	13.214	0.657	0.431	110.311	88.678	1.412	12.865	0.659	0.434	110.409	89.287	-1.062	13.673	0.653	0.427
14	113.212	91.255	-1.270	14.043	0.632	0.399	113.955	91.583	0.434	13.866	0.629	0.395	113.718	92.238	-1.607	14.447	0.625	0.391
15	113.074	90.555	**-0.156**	13.561	0.633	0.401	114.154	91.281	1.640	13.412	0.627	0.393	114.429	92.239	-0.582	14.144	0.619	0.383
16	115.144	91.901	-1.163	14.184	0.615	0.378	115.870	92.069	1.097	13.782	0.612	0.374	116.033	93.397	-1.370	14.644	0.605	0.366
17	113.382	91.739	-0.608	13.604	0.630	0.397	114.079	92.050	1.606	13.239	0.627	0.393	114.004	93.191	**-0.023**	13.710	0.623	0.388
18	107.922	86.882	-0.465	12.316	**0.674**	**0.454**	107.560	86.810	1.454	**11.941**	**0.679**	**0.461**	108.162	87.566	-0.911	12.677	0.670	0.449
19	115.047	93.056	-0.944	14.441	0.616	0.380	115.637	93.271	1.038	14.138	0.614	0.378	116.380	94.776	-0.197	14.669	0.602	0.362
20	110.776	89.236	-0.214	13.150	0.652	0.425	111.529	89.813	1.714	12.928	0.649	0.421	111.908	90.991	-0.139	13.584	0.640	0.410

Table 8 SCG 1 HL

Scaled Conjugate Gradient 2 HL																		
	Training						Validation						Testing					
N	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**	**RMSE**	**MAE**	**MBE**	**MAPE**	**R**	**R^2**
1	103.331	81.659	-0.224	11.394	0.707	0.499	103.944	82.288	2.236	10.888	0.705	0.497	104.734	83.209	-0.551	11.803	0.695	0.483
2	82.678	61.619	-0.780	7.435	0.824	0.679	83.762	62.095	1.364	6.821	0.820	0.673	84.357	62.953	-1.113	7.810	0.815	0.665
3	101.667	79.580	-0.529	10.991	0.718	0.515	101.847	80.067	1.756	10.432	0.719	0.517	102.652	81.241	0.236	11.262	0.710	0.504
4	105.910	84.076	-0.457	12.087	0.689	0.474	105.641	83.890	1.408	11.736	0.693	0.481	105.758	84.161	-1.632	12.593	0.688	0.473
5	73.836	53.371	-0.299	5.812	0.863	0.744	75.708	55.017	1.598	5.458	0.856	0.733	76.509	55.169	-0.386	6.047	0.851	0.724
6	98.159	76.157	-0.775	10.526	0.741	0.548	98.304	76.336	0.950	10.250	0.742	0.550	98.209	76.487	-1.260	10.892	0.739	0.546
7	88.709	67.399	-1.310	8.810	0.794	0.631	89.356	67.775	0.780	8.234	0.792	0.628	90.440	68.700	-2.257	9.415	0.784	0.615
8	116.844	93.658	-0.551	14.535	0.600	0.360	117.863	94.512	1.644	14.323	0.594	0.353	117.817	94.685	-0.380	14.872	0.588	0.346
9	103.149	81.730	**-0.219**	11.113	0.708	0.501	103.218	81.794	2.514	10.448	0.710	0.504	103.760	82.639	-0.704	11.522	0.702	0.493
10	114.366	92.241	-0.250	13.958	0.622	0.387	114.703	92.599	1.700	13.695	0.622	0.387	115.622	93.578	**0.257**	14.298	0.608	0.370
11	116.071	94.005	-0.600	14.510	0.607	0.369	116.791	94.526	1.851	14.095	0.604	0.365	117.114	95.183	-0.506	14.860	0.595	0.354
12	73.700	53.855	-0.246	5.663	0.863	0.745	74.434	54.810	1.642	5.067	0.861	0.742	76.143	55.110	0.447	5.797	0.853	0.727
13	74.101	53.581	-0.712	5.868	0.862	0.743	75.090	54.505	1.356	5.413	0.859	0.737	76.789	55.160	-0.531	6.081	0.850	0.722
14	78.631	58.170	-0.732	6.768	0.843	0.710	79.196	58.237	0.646	6.287	0.841	0.708	81.394	59.832	-0.603	6.927	0.829	0.688
15	101.541	79.435	-0.278	11.179	0.719	0.517	101.820	79.806	1.938	10.721	0.719	0.517	102.585	80.981	-0.619	11.705	0.710	0.505
16	115.788	93.716	-0.490	14.314	0.609	0.371	116.517	94.380	1.222	14.209	0.606	0.368	117.575	95.470	-0.081	14.745	0.590	0.349
17	105.501	83.874	-0.496	11.558	0.692	0.478	105.055	83.531	1.467	11.157	0.697	0.486	106.063	84.452	0.155	11.678	0.686	0.470
18	108.365	86.123	-0.949	12.781	0.671	0.450	108.101	85.725	1.115	12.370	0.675	0.456	109.601	87.710	-1.515	13.431	0.659	0.434
19	**69.330**	**51.489**	-0.413	**4.955**	**0.880**	**0.775**	**69.463**	**52.023**	**0.664**	**4.608**	**0.881**	**0.775**	**71.122**	**52.914**	0.291	**5.056**	**0.873**	**0.762**
20	87.094	66.074	-0.286	7.960	0.803	0.644	87.809	66.771	1.545	7.504	0.800	0.641	89.433	67.825	-0.701	8.646	0.789	0.623

Table 9 SCG 2 HL

Time [s]				
N	LM 2 HL	BR 2 HL	RPROP 2 HL	SCG 2 HL
1	2.546	3.142	7.132	4.184
2	2.243	9.546	7.956	9.189
3	1.381	4.991	4.933	4.165
4	5.435	21.805	8.677	2.703
5	20.034	55.226	6.639	18.694
6	9.086	35.587	5.447	4.966
7	24.009	78.431	10.903	8.57
8	9.683	91.192	11.105	1.119
9	15.822	120.874	12.138	3.557
10	29.682	99.202	12.667	2.169
11	23.896	194.357	10.664	1.819
12	70.022	240.681	11.95	24.24
13	45.496	299.577	14.446	78.181
14	73.443	354.875	14.655	54.532
15	84.544	432.693	15.56	6.252
16	75.391	537.919	14.876	2.972
17	260.202	679.136	9.941	3.694
18	277.037	839.569	17.033	3.737
19	222.367	1103.891	17.809	33.162
20	276.806	1499.604	17.41	12.923

Table 10 Architecture time

Epochs				
N	LM 2 HL	BR 2 HL	RPROP 2 HL	SCG 2 HL
1	150	219	744	226
2	166	684	985	613
3	60	212	565	247
4	182	634	1000	162
5	489	1000	668	1000
6	188	630	531	254
7	321	1000	946	400
8	124	1000	1000	53
9	144	1000	1000	151
10	220	639	1000	88
11	127	1000	788	69
12	324	1000	924	964
13	165	1000	1000	940
14	210	1000	1000	616
15	201	1000	1000	203
16	147	1000	1000	98
17	415	1000	607	113
18	356	1000	1000	109
19	204	998	1000	941
20	190	1000	1000	380

