FACULTAT
DE CIÈNCIES, TECNOLOGIA
I ENGINYERIES
UVIC | UVIC·UCC

Treball de Fi de Grau Experimental

PREDICCIÓ DE COMPORTAMENT D'AEROGENERADORS MITJANÇANT INFORMACIÓ CREUADA

XAVIER MICÓ PÉREZ

Grau en Enginyeria Mecatrònica

Tutor/a: Pere Martí Puig

Vic, gener de 2023

Agraïments

Agraeixo als meus pares per les idees i suport donat, al meu tutor per l'ajuda i guia que m'ha brindat, a la meva germana per escoltar-me tot i que no entenia el que deia i al meu amic David per ajudar-me a revisar el codi quan no funcionava i no trobava el motiu.

Resum

Títol: P*redicció de comportament d'aerogeneradors mitjançant informació creuada.*
Autora: Xavier Micó Pérez
Tutor: Dr. Pere Martí Puig
Data: Gener del 2023
Paraules clau: *Aerogeneradors, xarxes superficials, models de normalitat, predicció de comportament*

Aquest treball planteja predir el comportament de cinc aerogeneradors mitjançant informació creuada de les seves temperatures de funcionament. Per a aconseguir-ho es netegen els comportaments anòmals dels senyals i es fan servir xarxes superficials per a generar models de normalitat comparant el comportament real dels aerogeneradors amb el comportament predit per les xarxes.

Després de netejar les dades establint llindars d'acceptació, s'exploren les dimensions ideals que han de tenir les xarxes per a obtenir el menor error possible. Un cop trobades, es creen els models de normalitat amb els que s'assoleix predir els comportaments i la volatilitat dels aerogeneradors podent així afirmar que és possible predir el seu comportament mitjançant informació creuada dels aerogeneradors veïns.

Summary

Title: *Wind turbines behavior prediction with crossed information.*
Author: Xavier Micó Pérez
Supervisor: Dr. Pere Martí Puig
Date: January of 2023
Keywords: *Wind turbines, Shallow networks, normality models, behavior prediction*

This work aims to predict the behavior of five wind turbines through cross information of their working temperatures. To achieve this, the anomalous behaviors of the signals are cleaned, and shallow networks are used to generate normality models by comparing the actual behavior of the wind turbines with the behavior predicted by the networks.

After cleaning the data by establishing acceptance thresholds, the ideal dimensions that the networks should have, to obtain the lowest possible error, are explored. Once found, the normality models, with which the behavior and volatility of the wind turbines can be predicted, are created. Thus, being able to state that it is possible to predict their behavior through cross information from the neighboring wind turbines.

Índex de Continguts

Llista de Taules

Llista de Figures

1. Introducció

La reducció dels gasos d'efecte hivernacle i la no dependència de les energies fòssils són objectius principals de molts països (Appavou et al. (2016)) i poden ser aconseguits, entre altres maneres, produint electricitat verda (EUROSTAT 2018) (electricitat obtinguda només de fonts renovables). Les energies renovables són les fonts de subministrament d'electricitat de més ràpid creixement (augment en la seva popularitat i ús) arribant a conformar, el 2019, el 20% de tota l'electricitat generada mundialment (Niklas Hellberg. 2019) i superant en un 6% l'energia elèctrica verda gestada a Europa durant l'any 2020 (David Jones. 2020). Les energies amb un creixement més gran els últims anys són l'energia eòlica i l'energia solar, augmentant un 20% i un 10% respectivament en 2019, i creant una quantitat d'energia a la xarxa de 586 gigavats (GW) i 623 GW corresponentment (Niklas Hellberg. 2019). Per a més de mil milions de persones sense accés a l'electricitat, els projectes d'energies renovables distribuïdes, sobretot en les àrees rurals allunyades de les xarxes centralitzades, ofereixen opcions importants i, normalment, rendibles per a proporcionar aquest accés (SAWIN, Janet L., et al. 2017). L'energia renovable distribuïda és la generació i gestió d'energia elèctrica procedent de fonts renovables de forma descentralitzada, el més proper possible del seu lloc de consum. Això s'aconsegueix amb la creació de parcs o plantes solars i parcs d'aerogeneradors (parcs eòlics) entre d'altres. Un aerogenerador és un generador elèctric que converteix l'energia cinètica del vent en energia mecànica mitjançant una hèlix i, alhora, converteix l'energia mecànica de les hèlix en energia elèctrica a través d'un alternador (generador de corrent elèctric altern).