Table 11 Architecture epochs

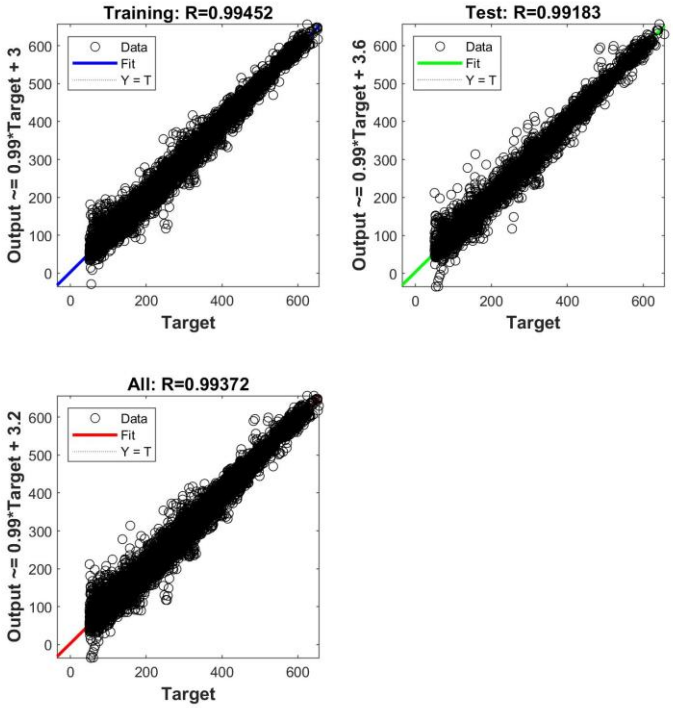


Fig. 118 Matlab regression plot BR 2 HL 7-19-57-1-1

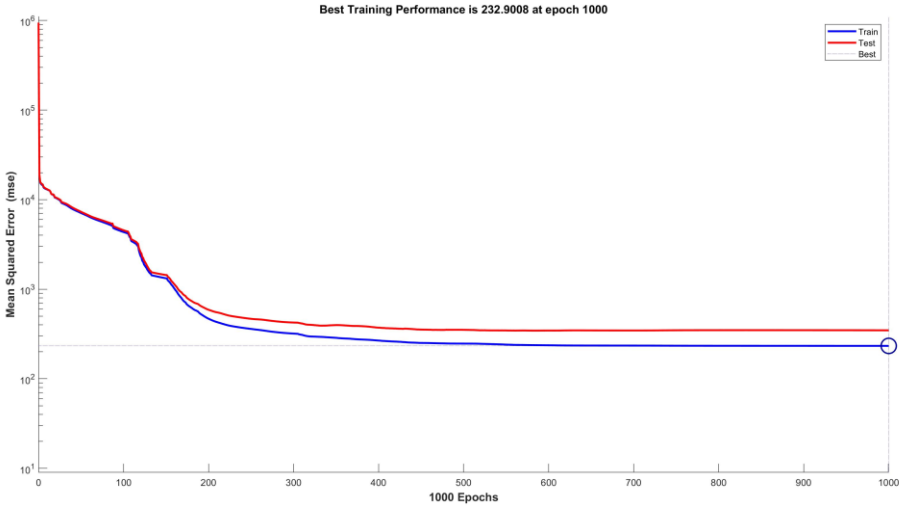


Fig. 119 Matlab performance plot BR 2 HL 7-19-57-1-1

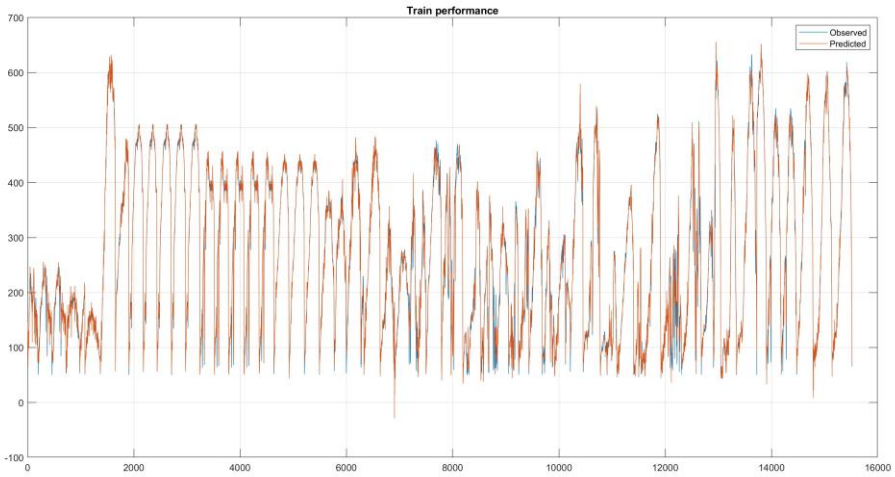


Fig. 120 Training forecast BR 2 HL 7-19-57-1-1

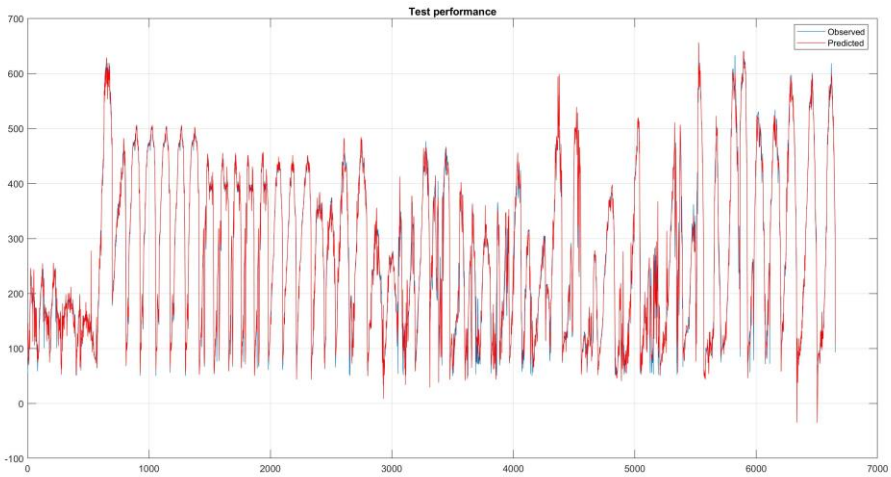


Fig. 121 Testing forecast BR 2 HL 7-19-57-1-1

5 Discussion

The outcome highlighted how architectures with two hidden layers can overcome their respective one layered architecture, and among the algorithms, LM and BR are the most promising. Despite the training velocity of LM 7-18-1-1, the best model to be considered is BR 7-19-57-1-1 thanks to its accuracy. These results are in agreement with several authors who conducted researches on ANN applications in many fields and who found good accuracy for the BR algorithm. Khosravi et al., have investigated which model, among the eleven present in MATLAB's Deep Learning Toolbox, best-fit wind speed and direction in Iran. In particular, their results have shown that BR had the best RMSE and R values, followed by LM, RPROP and SCG, but it also required the biggest computational time with over 5 minutes. LM continued to be the second most promising model in terms of time with about 2 mins; RPROP and SCG algorithms were the fastest, with 7 and 4 seconds, respectively. These performances continued in the epochs domain, where BR finished at 1059, LM at 423, RPROP at 361, and SCG at 115 [80]. Yacef et al., have studied the prediction of daily global solar irradiation, comparing BR and LM. The first approach has led to an increase in accuracy, with a diminution of RMSE from 17.06 to 12.80, MBE from 4.70 to 3.83, and MAE from 7.10 to 6.35, and an increase of R from training to testing phase. At the same time, LM correlation coefficient was decreased, and thus indicated over-fitting phenomena [81]. Alomari et al., have studied the power production in photovoltaic power plants using both LM and BR algorithms fed with real-time weather data. After many trainings, validation and testing experiments, they have decided that the BR algorithm, with 27 hidden neurons was the best model which yields on average RMSE= 0.0706 and R=0.9660 [82]. Madagouda et al., have evaluated LM, BR, SCG and the descendent gradient (DG) algorithm for localization in a wireless sensor network. They discovered that an architecture composed of nine inputs, twelve hidden neurons, and two outputs produced optimal results. BR has shown lower average errors, however it also required the longest training time, it is followed by LM, RP and SCG. The authors concluded with the suggestion of using BR for offline training and LM for online/real-time training [83].

However, in literature, there are many different architecture and learning method comparisons, but due to the nonlinearity hidden relationships of the world and input variables, there is not one common strand of thought or unique solution.

Yaïci et al., have proposed LM as the best algorithm in their demonstration of the effectiveness of ANN approach, by decreasing input number/the input numbers, on solar energy system performance. The study is conducted analyzing the statistics of the LM, BR and SCG algorithms with 16, 18, 20 and 22 hidden neurons. The results proved that prediction is attainable by decreasing the number of inputs and that the best performances are achieved with LM with $R^2 = 0.9999$ and $MAPE = 2.1910\%$ with 20 neurons over BR whose $R^2 = 0.9998$ and $MAPE = 2.9015\%$ with 16 neurons [84]. Mohd-Safar et al., have studied short-term localized weather forecasting using ANNs trained with LM, BR and SCG. In the first algorithm, MAE and RMSE yield the lowest, R the greatest; SCG carries the fastest time but it didn't produce a good convergence; despite the fact, BR took the longest time, it has converged faster [85]. Mohanraj et al., have discussed the performance of a solar-assisted heat pump with three different variants of the learning algorithm (LM, SCG, and Pola-Ribiere conjugate gradient (CGP)). The outcome has shown velocity and accuracy of the LM with 10 hidden neurons whose $R^2 = 0.999$ is maximum and RMS is minimum [86]. Farkas et al., have modelled an ANN on flat-plate solar collectors employing various methods. The ANNs were trained for three different collectors with the algorithms provided by MATLAB. Among these methods, LM has demonstrated that it produces the most accurate and valid results, showing an average deviation of $0.9°C$ [87].

6 Conclusion

This thesis aims to provide an accurate artificial neural network model for energy production, in terms of temperature, in solar dish concentrators. Due to the vast number of global variables and hidden relationships, literature can not provide a priori assumption on which model, architecture or algorithm is a better fit for the study. In this work, the investigation is made through the backpropagation neural network using different learning methods such as Levenberg-Marquardt, Bayesian Regularization, Residual Backpropagation and Scaled Conjugate Gradient. For a more conclusive result many different shapes of architecture have been considered, from single to double hidden layers and from one to twenty hidden neurons. Results have highlighted how increasing these two numbers can improve the overall accuracy of all architectures. This behaviour is reflected in the Taylor diagrams, where single-layered architectures are very close to each other and far from the observed model. In contrast, double hidden layered architectures are stretched more and are capable of getting closer to the objective. The proposed ANNs have been trained with seven meteorological parameters: humidity, air temperature, pressure, wind velocity, wind direction, global radiation, and rain; all these weather elements were taken from the Energy Center, Turin. The analysis of the most promising architecture has been conducted adhering to set protocols and correct procedures in the training, validation and testing phases and by normalizing data. To ensure transparency in the area of forecasting RMSE, MAE, MBE, MAPE, R, and R^2 errors and algorithm stopping criteria, time and epochs have all been taken into account. The two most promising architectures are respectively BR 2 HL, 7-19-57-1-1, and LM 2 HL, 7-18-54-1-1. Despite a large amount of time needed for training, BR 2 HL has demonstrated to be the most accurate model. The training phase is characterized by $RMSE \cong 14.487$; $MAE \cong 9.882$; $MBE \cong 0$; $MAPE \cong 0.220\%$; $R \cong 0.995$; $R^2 \cong 0.990$ and the testing phase by $RMSE \cong 18.034$; $MAE \cong 11.845$; $MBE \cong -0.232$; $MAPE \cong 0.337\%$; $R \cong 0.992$; $R^2 \cong 0.985$. The overall results have suggested that artificial neural networks are a strong, reliable, and important tool for prediction and they may help researchers to forecast trends of their studies in many different fields. For the study of dish solar concentrator located at Energy Center, BR 2 HL using 7-19-57-1-1 architecture is the best prediction model that may be used for conducting a limited number of experiments, under the specific input conditions.

7 Acknowledgement

In the first place, I would like to thank my supervisor Davide Papurello for the possibility he gave me to work and to develop my studying in his subject, and for his help along the path. Unfortunately, the Covid outbreak did not allow me either to work directly with him or to attend the Energy Center, which would have been a pleasure to enrich my knowledge.

I want to give a heartfelt thanks to my family, my parents who gave me the opportunity to study at Politecnico, and who took care of me despite the distance. A special thanks to my mom Nadia for all the calls, the messages, AND for the food supplies which reminded me home. A special thanks to my sister Anna who supported me during the darkest periods of all this trip and always believes in me by pushing towards a bright future. The most special thanks goes to my father Carlo who made me realize how much nature is beautiful, how much discovery is fascinating, how much research is interesting, and how much important it is pursuing my dreams.

The final special thanks goes to all the friends who accompanied me on this adventure, with smiles in moments of joy and with cheering up in moments of sadness.

Thank you all!

8 References

[1] T. H. Le, Y. Chang, and D. Park, Renewable and Nonrenewable Energy Consumption, Economic Growth, and Emissions: International Evidence. The Energy Journal. 41 (2020) 73-92. https://doi.org/10.5547/01956574.41.2.thle

[2] A. Gupta, Chapter 1 - Climate Change and Kyoto Protocol: An Overview, Handbook of Environmental and Sustainable Finance. Academic Press (2016) 3-23 ISBN 9780128036150, https://doi.org/10.1016/B978-0-12-803615-0.00001-7

[3] https://www.ipcc.ch/report/ar5/wg3/

[4] M. Guidolin, R. Guseo, The German energy transition: Modeling competition and substitution between nuclear power and Renewable Energy Technologies. Renewable and Sustainable Energy Reviews 60 (2016) 1498-1504, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2016.03.022

[5] C. Marchetti, Society as a learning system: Discovery, invention, and innovation cycles revisited. Technological Forecasting and Social Change 18 (1980) 267-282, ISSN 0040-1625, https://doi.org/10.1016/0040-1625(80)90090-6

[6] V. Ramiah, M. Gangemi, M. Liu, Chapter 2 - Environmental Policies Post the Kyoto Protocol on Climate Change: Evidence from the US and Japan. Handbook of Environmental and Sustainable Finance, Academic Press (2016) 25-54, ISBN 9780128036150, https://doi.org/10.1016/B978-0-12-803615-0.00002-9.

[7] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52019DC0640

[8]https://ec.europa.eu/eurostat/statistics-xplained/index.php?title=Renewable_energy_statistics

[9] K. Butti, J. Perlin, A Golden Thread: 2500 Years of Solar Architecture and Technology. Cheshire Books (1980) 76 and following ISBN 0442240058

[10] V.G. Belessiotis and E. Papanicolaou, History of Solar Energy. Comprehensive Renewable Energy 3 (2012) 85-102 https://doi.org/10.1016/B978-0-08-087872-0.00303-6

[11] C. Silvi, The Pioneering Work on Linear Fresnel Reflector Concentrators (LFCs) in Italy, Proceedings of the Congress SolarPaces 2009, Electricity, fuels and clean water powered by the sun, Berlin 15-18 September 2009

[12] G. Francia, Pilot plants of solar steam generating stations. Solar Energy, 12/1 (1968) 51-64, ISSN 0038-092X, https://doi.org/10.1016/0038-092X(68)90024-8

[13] C. Silvi, Eurelios fu un abbaglio? Sapere, (2011) 36-45, Giugno, 2011

[14] W. S. Mcculloch, and W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5 (1943) 115-133 https://doi.org/10.1007/BF02478259

[15] D.O. Hebb, The Organization of Behavior, Wiley New York (1949) doi: 10.1016/s0361-9230(99)00182-3.

[16] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65 (1958) 386–408 https://doi.org/10.1037/h0042519

[17] B. Widrow, M.E. Hoff, Adaptive Switching Circuits. IRE WESCON Convention Record, 4:96-104, August 1960

[18] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[19] J. D. Cowan, A mathematical theory of central nervous activity. Ph. D. Dissertation, Univ. London, 1967

[20] M. Minsky and S. Papert, A Review of "Perceptrons: An Introduction to Computational Geometry". The M.I.T. Press, Cambridge, Mass., 1969

[21] D.E. Rumelhart, G.E. Hinton & R.J. William, Learning representations by back-propagating errors. Nature 323 (1986) 533-536 https://doi.org/10.1007/BF02124743

[22] W.S. Alaloul and A.H. Qureshi, Data Processing Using Artificial Neural Networks (2020) DOI: 10.5772/intechopen.91935

[23] M.T. Islama, N. Hudaa, A.B. Abdullahb, R. Saidur, A comprehensive review of state-of-the-art concentrating solar power (CSP) technologies: Current status and research trends. Renewable and Sustainable Energy Reviews 91 (2018) 987-1018 https://doi.org/10.1016/j.rser.2018.04.097

[24] M. J. Emes, M. Arjomandi, G.J. Nathan, Effect of heliostat design wind speed on the levelised cost of electricity from concentrating solar thermal power tower plants. Solar Energy 115 (2015) 441–451 https://doi.org/10.1016/j.solener.2015.02.047

[25] A. Pfahl, J. Coventry, M. Röger, F. Wolfertstetter, J. F. Vásquez-Arango, F. Gross, M. Arjomandi, P. Schwarzbözl, M. Geiger, P. Liedke, Progress in heliostat development, Solar Energy, 152 (2017), 3-37, https://doi.org/10.1016/j.solener.2017.03.029

[26] C.K. Ho and B.D. Iverson, Review of Central Receiver Designs for High-Temperature Power Cycles. SolarPACES 2012, Marrakech, Morocco, September 11-14. 2012 http://dx.doi.org/10.1016/j.rser.2013.08.099

[27] P. K. Falcone, A handbook for solar central receiver design. United States. https://doi.org/10.2172/6545992

[28] T. Fend, B. Hoffschmidt, R. Pitz-Paal, O. Reutter and P.Rietbrock, Porous materials as open volumetric solar receivers: Experimental determination of thermophysical and heat transfer properties. Energy 29 (2004) 823–833 doi:10.1016/S0360-5442(03)00188-9

[29] R. Buck, S. Giuliano, R. Uhlig, 16 - Central tower systems using the Brayton cycle, Energy, Advances. Concentrating Solar Thermal Research and Technology, 2017, 353-382, ISBN 9780081005163, https://doi.org/10.1016/B978-0-08-100516-3.00016-2

[30] K.R. Zada, M.B. Hyder, M.K. Drost, B. M. Fronk, Numbering-Up of Microscale Devices for Megawatt-Scale Supercritical Carbon Dioxide Concentrating Solar Power Receivers. Journal of Solar Energy Engineering 138 (2016)

[31] T. L'Estrange, E. Truong, C. Rymal, E. Rasouli, V. Narayanan, S. Apte and K. Drost, High flux microscale solar thermal receiver for supercritical carbon dioxide cycles. Proceedings of the ASME 2015 13th International Conference on Nanochannels, Microchannels, and Minichannels 2015 July 6-9, San Francisco, California, USA; https://doi.org/10.1115/ICNMM2015-48233

[32] C.K. Ho, Advances in central receivers for concentrating solar applications, Solar Energy, 152 (2017) 38-56 ISSN 0038-092X https://doi.org/10.1016/j.solener.2017.03.048

[33] B.D. Kelly, Advanced Thermal Storage for Central Receivers with Supercritical Coolants. United States (2010) https://doi.org/10.2172/981926

[34] M. Neber and H. Lee, Silicon Carbide Solar Receiver for Residential Scale Concentrated Solar Power. Proceedings of the ASME 2012 International Mechanical Engineering Congress and Exposition. Houston, Texas, USA. (2012) 371-376 https://doi.org/10.1115/IMECE2012-88372

[35] Z. Liao, A. Faghri, Thermal analysis of a heat pipe solar central receiver for concentrated solar power tower, Applied Thermal Engineering 102 (2016) 952-960, ISSN 1359-4311 https://doi.org/10.1016/j.applthermaleng.2016.04.043

[36] S. Obrey, J. SteOenheim, T. McBride, M. Hehlen, R. Reid and T. Jankowski, High Temperature Heat Pipe Receiver for Parabolic Trough Collectors. (2015)

[37] K. Jiang, X. Du, Y. Kong, C. Xu, X. Ju, A comprehensive review on solid particle receivers of concentrated solar power, Renewable and Sustainable Energy Reviews, 116 (2019) 109463, ISSN 1364-0321 https://doi.org/10.1016/j.rser.2019.109463

[38] P. Chang, Y. Tang, Y. Huang, J. Zhao, C. Li and C. Yuan, Optimal Operation of Concentrating Solar Power Station in Power System with High Penetration of Photovoltaic Generation (2019) IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Macao, China (2019) 1-6 doi: 10.1109/APPEEC45492.2019.8994347

[39] I. Sarbu, C. Sebarchievici, A comprehensive review of thermal energy storage. Sustainability 10 (2018) https://doi.org/10.3390/su10010191

[40] A. Stamatiou, M. Obermeyer, L. J. Fischer, P. Schuetz, J. Worlitschek, Investigation of unbranched, saturated, carboxylic esters as phase change materials, Renewable Energy 108 (2017) 401-409 ISSN 0960-1481 https://doi.org/10.1016/j.renene.2017.02.056

[41] G. Alva, Y. Lin, G. Fang, An overview of thermal energy storage systems, Energy 144 (2018) 341-378 ISSN 0360-5442 https://doi.org/10.1016/j.energy.2017.12.037

[42] CASIO (2013). What the Duck Curve Tells us about Managing a Green Grid

[43] Md T. Islam, N. Huda, A.B. Abdullah, R. Saidur, A comprehensive review of state-of-the-art concentrating solar power (CSP) technologies: Current status and research trends, Renewable and Sustainable Energy Reviews, 91 (2018) 987-1018 ISSN 1364-0321 https://doi.org/10.1016/j.rser.2018.04.097

[44] N. El Gharbi, H. Derbal, S. Bouaichaoui, N. Said, A comparative study between parolic trough collector and linear Fresnel reflector technologies, Energy Procedia (2011) 565-572 ISSN 1876-6102 https://doi.org/10.1016/j.egypro.2011.05.065

[45] G. Zhang, B. E. Patuwo, M. Y. Hu, Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting 14 (1998) 35–62 https://doi.org/10.1016/S0169-2070(97)00044-7

[46] T.Y. Pan, T.Y. Yang, H.C. Kuo, Y.C. Tan, J.S. Lai, T.J.Chang , C.S. Lee, K. H. Hsu, Improvement of Statistical Typhoon Rainfall Forecasting with ANN-Based Southwest Monsoon Enhancement. Terrestrial, Atmospheric and Oceanic Sciences 22 (2011) 633-645 doi: 10.3319/TAO.2011.07.04.01(TM)

[47] S. Sharma, S. Sharma, A. Athaiya, Activation Functions In Neural Networks. International Journal of Engineering Applied Sciences and Technology 4/12 (2020) 310-316 ISSN No. 2455-2143,

[48] C. Lv et al, Levenberg-Marquardt Backpropagation Training of Multilayer Neural Networks for State Estimation of A Safety Critical Cyber-Physical System, IEEE Transactions on Industrial Informatics 14/8 (2018) 3436-3446 doi: 10.1109/TII.2017.2777460

[49] D.W. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear Parameters, Journal of the Society for Industrial and Applied Mathematics, 11/2 (1963) 431–441 https://doi.org/10.1137/0111030

[50] F. Burden and D. Winkler, Bayesian Regularization of Neural Networks, Livingstone D.J. (eds) Artificial Neural Networks. Methods in Molecular Biology, vol 458 Humana Press (2008) https://doi.org/10.1007/978-1-60327-101-1_3

[51] F. Dan Foresee and M.T. Hagan, Gauss-Newton approximation to bayesian learning. Proceedings of the International Conference on Neural Networks, (1997) 1930-1935 DOI:10.1109/ICNN.1997.614194

[52] M. Riedmiller H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm. IEEE International Conference on Neural Networks San Francisco, CA, USA, (1993) 586-591 DOI:10.1109/ICNN.1993.298623

[53] F. Günther and S. Fritsch, neuralnet: Training of Neural Networks. The R Journal 2 (2010) 30-38 ISSN 2073-4859 doi:10.32614/RJ-2010-006

[54] H. Chel, A. Majumder, and D. Nandi, Scaled conjugate gradient algorithm in neural network based approach for handwritten text recognition, inTrends in computer science, engineering and information technology communications in computer and information science, Springer Berlin 204 (2011) 196-210 https://doi.org/10.1007/978-3-642-24043-0_21

[55] M. F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks, 6/4 (1993) 525-533, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(05)80056-5

[56] M. S. Başar, and H. Küçükönder, Measuring the Correlation between Commercial and Economic States of Countries (B2G Relations) and the E-Government Readiness Index by Using Neural Networks, Open Journal of Business and Management 2. (2014) 110-115. doi:10.4236/ojbm.2014.22014

[57] K. E. Taylor, Summarizing multiple aspects of model performance in a single diagram, Journal of Geophysical Research: Atmospheres 106/7 (2001) 7183-7192 doi:10.1029/2000JD900719

[58] W.W. Hsieh, B. Tang, Applying Neural Network Models to Prediction and Data Analysis in Meteorology and Oceanography, Bulletin of the American Meteorological Society 79/9 (1998) 1855-1870, doi:10.1175/1520-0477(1998)079<1855:ANNMTP>2.0.CO;2

[59] C. W. Dawson and R. Wilby, An artificial neural network approach to rainfall-runoff modelling, Hydrological Sciences Journal, 43/1 (1998) 47-66, doi:10.1080/02626669809492102

[60] K. C. Luk, J.E. Ball, A. Sharma, An application of artificial neural networks for rainfall forecasting, Mathematical and Computer Modelling, 33/6–7 (2001) 683-693 https://doi.org/10.1016/S0895-7177(00)00272-7

[61] S. Chattopadhyay and G. Chattopadhyay, Identification of the best hidden layer size for three-layered neural net in predicting monsoon rainfall in India, Journal of Hydroinformatics 10/2 (2008) 181-188 doi:10.2166/HYDRO.2008.017

[62] P. Singh and B. Borah, Indian summer monsoon rainfall prediction using artificial neural network. Stochastic Environmental Research and Risk Assessment 27 (2013) 1585–1599 DOI: 10.1007/s00477-013-0695-0

[63] T.L. Lee, Back-propagation neural network for long-term tidal predictions, Ocean Engineering, 31/2 (2004) 225-238 https://doi.org/10.1016/S0029-8018(03)00115-X

[64] S. Chaudhuri and S. Chattopadhyay, Neuro-computing based short range prediction of some meteorological parameters during the pre-monsoon season. Soft Comput 9 (2005) 349-354 https://doi.org/10.1007/s00500-004-0414-3

[65] J. Vandegriff, K. Wagstaff, G. Ho, J. Plauger, Forecasting space weather: Predicting interplanetary shocks using neural networks, Advances in Space Research, 36/12 (2005) 2323-2327 https://doi.org/10.1016/j.asr.2004.09.022

[66] R. Bustami, N. Bessaih, C. Bong, S. Suhaili, Artificial Neural Network for Precipitation and Water Level Predictions of Bedup River, IAENG International Journal of Computer Science 34/2 (2007) 228-233

[67] M. Hayati and Z. Mohebi, Application of Artificial Neural Networks for Temperature Forecasting, World Academy of Science, Engineering and Technology 28 (2007) 275-279 doi.org/10.5281/zenodo.1070987

[68] H. Ali, L. T. Shui and G. Ehsan, Estimation of Yield Sediment Using Artificial Neural Network at Basin Scale, Australian Journal of Basic and Applied Sciences 4/7 (2010) 1668-1675

[69] M. M. Raju, R. K. Srivastava, D. C. S. Bisht, H. C. Sharma and A. Kumar, Development of Artificial Neural-Network-Based Models for the Simulation of Spring Discharge; Advances in Artificial Intelligence (2011) https://doi.org/10.1155/2011/686258

[70] Sawaitul Sanjay D., Prof. Wagh K. P., Dr. Chatur P. N., Classification and prediction of future weather by using back propagation algorithm-an approach, International Journal of Emerging Technology and Advanced Engineering, 2/1 (2012) 110-113

[71] A. K. Yadav, S.S. Chandel, Solar radiation prediction using Artificial Neural Network techniques: A review, Renewable and Sustainable Energy Reviews, 33 (2014) 772-781 https://doi.org/10.1016/j.rser.2013.08.055

[72] https://modis.gsfc.nasa.gov/data/dataprod/mod07.php

[73] J. Feng, W. Wang, W and J. Li, An LM-BP Neural Network Approach to Estimate Monthly-Mean Daily Global Solar Radiation Using MODIS Atmospheric Products. Energies 11 (2018) https://doi.org/10.3390/en11123510

[74] A. Sarkar, P. Pandey, River Water Quality Modelling Using Artificial Neural Network Technique, Aquatic Procedia 4 (2015) 1070-1077 https://doi.org/10.1016/j.aqpro.2015.02.135

[75] H. Maleki, A. Sorooshian, G. Goudarzi, Z. Baboli, Y.T. Birgani and M. Rahmati, Air pollution prediction by using an artificial neural network model, Clean Technologies and Environmental Policy 21 3 (2019) doi10.1007/s10098-019-01709-w

[76] J. Ballestrín, E. Carra, J. Alonso-Montesinos, G. López, J. Polo, A. Marzo, J. Fernández-Reche, J. Barbero, F.J. Batlles, Modeling solar extinction using artificial neural networks. Application to solar tower plants, Energy 199 (2020) ISSN 360-5442, https://doi.org/10.1016/j.energy.2020.117432

[77] Z. Mosaffaei A. Jahani, M.A.Z. Chahouki, et al., Soil texture and plant degradation predictive model (STPDPM) in national parks using artificial neural network (ANN). Model. Earth Syst. Environ. 6 (2020) 715–729 https://doi.org/10.1007/s40808-020-00723-y

[78] G.R. Ruiz, C.F. Bandera, Validation of Calibrated Energy Models: Common Errors. Energies 10 (2017) https://doi.org/10.3390/en10101587

[79] M. Theristis, V. Venizelou, G. Makrides, G. E. Georghiou, Chapter II-1-B - Energy Yield in Photovoltaic Systems, in Handbook of Photovoltaics (Third Edition), Academic Press, 2018 671-713, ISBN 9780128099216, https://doi.org/10.1016/B978-0-12-809921-6.00017-3

[80] A. Khosravi, R.N.N. Koury, L. Machado, J.J.G. Pabon, Prediction of wind speed and wind direction using artificial neural network, support vector regression and adaptive neuro-fuzzy inference system, Sustainable Energy Technologies and Assessments 25 (2018) 146-160 https://doi.org/10.1016/j.seta.2018.01.001

[81] R. Yacef, M. Benghanem, A. Mellit, Prediction of daily global solar irradiation data using Bayesian neural network: A comparative study, Renewable Energy 48 (2012) 146-154 https://doi.org/10.1016/j.renene.2012.04.036

[82] M.H. Alomari, O. Younis and S. M. A. Hayajneh, "A Predictive Model for Solar Photovoltaic Power using the Levenberg-Marquardt and Bayesian Regularization Algorithms and Real-Time Weather Data, International Journal of Advanced Computer Science and Applications 9/1 (2018) http://dx.doi.org/10.14569/IJACSA.2018.090148

[83] B. Madagouda and R. Sumathi, Artificial Neural Network Approach using Mobile Agent for Localization in Wireless Sensor Networks, Advances in Science, Technology and Engineering Systems Journal 6 (2021) 1137-1144 DOI:10.25046/aj0601127

[84] W. Yaïci, M. Longo, E. Entchev and F. Foiadelli, Simulation Study on the Effect of Reduced Inputs of Artificial Neural Networks on the Predictive Performance of the Solar Energy System, Sustainability 9 (2017) https://doi.org/10.3390/su9081382

[85] N. Z. Mohod Safar, D. Ndzi, I. Kagalidis, Y. Yang and A. Zakaria, Short-term localized weather forecasting by using different artificial neural network algorithm in tropical climate. Proceedings of SAI Intelligent Systems Conference 2016, DOI 10.1007/978-3-319-56991-8_35

[86] M. Mohanraj, S. Jayaraj, C. Muraleedharan, Performance prediction of a direct expansion solar assisted heat pump using artificial neural networks, Applied Energy, 86/9 (2009) 1442-1449, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2009.01.001

[87] I. Farkas, P. Géczy-Víg, M. Tóth, Neural Network Modelling of Solar Collectors, IFAC Proceedings 34/26 (2001) 55-60 https://doi.org/10.1016/S1474-6670(17)33632-7

9 Appendix

```
close all
clear all
clc
rng('default')
```

Loading Meteo

```
Base = 'C:\Users\leont\OneDrive\Desktop\SCRIPT TESI\Dati Meteo';
List = dir(fullfile(Base, '*.xlsx'));
Result = cell(1, numel(List));
datetime.setDefaultFormats('default','dd/MM/yyyy HH:mm:ss')
for k = 1:numel(List)
    File   = fullfile(Base, List(k).name);
    MeteoResult = readtable(File,"VariableNamingRule",'preserve');
    MeteoDate(k)= MeteoResult.Data(1);
    StartingTimeMeteo(k)=MeteoResult.UTCTime(1);
    EndingTimeMeteo(k)=MeteoResult.UTCTime(end);
    MeteoResult=table2timetable(MeteoResult(:,3:10));
    MeteoDailyFile{k}=MeteoResult;
end
```

Loading Tower

```
Base = 'C:\Users\leont\OneDrive\Desktop\SCRIPT TESI \Dati Tower';
List = dir(fullfile(Base, '*.xlsx'));
Result = cell(1, numel(List));
Nanflag=0;
for k = 1:numel(List)
    File   = fullfile(Base, List(k).name);
    TowerResult = readtable(File,"VariableNamingRule",'preserve');
    TowerDate(k)=TowerResult.Date(1);
    StartingTimeTower(k)=TowerResult.UTCTime(1);
    EndingTimeTower(k)=TowerResult.UTCTime(end);
    TowerDailyFile{k}=TowerResult;
    p=1;
    for l=1:height(TowerDailyFile{k})
          if ((TowerResult.Treatt_0(l)>50)&(TowerResult.Treatt_0(l)<1800))
              NewTowerResult(p,:)=TowerResult(l,:);
              p=p+1;
          end
    end
    if (p~=1)
        NewTowerDailyFile{k-Nanflag}=table2timetable(NewTowerResult(:,3:12));
        clear NewTowerResult
    else
        Nanflag=Nanflag+1;
    end
end
```

Date Measurements

```
if (MeteoDailyFile{1}.Time(1)<TowerDailyFile{1}.Time(1))
    StartingDay=MeteoDailyFile{1}.Time(1);
    MeteoFlag=1;
else
    StartingDay=TowerDailyFile{1}.Time(1);
end
if (MeteoDailyFile{end}.Time(1)<TowerDailyFile{end}.Time(1))
    EndingDay=MeteoDailyFile{end}.Time(1);
else
```

```
    EndingDay=TowerDailyFile{end}.Time(1);
end
DaysCounter=round(days(EndingDay-StartingDay));
MeteoDayMeasurements(DaysCounter)=0;
TowerDayMeasurements(DaysCounter)=0;
if MeteoFlag==1
    MeteoDayMeasurements(1)=1;
else
    TowerDayMeasurement(1)=1;
end
for i=1:DaysCounter
    for j=1:length(MeteoDate)
        if datetime(MeteoDate(j))==StartingDay+i
            MeteoDayMeasurements(i+1)=1;
        end
    end
    for h=1:length(TowerDate)
        if datetime(TowerDate(h))==StartingDay+i
            TowerDayMeasurements(i+1)=1;
        end
    end

end
figure
plot(StartingDay:caldays(1):EndingDay,MeteoDayMeasurements,'ob');hold on;grid on
plot(StartingDay:caldays(1):EndingDay,TowerDayMeasurements,'xr');
title('Data availability')
legend('MeteoDayMeasurements','TowerDayMeasurements')
xlabel('Time')
ylabel('Measured')
NO_METEODAYS=length(MeteoDailyFile);
NO_TOWERDAYS=length(TowerDailyFile)-Nanflag;
```

Data consistency

```
n=1;
for i=1:NO_METEODAYS
   for j=1:NO_TOWERDAYS
       if MeteoDate(i)==TowerDate(j)
           M_DATA{n}=MeteoDailyFile{i};
           M_END(n)=EndingTimeMeteo(i);
           T_DATA{n}=NewTowerDailyFile{j};
           T_START(n)=StartingTimeTower(j);
           n=n+1;
       end
   end
end
```

9.1 Creating Data for Meteo (extending)

```
for i=1:n-1
    NORMALIZED_VAR=M_DATA{i}.Variables;
    NM=zeros(15*(size(NORMALIZED_VAR,1)-1),size(NORMALIZED_VAR,2));
    UTCTimeMeteoExtended=zeros(size(NORMALIZED_VAR,1),1);
    cnt_minutes=1;
    for k=1:size(NORMALIZED_VAR,1)-1
        for j=1:size(NORMALIZED_VAR,2)
            C(:,j)=linspace(NORMALIZED_VAR(k,j),NORMALIZED_VAR(k+1,j),15);
        end
        NM([cnt_minutes:15*k],:)=C;
        cnt_minutes=cnt_minutes+15;
    end
    UTCTimeMeteoExtended=linspace(0,(cnt_minutes-1)/60/24,cnt_minutes-1)';
    EXTENDED_METEO_DATA{i}=[UTCTimeMeteoExtended NM];
end
% figure
% subplot(1,2,1)
```

```
% plot(M_DATA{1}.Umiditarel,'r');grid on
% title('Humidity')
% subplot(1,2,2)
% plot(EXTENDED_METEO_DATA{1}(:,2),'b');grid on
% title('Stretched Humidity')
% for i=1:length(T_DATA)
%     figure
%     plot(T_DATA{i}.Treatt_0)
%     end
```

9.2 Preparing Data for NET (consistency of Tower and Meteo data per minute)

```
for i=1:length(T_DATA)
    DataMeteoExtended=EXTENDED_METEO_DATA{i}(:,2:end);
    UTCTimeMeteoExtended=UTCTimeMeteoExtended(:,1);
    EndingSamples=min(UTCTimeMeteoExtended(end),T_DATA{i}.UTCTime(end));
    for j=1:height(EXTENDED_METEO_DATA{i})
        if (abs(UTCTimeMeteoExtended(j)-T_START(i))<3e-04)
            if (ismember(EndingSamples,T_DATA{i}.UTCTime))

readyMETEOdata=zeros(size(T_DATA{i},1),size(EXTENDED_METEO_DATA{i},2));
                    readyMETEOdata=DataMeteoExtended(j:height(T_DATA{i})+j-1,:);
                    readyTOWERdata=T_DATA{i}.Treatt_0;
                    LOLTOWERConfiguration=T_DATA{i}.Configuration;
            else
                    readyMETEOdata=zeros(size(EXTENDED_METEO_DATA{i},1)-
j+1,size(EXTENDED_METEO_DATA{i},2));
                    readyMETEOdata=DataMeteoExtended(j:end,:);
                    readyTOWERdata=T_DATA{i}.Treatt_0(1:length(DataMeteoExtended)-j+1);

LOLTOWERConfiguration=T_DATA{i}.Configuration(1:length(DataMeteoExtended)-j+1);
            end
        end
    end
    netDATAMETEO{i}=readyMETEOdata;
    netDATATOWER{i}=readyTOWERdata;
end
```

Data net

```
xt=netDATAMETEO{1}';
yt=netDATATOWER{1}';
 for i=2:length(netDATAMETEO)
    xt=catsamples(xt,netDATAMETEO{i}');
    yt=catsamples(yt,netDATATOWER{i}');
 end
% NORMALIZATION OF ALL INPUT IN [-1,1]
 for j=1:7
    xt(j,:)=2*((xt(j,:)-min(xt(j,:)))./(max(xt(j,:))-min(xt(j,:))))-1;
 end
```

Neurons number of 1st hidden layer

```
 S=1;
 N=20;
```

Errors initialization

```
RMSET=zeros(N,8); MAET=RMSET; MBET=RMSET; MAPET=RMSET; RT=RMSET; R2T=RMSET;
RMSEV=zeros(N,8); MAEV=RMSEV; MBEV=RMSEV; MAPEV=RMSEV; RV=RMSEV; R2V=RMSEV;
RMSETe=zeros(N,8); MAETe=RMSETe; MBETe=RMSETe; MAPETe=RMSETe; RTe=RMSETe; R2Te=RMSETe;
```

Levenberg-Marquardt 1 HL

```
NUMBERS='';
load('teInd.mat');
load('tInd.mat');
load('vInd.mat');
 for i=S:N
hiddenLayer1Size=i;
net=fitnet(hiddenLayer1Size);

net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices

net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainlm(net,xt,yt); %tr=info

% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainLevenberg_Marquardt1L(i)=mean(yTrain-yTrainTrue);
MAEtrainLevenberg_Marquardt1L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainLevenberg_Marquardt1L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainLevenberg_Marquardt1L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainLevenberg_Marquardt1L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSE1L(i)=sqrt(mean((yTrain-mean(yTrain)-yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSE1L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,1)=RMSEtrainLevenberg_Marquardt1L(i);
MAET(i,1)=MAEtrainLevenberg_Marquardt1L(i);
MBET(i,1)=MBEtrainLevenberg_Marquardt1L(i);
MAPET(i,1)=MAPEtrainLevenberg_Marquardt1L(i);
RT(i,1)=Rtrain(i);
R2T(i,1)=RsquaredtrainLevenberg_Marquardt1L(i);
TimeLevenberg_Marquardt1L(i)=tr.time(end);
EpochsLevenberg_Marquardt1L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalLevenberg_Marquardt1L(i)=mean(yVal-yValTrue);
MAEvalLevenberg_Marquardt1L(i)=mean(abs(yVal-yValTrue));
MAPEvalLevenberg_Marquardt1L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalLevenberg_Marquardt1L(i)=Rval(i)^2;
RMSEvalLevenberg_Marquardt1L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSE1L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSE1L(i)=mean(yVal)-mean(yValTrue);
RMSEV(i,1)=RMSEvalLevenberg_Marquardt1L(i);
MAEV(i,1)=MAEvalLevenberg_Marquardt1L(i);
MBEV(i,1)=MBEvalLevenberg_Marquardt1L(i);
MAPEV(i,1)=MAPEvalLevenberg_Marquardt1L(i);
RV(i,1)=Rval(i);
R2V(i,1)=RsquaredvalLevenberg_Marquardt1L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestLevenberg_Marquardt1L(i)=mean(yTest-yTestTrue);
```

```
MAEtestLevenberg_Marquardt1L(i)=mean(abs(yTest-yTestTrue));
MAPEtestLevenberg_Marquardt1L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestLevenberg_Marquardt1L(i)=Rtest(i)^2;
RMSEtestLevenberg_Marquardt1L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSELM1L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSELM1L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,1)=RMSEtestLevenberg_Marquardt1L(i);
MAETe(i,1)=MAEtestLevenberg_Marquardt1L(i);
MBETe(i,1)=MBEtestLevenberg_Marquardt1L(i);
MAPETe(i,1)=MAPEtestLevenberg_Marquardt1L(i);
RTe(i,1)=Rtest(i);
R2Te(i,1)=RsquaredtestLevenberg_Marquardt1L(i);
 end
STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
RMS_A=[0 centeredtrainRMSE1L;0 centeredvalRMSE1L;0 centeredtestRMSELM1L];
NUMBERS= ['0',NUMBERS];
% Taylor Diagram
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Levenber-Marquardt 1 HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Levenber-Marquardt 1
HL'),'fontweight','bold',...
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Levenber-Marquardt 1 HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Levenberg-Marquardt 2 HL

```
 NUMBERS='';
 for i=S:N
hiddenLayer1Size=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
```

```
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
[net,tr]=trainlm(net,xt,yt); %tr=info

% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainLevenberg_Marquardt2L(i)=mean(yTrain-yTrainTrue);
MAEtrainLevenberg_Marquardt2L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainLevenberg_Marquardt2L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainLevenberg_Marquardt2L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainLevenberg_Marquardt2L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSELM2L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSELM2L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,2)=RMSEtrainLevenberg_Marquardt2L(i);
MAET(i,2)=MAEtrainLevenberg_Marquardt2L(i);
MBET(i,2)=MBEtrainLevenberg_Marquardt2L(i);
MAPET(i,2)=MAPEtrainLevenberg_Marquardt2L(i);
RT(i,2)=Rtrain(i);
R2T(i,2)=RsquaredtrainLevenberg_Marquardt2L(i);
TimeLevenberg_Marquardt2L(i)=tr.time(end);
EpochsLevenberg_Marquardt2L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalLevenberg_Marquardt2L(i)=mean(yVal-yValTrue);
MAEvalLevenberg_Marquardt2L(i)=mean(abs(yVal-yValTrue));
MAPEvalLevenberg_Marquardt2L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalLevenberg_Marquardt2L(i)=Rval(i)^2;
RMSEvalLevenberg_Marquardt2L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSELM2L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSELM2L(i)=mean(yVal)-mean(yValTrue);

RMSEV(i,2)=RMSEvalLevenberg_Marquardt2L(i);
MAEV(i,2)=MAEvalLevenberg_Marquardt2L(i);
MBEV(i,2)=MBEvalLevenberg_Marquardt2L(i);
MAPEV(i,2)=MAPEvalLevenberg_Marquardt2L(i);
RV(i,2)=Rval(i);
R2V(i,2)=RsquaredvalLevenberg_Marquardt2L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestLevenberg_Marquardt2L(i)=mean(yTest-yTestTrue);
MAEtestLevenberg_Marquardt2L(i)=mean(abs(yTest-yTestTrue));
MAPEtestLevenberg_Marquardt2L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
% RMSEtrainLevenberg_Marquardt1L(i)=sqrt(stdTrue(i).^2+stdOut(i).^2-
2*stdTrue(i)*stdOut(i).*R(i))
RsquaredtestLevenberg_Marquardt2L(i)=Rtest(i)^2;
RMSEtestLevenberg_Marquardt2L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSELM2L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSELM2L(i)=mean(yTest)-mean(yTestTrue);

RMSETe(i,2)=RMSEtestLevenberg_Marquardt2L(i);
MAETe(i,2)=MAEtestLevenberg_Marquardt2L(i);
MBETe(i,2)=MBEtestLevenberg_Marquardt2L(i);
MAPETe(i,2)=MAPEtestLevenberg_Marquardt2L(i);
RTe(i,2)=Rtest(i);
R2Te(i,2)=RsquaredtestLevenberg_Marquardt2L(i);
 end
```

```
STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
RMS_A=[0 centeredtrainRMSELM2L;0 centeredvalRMSELM2L;0 centeredtestRMSELM2L];
NUMBERS= ['0',NUMBERS];
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Levenber-Marquardt 2 HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Levenber-Marquardt 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Levenber-Marquardt 2 HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Bayesian regularization 1 HL

```
NUMBERS='';
for i=S:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.testInd= sort([teInd vInd]);  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainbr(net,xt,yt); %tr=info
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainBayesian_regularization1L(i)=mean(yTrain-yTrainTrue);
MAEtrainBayesian_regularization1L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainBayesian_regularization1L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainBayesian_regularization1L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainBayesian_regularization1L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSEBR1L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
```

```
OverallBiastrainRMSEBR1L(i)=mean(yTrain)-mean(yTrainTrue);

RMSET(i,3)=RMSEtrainBayesian_regularization1L(i);
MAET(i,3)=MAEtrainBayesian_regularization1L(i);
MBET(i,3)=MBEtrainBayesian_regularization1L(i);
MAPET(i,3)=MAPEtrainBayesian_regularization1L(i);
RT(i,3)=Rtrain(i);
R2T(i,3)=RsquaredtrainBayesian_regularization1L(i);
TimeBayesian_regularization1L(i)=tr.time(end);
EpochsBayesian_regularization1L(i)=tr.epoch(end);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestBayesian_regularization1L(i)=mean(yTest-yTestTrue);
MAEtestBayesian_regularization1L(i)=mean(abs(yTest-yTestTrue));
MAPEtestBayesian_regularization1L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestBayesian_regularization1L(i)=Rtest(i)^2;
RMSEtestBayesian_regularization1L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSEBR1L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSEBR1L(i)=mean(yTest)-mean(yTestTrue);

RMSETe(i,3)=RMSEtestBayesian_regularization1L(i);
MAETe(i,3)=MAEtestBayesian_regularization1L(i);
MBETe(i,3)=MBEtestBayesian_regularization1L(i);
MAPETe(i,3)=MAPEtestBayesian_regularization1L(i);
RTe(i,3)=Rtest(i);
R2Te(i,3)=RsquaredtestBayesian_regularization1L(i);
 end

STD_A=[stdtrainTrue(1) stdtrainOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rtest];
RMS_A=[0 centeredtrainRMSEBR1L;0 centeredtestRMSEBR1L];
NUMBERS= ['0',NUMBERS];
for j=1:2
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Bayesian regularization 1
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Test BPANN Bayesian regularization 1 HL'),'fontweight','bold',...
'fontsize',12);
end
end
```

Bayesian regularization with 2 Hidden

```
NUMBERS='';
for i=S:N
hiddenLayer1Size=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.testInd= sort([teInd vInd]);  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
net.layers{3}.transferFcn='purelin';
[net,tr]=trainbr(net,xt,yt); %tr=info

% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainBayesian_regularization2L(i)=mean(yTrain-yTrainTrue);
MAEtrainBayesian_regularization2L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainBayesian_regularization2L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainBayesian_regularization2L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainBayesian_regularization2L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSEBR2L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSEBR2L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,4)=RMSEtrainBayesian_regularization2L(i);
MAET(i,4)=MAEtrainBayesian_regularization2L(i);
MBET(i,4)=MBEtrainBayesian_regularization2L(i);
MAPET(i,4)=MAPEtrainBayesian_regularization2L(i);
RT(i,4)=Rtrain(i);
R2T(i,4)=RsquaredtrainBayesian_regularization2L(i);
TimeBayesian_regularization2L(i)=tr.time(end);
EpochsBayesian_regularization2L(i)=tr.epoch(end);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestBayesian_regularization2L(i)=mean(yTest-yTestTrue);
MAEtestBayesian_regularization2L(i)=mean(abs(yTest-yTestTrue));
MAPEtestBayesian_regularization2L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestBayesian_regularization2L(i)=Rtest(i)^2;
RMSEtestBayesian_regularization2L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSEBR2L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSEBR2L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,4)=RMSEtestBayesian_regularization2L(i);
MAETe(i,4)=MAEtestBayesian_regularization2L(i);
MBETe(i,4)=MBEtestBayesian_regularization2L(i);
MAPETe(i,4)=MAPEtestBayesian_regularization2L(i);
RTe(i,4)=Rtest(i);
R2Te(i,4)=RsquaredtestBayesian_regularization2L(i);
 end
STD_A=[stdtrainTrue(1) stdtrainOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rtest];
RMS_A=[0 centeredtrainRMSEBR2L;0 centeredtestRMSEBR2L];
NUMBERS= ['0',NUMBERS];
for j=1:2
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
```

```
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Bayesian regularization 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Test BPANN Bayesian regularization 2 HL'),'fontweight','bold',...
'fontsize',12);
end
end
```

Resilient backpropagation 1 HL

```
 NUMBERS='';
 for i=S:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='logsig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainrp(net,xt,yt); %tr=info
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainResilient_Backpropagation1L(i)=mean(yTrain-yTrainTrue);
MAEtrainResilient_Backpropagation1L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainResilient_Backpropagation1L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainResilient_Backpropagation1L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainResilient_Backpropagation1L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSERB1L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSERB1L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,5)=RMSEtrainResilient_Backpropagation1L(i);
MAET(i,5)=MAEtrainResilient_Backpropagation1L(i);
MBET(i,5)=MBEtrainResilient_Backpropagation1L(i);
MAPET(i,5)=MAPEtrainResilient_Backpropagation1L(i);
RT(i,5)=Rtrain(i);
R2T(i,5)=RsquaredtrainResilient_Backpropagation1L(i);
TimeResilient_Backpropagation1L(i)=tr.time(end);
EpochsResilient_Backpropagation1L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalResilient_Backpropagation1L(i)=mean(yVal-yValTrue);
```

```
MAEvalResilient_Backpropagation1L(i)=mean(abs(yVal-yValTrue));
MAPEvalResilient_Backpropagation1L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalResilient_Backpropagation1L(i)=Rval(i)^2;
RMSEvalResilient_Backpropagation1L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSERB1L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSERB1L(i)=mean(yVal)-mean(yValTrue);
RMSEV(i,5)=RMSEvalResilient_Backpropagation1L(i);
MAEV(i,5)=MAEvalResilient_Backpropagation1L(i);
MBEV(i,5)=MBEvalResilient_Backpropagation1L(i);
MAPEV(i,5)=MAPEvalResilient_Backpropagation1L(i);
RV(i,5)=Rval(i);
R2V(i,5)=RsquaredvalResilient_Backpropagation1L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestResilient_Backpropagation1L(i)=mean(yTest-yTestTrue);
MAEtestResilient_Backpropagation1L(i)=mean(abs(yTest-yTestTrue));
MAPEtestResilient_Backpropagation1L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestResilient_Backpropagation1L(i)=Rtest(i)^2;
RMSEtestResilient_Backpropagation1L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSERB1L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSERB1L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,5)=RMSEtestResilient_Backpropagation1L(i);
MAETe(i,5)=MAEtestResilient_Backpropagation1L(i);
MBETe(i,5)=MBEtestResilient_Backpropagation1L(i);
MAPETe(i,5)=MAPEtestResilient_Backpropagation1L(i);
RTe(i,5)=Rtest(i);
R2Te(i,5)=RsquaredtestResilient_Backpropagation1L(i);
 end
STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
RMS_A=[0 centeredtrainRMSERB1L;0 centeredvalRMSERB1L;0 centeredtestRMSERB1L];
NUMBERS= ['0',NUMBERS];
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Resilient Backpropagation 1
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Resilient Backpropagation 1
HL'),'fontweight','bold',...
```

```
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Resilient Backpropagation 1
HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Resilient backpropagation 2 HL

```
NUMBERS='';
 for i=S:N
hiddenLayerSize=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='logsig';
net.layers{2}.transferFcn='logsig';
net.layers{3}.transferFcn='purelin';

[net,tr]=trainrp(net,xt,yt); %tr=info
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainResilient_Backpropagation2L(i)=mean(yTrain-yTrainTrue);
MAEtrainResilient_Backpropagation2L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainResilient_Backpropagation2L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainResilient_Backpropagation2L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainResilient_Backpropagation2L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSERB2L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSERB2L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,6)=RMSEtrainResilient_Backpropagation2L(i);
MAET(i,6)=MAEtrainResilient_Backpropagation2L(i);
MBET(i,6)=MBEtrainResilient_Backpropagation2L(i);
MAPET(i,6)=MAPEtrainResilient_Backpropagation2L(i);
RT(i,6)=Rtrain(i);
R2T(i,6)=RsquaredtrainResilient_Backpropagation2L(i);
TimeResilient_Backpropagation2L(i)=tr.time(end);
EpochsResilient_Backpropagation2L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalResilient_Backpropagation2L(i)=mean(yVal-yValTrue);
MAEvalResilient_Backpropagation2L(i)=mean(abs(yVal-yValTrue));
MAPEvalResilient_Backpropagation2L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalResilient_Backpropagation2L(i)=Rval(i)^2;
RMSEvalResilient_Backpropagation2L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSERB2L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSERB2L(i)=mean(yVal)-mean(yValTrue);
RMSEV(i,6)=RMSEvalResilient_Backpropagation2L(i);
MAEV(i,6)=MAEvalResilient_Backpropagation2L(i);
MBEV(i,6)=MBEvalResilient_Backpropagation2L(i);
MAPEV(i,6)=MAPEvalResilient_Backpropagation2L(i);
RV(i,6)=Rval(i);
R2V(i,6)=RsquaredvalResilient_Backpropagation2L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
```

```
MBEtestResilient_Backpropagation2L(i)=mean(yTest-yTestTrue);
MAEtestResilient_Backpropagation2L(i)=mean(abs(yTest-yTestTrue));
MAPEtestResilient_Backpropagation2L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestResilient_Backpropagation2L(i)=Rtest(i)^2;
RMSEtestResilient_Backpropagation2L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSERB2L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSERB2L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,6)=RMSEtestResilient_Backpropagation2L(i);
MAETe(i,6)=MAEtestResilient_Backpropagation2L(i);
MBETe(i,6)=MBEtestResilient_Backpropagation2L(i);
MAPETe(i,6)=MAPEtestResilient_Backpropagation2L(i);
RTe(i,6)=Rtest(i);
R2Te(i,6)=RsquaredtestResilient_Backpropagation2L(i);
 end
STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
RMS_A=[0 centeredtrainRMSERB2L;0 centeredvalRMSERB2L;0 centeredtestRMSERB2L];
NUMBERS= ['0',NUMBERS];
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Resilient Backpropagation 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Resilient Backpropagation 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Resilient Backpropagation 2
HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Scaled conjugate gradient 1 HL

```
NUMBERS='';
  for i=S:N
hiddenLayerSize=i;
net=fitnet(hiddenLayerSize);
net.divideFcn= 'divideind'; % divide the data manually
```

```
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
[net,tr]=trainscg(net,xt,yt); %tr=info
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainScaled_Conjugate_Gradient1L(i)=mean(yTrain-yTrainTrue);
MAEtrainScaled_Conjugate_Gradient1L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainScaled_Conjugate_Gradient1L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainScaled_Conjugate_Gradient1L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
RMSEtrainScaled_Conjugate_Gradient1L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSESCG1L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSESCG1L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,7)=RMSEtrainScaled_Conjugate_Gradient1L(i);
MAET(i,7)=MAEtrainScaled_Conjugate_Gradient1L(i);
MBET(i,7)=MBEtrainScaled_Conjugate_Gradient1L(i);
MAPET(i,7)=MAPEtrainScaled_Conjugate_Gradient1L(i);
RT(i,7)=Rtrain(i);
R2T(i,7)=RsquaredtrainScaled_Conjugate_Gradient1L(i);
TimeScaled_Conjugate_Gradient1L(i)=tr.time(end);
EpochsScaled_Conjugate_Gradient1L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalScaled_Conjugate_Gradient1L(i)=mean(yVal-yValTrue);
MAEvalScaled_Conjugate_Gradient1L(i)=mean(abs(yVal-yValTrue));
MAPEvalScaled_Conjugate_Gradient1L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalScaled_Conjugate_Gradient1L(i)=Rval(i)^2;
RMSEvalScaled_Conjugate_Gradient1L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSESCG1L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSESCG1L(i)=mean(yVal)-mean(yValTrue);
RMSEV(i,7)=RMSEvalScaled_Conjugate_Gradient1L(i);
MAEV(i,7)=MAEvalScaled_Conjugate_Gradient1L(i);
MBEV(i,7)=MBEvalScaled_Conjugate_Gradient1L(i);
MAPEV(i,7)=MAPEvalScaled_Conjugate_Gradient1L(i);
RV(i,7)=Rval(i);
R2V(i,7)=RsquaredvalScaled_Conjugate_Gradient1L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestScaled_Conjugate_Gradient1L(i)=mean(yTest-yTestTrue);
MAEtestScaled_Conjugate_Gradient1L(i)=mean(abs(yTest-yTestTrue));
MAPEtestScaled_Conjugate_Gradient1L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestScaled_Conjugate_Gradient1L(i)=Rtest(i)^2;
RMSEtestScaled_Conjugate_Gradient1L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSESCG1L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSESCG1L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,7)=RMSEtestScaled_Conjugate_Gradient1L(i);
MAETe(i,7)=MAEtestScaled_Conjugate_Gradient1L(i);
MBETe(i,7)=MBEtestScaled_Conjugate_Gradient1L(i);
MAPETe(i,7)=MAPEtestScaled_Conjugate_Gradient1L(i);
RTe(i,7)=Rtest(i);
R2Te(i,7)=RsquaredtestScaled_Conjugate_Gradient1L(i);
 end

STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
```

```
RMS_A=[0 centeredtrainRMSESCG1L;0 centeredvalRMSESCG1L;0 centeredtestRMSESCG1L];
NUMBERS= ['0',NUMBERS];
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Scaled Conjugate Gradient 1
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Scaled Conjugate Gradient 1
HL'),'fontweight','bold',...
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Scaled Conjugate Gradient 1
HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Scaled conjugate gradient 2 HL

```
NUMBERS='';
  for i=S:N
hiddenLayerSize=i;
hiddenLayer2Size=3*i;
net=fitnet([hiddenLayer1Size hiddenLayer2Size]);
net.divideFcn= 'divideind'; % divide the data manually
net.divideParam.trainInd= tInd; % training data indices
net.divideParam.valInd= vInd; % validation data indices
net.divideParam.testInd= teInd;  % testing data indices
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
net.layers{3}.transferFcn='purelin';
[net,tr]=trainscg(net,xt,yt); %tr=info
% Train
yTrain=net(xt(:,tr.trainInd));  %predicted output of NN
yTrainTrue= yt(tr.trainInd); %real output
MBEtrainScaled_Conjugate_Gradient2L(i)=mean(yTrain-yTrainTrue);
MAEtrainScaled_Conjugate_Gradient2L(i)=mean(abs(yTrain-yTrainTrue));
MAPEtrainScaled_Conjugate_Gradient2L(i)=mean(abs((yTrain-yTrainTrue)/yTrainTrue))*100;
stdtrainTrue(i)=sqrt(mean((yTrainTrue-mean(yTrainTrue)).^2));
stdtrainOut(i)=sqrt(mean((yTrain-mean(yTrain)).^2));
Rtrain(i)=mean((yTrain-mean(yTrain)).*(yTrainTrue-
mean(yTrainTrue)))/stdtrainTrue(i)/stdtrainOut(i);
RsquaredtrainScaled_Conjugate_Gradient2L(i)=Rtrain(i)^2;
NUMBERS=split([NUMBERS, num2str(i)]);
```

```
RMSEtrainScaled_Conjugate_Gradient2L(i)=sqrt(mean((yTrain-yTrainTrue).^2));
centeredtrainRMSESCG2L(i)=sqrt(mean((yTrain-mean(yTrain)-
yTrainTrue+mean(yTrainTrue)).^2));
OverallBiastrainRMSESCG2L(i)=mean(yTrain)-mean(yTrainTrue);
RMSET(i,8)=RMSEtrainScaled_Conjugate_Gradient2L(i);
MAET(i,8)=MAEtrainScaled_Conjugate_Gradient2L(i);
MBET(i,8)=MBEtrainScaled_Conjugate_Gradient2L(i);
MAPET(i,8)=MAPEtrainScaled_Conjugate_Gradient2L(i);
RT(i,8)=Rtrain(i);
R2T(i,8)=RsquaredtrainScaled_Conjugate_Gradient2L(i);
TimeScaled_Conjugate_Gradient2L(i)=tr.time(end);
EpochsScaled_Conjugate_Gradient2L(i)=tr.epoch(end);
% Val
yVal=net(xt(:,tr.valInd));  %predicted output of NN
yValTrue= yt(tr.valInd); %real output
MBEvalScaled_Conjugate_Gradient2L(i)=mean(yVal-yValTrue);
MAEvalScaled_Conjugate_Gradient2L(i)=mean(abs(yVal-yValTrue));
MAPEvalScaled_Conjugate_Gradient2L(i)=mean(abs((yVal-yValTrue)/yValTrue))*100;
stdvalTrue(i)=sqrt(mean((yValTrue-mean(yValTrue)).^2));
stdvalOut(i)=sqrt(mean((yVal-mean(yVal)).^2));
Rval(i)=mean((yVal-mean(yVal)).*(yValTrue-mean(yValTrue)))/stdvalTrue(i)/stdvalOut(i);
RsquaredvalScaled_Conjugate_Gradient2L(i)=Rval(i)^2;
RMSEvalScaled_Conjugate_Gradient2L(i)=sqrt(mean((yVal-yValTrue).^2));
centeredvalRMSESCG2L(i)=sqrt(mean((yVal-mean(yVal)-yValTrue+mean(yValTrue)).^2));
OverallBiasvalRMSESCG2L(i)=mean(yVal)-mean(yValTrue);
RMSEV(i,8)=RMSEvalScaled_Conjugate_Gradient2L(i);
MAEV(i,8)=MAEvalScaled_Conjugate_Gradient2L(i);
MBEV(i,8)=MBEvalScaled_Conjugate_Gradient2L(i);
MAPEV(i,8)=MAPEvalScaled_Conjugate_Gradient2L(i);
RV(i,8)=Rval(i);
R2V(i,8)=RsquaredvalScaled_Conjugate_Gradient2L(i);
% Test
yTest=net(xt(:,tr.testInd));  %predicted output of NN
yTestTrue= yt(tr.testInd); %real output
MBEtestScaled_Conjugate_Gradient2L(i)=mean(yTest-yTestTrue);
MAEtestScaled_Conjugate_Gradient2L(i)=mean(abs(yTest-yTestTrue));
MAPEtestScaled_Conjugate_Gradient2L(i)=mean(abs((yTest-yTestTrue)/yTestTrue))*100;
stdtestTrue(i)=sqrt(mean((yTestTrue-mean(yTestTrue)).^2));
stdtestOut(i)=sqrt(mean((yTest-mean(yTest)).^2));
Rtest(i)=mean((yTest-mean(yTest)).*(yTestTrue-
mean(yTestTrue)))/stdtestTrue(i)/stdtestOut(i);
RsquaredtestScaled_Conjugate_Gradient2L(i)=Rtest(i)^2;
RMSEtestScaled_Conjugate_Gradient2L(i)=sqrt(mean((yTest-yTestTrue).^2));
centeredtestRMSESCG2L(i)=sqrt(mean((yTest-mean(yTest)-yTestTrue+mean(yTestTrue)).^2));
OverallBiastestRMSESCG2L(i)=mean(yTest)-mean(yTestTrue);
RMSETe(i,8)=RMSEtestScaled_Conjugate_Gradient2L(i);
MAETe(i,8)=MAEtestScaled_Conjugate_Gradient2L(i);
MBETe(i,8)=MBEtestScaled_Conjugate_Gradient2L(i);
MAPETe(i,8)=MAPEtestScaled_Conjugate_Gradient2L(i);
RTe(i,8)=Rtest(i);
R2Te(i,8)=RsquaredtestScaled_Conjugate_Gradient2L(i);
 end

STD_A=[stdtrainTrue(1) stdtrainOut;stdvalTrue(1) stdvalOut;stdtestTrue(1) stdtestOut];
CORR_A=[1 Rtrain;1 Rval;1 Rtest];
RMS_A=[0 centeredtrainRMSESCG2L;0 centeredvalRMSESCG2L;0 centeredtestRMSESCG2L];
NUMBERS= ['0',NUMBERS];
for j=1:3
figure
    if j==1
        [hp ht axl,color1,color2,color3] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,0);
        color=[color1,color2,color3];
    else
        [hp ht axl] =
taylordiag(squeeze(STD_A(j,:)),squeeze(RMS_A(j,:)),squeeze(CORR_A(j,:)),NUMBERS,j,color)
;
end
for ii = 1 : length(ht)
    if ii == 1
        set(ht(ii),'String','Obs');
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
```

```
        legendinfo{ii}=['Observed'];
    else
        set(ht(ii),'fontsize',9,'fontweight','bold','color',color(ii,:))
        set(hp(ii),'markersize',12,'MarkerEdgeColor',color(ii,:))
        legendinfo{ii}=['7-' num2str(ii-1) '-1-1'];
    end
end
xlabel('Standard deviation','fontweight','bold')
legend(hp(1:ii),legendinfo,'Location',[0.8 0.7 0.1778 0.1957])
switch j
    case 1
        title(sprintf('Training BPANN Scaled Conjugate Gradient 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 2
        title(sprintf('Validation BPANN Scaled Conjugate Gradient 2
HL'),'fontweight','bold',...
'fontsize',12);
    case 3
        title(sprintf('Test BPANN Scaled Conjugate Gradient 2
HL'),'fontweight','bold',...
'fontsize',12);
    end
end
```

Figures plot

```
figure
title('RMSE TRAIN');hold on; grid on
plot(1:i, RMSEtrainLevenberg_Marquardt1L,'b');
plot(1:i, RMSEtrainLevenberg_Marquardt2L,'--b')
plot(1:i, RMSEtrainBayesian_regularization1L,'g');
plot(1:i, RMSEtrainBayesian_regularization2L,'--g');
plot(1:i, RMSEtrainResilient_Backpropagation1L,'c')
plot(1:i, RMSEtrainResilient_Backpropagation2L,'--c')
plot(1:i, RMSEtrainScaled_Conjugate_Gradient1L,'r')
plot(1:i, RMSEtrainScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('MAE TRAIN');hold on ; grid on
plot(1:i, MAEtrainLevenberg_Marquardt1L,'b');
plot(1:i, MAEtrainLevenberg_Marquardt2L,'--b')
plot(1:i, MAEtrainBayesian_regularization1L,'g');
plot(1:i, MAEtrainBayesian_regularization2L,'--g');
plot(1:i, MAEtrainResilient_Backpropagation1L,'c')
plot(1:i, MAEtrainResilient_Backpropagation2L,'--c')
plot(1:i, MAEtrainScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAEtrainScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('MBE TRAIN');hold on; grid on
plot(1:i, MBEtrainLevenberg_Marquardt1L,'b');
plot(1:i, MBEtrainLevenberg_Marquardt2L,'--b')
plot(1:i, MBEtrainBayesian_regularization1L,'g');
plot(1:i, MBEtrainBayesian_regularization2L,'--g');
plot(1:i, MBEtrainResilient_Backpropagation1L,'c')
plot(1:i, MBEtrainResilient_Backpropagation2L,'--c')
plot(1:i, MBEtrainScaled_Conjugate_Gradient1L,'r')
plot(1:i, MBEtrainScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
```

```
title('MAPE TRAIN'); hold on; grid on
plot(1:i, MAPEtrainLevenberg_Marquardt1L,'b');
plot(1:i, MAPEtrainLevenberg_Marquardt2L,'--b')
plot(1:i, MAPEtrainBayesian_regularization1L,'g');
plot(1:i, MAPEtrainBayesian_regularization2L,'--g');
plot(1:i, MAPEtrainResilient_Backpropagation1L,'c')
plot(1:i, MAPEtrainResilient_Backpropagation2L,'--c')
plot(1:i, MAPEtrainScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAPEtrainScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('R TRAIN'); hold on; grid on
plot(1:i, sqrt(RsquaredtrainLevenberg_Marquardt1L),'b');
plot(1:i, sqrt(RsquaredtrainLevenberg_Marquardt2L),'--b')
plot(1:i, sqrt(RsquaredtrainBayesian_regularization1L),'g');
plot(1:i, sqrt(RsquaredtrainBayesian_regularization2L),'--g');
plot(1:i, sqrt(RsquaredtrainResilient_Backpropagation1L),'c')
plot(1:i, sqrt(RsquaredtrainResilient_Backpropagation2L),'--c')
plot(1:i, sqrt(RsquaredtrainScaled_Conjugate_Gradient1L),'r')
plot(1:i, sqrt(RsquaredtrainScaled_Conjugate_Gradient2L),'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('Rsquared TRAIN'); hold on; grid on
plot(1:i, RsquaredtrainLevenberg_Marquardt1L,'b');
plot(1:i, RsquaredtrainLevenberg_Marquardt2L,'--b')
plot(1:i, RsquaredtrainBayesian_regularization1L,'g');
plot(1:i, RsquaredtrainBayesian_regularization2L,'--g');
plot(1:i, RsquaredtrainResilient_Backpropagation1L,'c')
plot(1:i, RsquaredtrainResilient_Backpropagation2L,'--c')
plot(1:i, RsquaredtrainScaled_Conjugate_Gradient1L,'r')
plot(1:i, RsquaredtrainScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('RMSE VAL');hold on; grid on
plot(1:i, RMSEvalLevenberg_Marquardt1L,'b');
plot(1:i, RMSEvalLevenberg_Marquardt2L,'--b')
plot(1:i, RMSEvalResilient_Backpropagation1L,'c')
plot(1:i, RMSEvalResilient_Backpropagation2L,'--c')
plot(1:i, RMSEvalScaled_Conjugate_Gradient1L,'r')
plot(1:i, RMSEvalScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')

figure
title('MAE VAL');hold on ; grid on
plot(1:i, MAEvalLevenberg_Marquardt1L,'b');
plot(1:i, MAEvalLevenberg_Marquardt2L,'--b')
plot(1:i, MAEvalResilient_Backpropagation1L,'c')
plot(1:i, MAEvalResilient_Backpropagation2L,'--c')
plot(1:i, MAEvalScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAEvalScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')

figure
title('MBE VAL');hold on; grid on
plot(1:i, MBEvalLevenberg_Marquardt1L,'b');
plot(1:i, MBEvalLevenberg_Marquardt2L,'--b')
plot(1:i, MBEvalResilient_Backpropagation1L,'c')
plot(1:i, MBEvalResilient_Backpropagation2L,'--c')
plot(1:i, MBEvalScaled_Conjugate_Gradient1L,'r')
plot(1:i, MBEvalScaled_Conjugate_Gradient2L,'--r')
```

```
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')

figure
title('MAPE VAL'); hold on; grid on
plot(1:i, MAPEvalLevenberg_Marquardt1L,'b');
plot(1:i, MAPEvalLevenberg_Marquardt2L,'--b')
plot(1:i, MAPEvalResilient_Backpropagation1L,'c')
plot(1:i, MAPEvalResilient_Backpropagation2L,'--c')
plot(1:i, MAPEvalScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAPEvalScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')

figure
title('R VAL'); hold on; grid on
plot(1:i, sqrt(RsquaredvalLevenberg_Marquardt1L),'b');
plot(1:i, sqrt(RsquaredvalLevenberg_Marquardt2L),'--b')
plot(1:i, sqrt(RsquaredvalResilient_Backpropagation1L),'c')
plot(1:i, sqrt(RsquaredvalResilient_Backpropagation2L),'--c')
plot(1:i, sqrt(RsquaredvalScaled_Conjugate_Gradient1L),'r')
plot(1:i, sqrt(RsquaredvalScaled_Conjugate_Gradient2L),'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')
figure
title('Rsquared VAL'); hold on; grid on
plot(1:i, RsquaredvalLevenberg_Marquardt1L,'b');
plot(1:i, RsquaredvalLevenberg_Marquardt2L,'--b')
plot(1:i, RsquaredvalResilient_Backpropagation1L,'c')
plot(1:i, RsquaredvalResilient_Backpropagation2L,'--c')
plot(1:i, RsquaredvalScaled_Conjugate_Gradient1L,'r')
plot(1:i, RsquaredvalScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Resilient
Backpropagation','Resilient Backpropagation2L','Scaled Conjugate Gradient','Scaled
Conjugate Gradient2L')

figure
title('RMSE TEST');hold on; grid on
plot(1:i, RMSEtestLevenberg_Marquardt1L,'b');
plot(1:i, RMSEtestLevenberg_Marquardt2L,'--b')
plot(1:i, RMSEtestBayesian_regularization1L,'g');
plot(1:i, RMSEtestBayesian_regularization2L,'--g');
plot(1:i, RMSEtestResilient_Backpropagation1L,'c')
plot(1:i, RMSEtestResilient_Backpropagation2L,'--c')
plot(1:i, RMSEtestScaled_Conjugate_Gradient1L,'r')
plot(1:i, RMSEtestScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('MAE TEST');hold on ; grid on
plot(1:i, MAEtestLevenberg_Marquardt1L,'b');
plot(1:i, MAEtestLevenberg_Marquardt2L,'--b')
plot(1:i, MAEtestBayesian_regularization1L,'g');
plot(1:i, MAEtestBayesian_regularization2L,'--g');
plot(1:i, MAEtestResilient_Backpropagation1L,'c')
plot(1:i, MAEtestResilient_Backpropagation2L,'--c')
plot(1:i, MAEtestScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAEtestScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('MBE TEST');hold on; grid on
plot(1:i, MBEtestLevenberg_Marquardt1L,'b');
plot(1:i, MBEtestLevenberg_Marquardt2L,'--b')
plot(1:i, MBEtestBayesian_regularization1L,'g');
plot(1:i, MBEtestBayesian_regularization2L,'--g');
```

```
plot(1:i, MBEtestResilient_Backpropagation1L,'c')
plot(1:i, MBEtestResilient_Backpropagation2L,'--c')
plot(1:i, MBEtestScaled_Conjugate_Gradient1L,'r')
plot(1:i, MBEtestScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('MAPE TEST'); hold on; grid on
plot(1:i, MAPEtestLevenberg_Marquardt1L,'b');
plot(1:i, MAPEtestLevenberg_Marquardt2L,'--b')
plot(1:i, MAPEtestBayesian_regularization1L,'g');
plot(1:i, MAPEtestBayesian_regularization2L,'--g');
plot(1:i, MAPEtestResilient_Backpropagation1L,'c')
plot(1:i, MAPEtestResilient_Backpropagation2L,'--c')
plot(1:i, MAPEtestScaled_Conjugate_Gradient1L,'r')
plot(1:i, MAPEtestScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('R TEST'); hold on; grid on
plot(1:i, sqrt(RsquaredtestLevenberg_Marquardt1L),'b');
plot(1:i, sqrt(RsquaredtestLevenberg_Marquardt2L),'--b')
plot(1:i, sqrt(RsquaredtestBayesian_regularization1L),'g');
plot(1:i, sqrt(RsquaredtestBayesian_regularization2L),'--g');
plot(1:i, sqrt(RsquaredtestResilient_Backpropagation1L),'c')
plot(1:i, sqrt(RsquaredtestResilient_Backpropagation2L),'--c')
plot(1:i, sqrt(RsquaredtestScaled_Conjugate_Gradient1L),'r')
plot(1:i, sqrt(RsquaredtestScaled_Conjugate_Gradient2L),'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')

figure
title('Rsquared TEST'); hold on; grid on
plot(1:i, RsquaredtestLevenberg_Marquardt1L,'b');
plot(1:i, RsquaredtestLevenberg_Marquardt2L,'--b')
plot(1:i, RsquaredtestBayesian_regularization1L,'g');
plot(1:i, RsquaredtestBayesian_regularization2L,'--g');
plot(1:i, RsquaredtestResilient_Backpropagation1L,'c')
plot(1:i, RsquaredtestResilient_Backpropagation2L,'--c')
plot(1:i, RsquaredtestScaled_Conjugate_Gradient1L,'r')
plot(1:i, RsquaredtestScaled_Conjugate_Gradient2L,'--r')
legend('Levenberg-Marquardt1L','Levenberg-Marquardt2L','Bayesian
regularization','Bayesian regularization2L','Resilient Backpropagation','Resilient
Backpropagation2L','Scaled Conjugate Gradient','Scaled Conjugate Gradient2L')
```

Exports

```
for i=1:8
    switch i
        case 1
            ErrorsLM1L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsLM1L,'ErrorsLM1L.xlsx');
        case 2
            ErrorsLM2L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsLM2L,'ErrorsLM2L.xlsx');
        case 3
            ErrorsBR1L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsBR1L,'ErrorsBR1L.xlsx');
        case 4
```

```
            ErrorsBR2L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsBR2L,'ErrorsBR2L.xlsx');
        case 5
            ErrorsRB1L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsRB1L,'ErrorsRB1L.xlsx');
        case 6
            ErrorsRB2L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsRB2L,'ErrorsRB2L.xlsx');
        case 7
            ErrorsSCG1L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsSCG1L,'ErrorsSCG1L.xlsx');
        case 8
            ErrorsSCG2L=[[1:N]', RMSET(:,i), MAET(:,i), MBET(:,i), MAPET(:,i), RT(:,i),
R2T(:,i), RMSEV(:,i), MAEV(:,i), MBEV(:,i), MAPEV(:,i), RV(:,i), R2V(:,i), RMSETe(:,i),
MAETe(:,i), MBETe(:,i), MAPETe(:,i), RTe(:,i), R2Te(:,i)];
            writematrix(ErrorsSCG2L,'ErrorsSCG2L.xlsx');
    end
end

figure
title('Time required in seconds'); hold on; grid on
bar(1:i,[ TimeLevenberg_Marquardt2L;  TimeBayesian_regularization2L;
TimeResilient_Backpropagation2L;  TimeScaled_Conjugate_Gradient2L])
legend('Levenberg-Marquardt2L','Bayesian regularization2L','Resilient
Backpropagation2L','Scaled Conjugate Gradient2L')

figure
title('Epochs required'); hold on; grid on
bar(1:i,[EpochsLevenberg_Marquardt2L; EpochsBayesian_regularization2L;
EpochsResilient_Backpropagation2L; EpochsScaled_Conjugate_Gradient2L])
legend('Levenberg-Marquardt2L','Bayesian regularization2L','Resilient
Backpropagation2L','Scaled Conjugate Gradient2L')

timematrix=[[1:N]',TimeLevenberg_Marquardt2L', TimeBayesian_regularization2L'
TimeResilient_Backpropagation2L'  TimeScaled_Conjugate_Gradient2L']
writematrix(timematrix,'Time2HL.xlsx')
epochsmatrix=[[1:N]',EpochsLevenberg_Marquardt2L', EpochsBayesian_regularization2L'
EpochsResilient_Backpropagation2L' EpochsScaled_Conjugate_Gradient2L']
writematrix(epochsmatrix,'Epochs2HL.xlsx')
```

Net parameters

```
load('iw.mat');
writematrix(cell2mat(Iw(1,1)),'iw.xlsx')
load('Lw.mat');
writematrix(lw{2,1},'LwH1toH2.xlsx')
writematrix(lw{3,2},'LwH2toO')
load('bias.mat');
writecell(b(3),'b3.xlsx')
writecell(b(2),'b2.xlsx')
writecell(b(1),'b1.xlsx')
```

T test with net parameters

```
Y=cell2mat(Iw(1))*xt(:,teInd(end))
yn=net.lw{3,2}*(tansig(net.lw{2,1}*(tansig(cell2mat(net.Iw(1))*xt(:,sort([teInd
vInd]))+net.b{1}))+net.b{2}))+net.b{3}
T=(max(yt)-min(yt))*(yn+1)/2+min(yt)
```

```
figure
plot(T,'--r');hold on; grid on; plot(yTest,'b')
```

Published with MATLAB® R2020b