Un dels impactes econòmics més gran en el cost de l'electricitat dels aerogeneradors (AG) és l'associat a les operacions i tasques de manteniment (Marti-Puig et al. 2022). S'han proposat diferents esquemes per a reduir aquests costos i una visió general de les diferents estratègies actuals pot ser trobada en (Hongzhou Wang 2002), (Pinar et al. 2013), (Zhang i Lu 2019). En l'actualitat, una de les opcions més populars és el Manteniment Predictiu, que consisteix en detectar comportaments anòmals i mal funcionaments en els AG que puguin induir al sistema a una fallida, per corregir-los com més ràpidament millor, evitant danys majors en els subsistemes d'AG i una disminució de l'electricitat generada en el sistema elèctric.

L'estructura d'un sistema de Manteniment Predictiu es pot resumir en tres components principals tal com s'explica a (Iberdrola 2023). El primer component consisteix en els sensors i dispositius connectats a les màquines, en aquest cas als AG, que registren les dades i les envien a una base de dades al núvol o a un servidor remot. El segon component són la base de dades on s'emmagatzema tota la informació obtinguda dels sensors, i els algoritmes de mineria de dades *(data mining)* i *big data*. Aquest segon component es fa servir per a recopilar i analitzar l'enorme quantitat de dades que es genera en un conjunt tan gran com és un parc eòlic. Típicament, els AG tenen un sistema de control de supervisió i adquisició de dades (SCADA) que proporciona una gran quantitat d'informació en supervisar l'estat dels subsistemes. Més específicament, un sistema SCADA recopila informació dels sistemes d'AG i dels seus respectius subsistemes, i analitza i visualitza les dades aconseguint mesures importants com la temperatura, la velocitat de rotació, el consum, etc., sense haver de visitar cada AG per separat (Marti-Puig et al. 2022). El tercer component són els models predictius que s'alimenten amb les dades processades i utilitzen sistemes d'aprenentatge automàtic (*Machine learning*) com poden ser el *shallow learning* i *deep learning* (Figura 1) per a establir patrons i comparacions, elaborar prediccions d'errors i programar manteniments abans que els errors apareguin.

En aquest treball s'explora una estratègia particular, poc tractada a la literatura científica. Fent servir una base de dades amb dades de cinc AG (del mateix fabricant i del mateix model) amb sensors situats a diferents punts de control (els mateixos punts per a cada AG), (Marti-Puig et al.

2021) es planteja predir el comportament d'una variable (per exemple, la temperatura) en un punt d'un AG particular a partir de les mesures obtingudes en el mateix punt dels altres generadors.

Més endavant a (Marti-Puig et al. 2022) van dur a terme un estudi per predir errors dels AG analitzant les dades obtingudes dels altres quatre. Aquest estudi es va concentrar en realitzar models de normalitat per comparar les mostres de dades de les diferents variables de control i analitzar les diferències per a extreure un patró de funcionament. En aquesta primera exploració es pot observar com el que ells anomenen residu, la diferència entre els senyals de cada AG i el senyal de referència (patró), té un gran potencial per a generar alarmes que estarien relacionades amb l'estat de l'AG. A més també van observar que la detecció d'errors no apareixia només en els senyals associats al subsistema analitzat sinó també a la resta de variables del sistema (totes les variables analitzades de l'AG). D'aquesta manera van determinar que la detecció de comportament anòmal no depèn de l'exploració de només un senyal concret, fent possible l'exploració de qualsevol dels senyals.

Aquest treball busca aprofundir en el manteniment predictiu dels AG amb informació creuada d'un grup d'AG veïns tal com es proposa a (Marti-Puig et al. 2022) implementant l'ús de xarxes neuronals superficials o *shallow networks* (SN) per a realitzar models de normalitat.

Un model de normalitat consisteix en la comparació de les dades obtingudes, en aquest cas dels AG, en un comportament de funcionament normal (valors obtinguts de les SN amb una mateixa mitjana i desviació de normalitat) amb el comportament real. El fet d'utilitzar SN per a aquest treball resideix en la simplicitat d'entrenar aquestes xarxes i el poc temps que triguen a processar la informació respecte a una xarxa neuronal profunda o *deep network* (DN).

Una SN és una xarxa neuronal amb només una capa oculta entre les capes d'entrada i sortida mentre que les DN en poden arribar a tenir centenars (Figura 1). Les capes ocultes d'una xarxa neuronal contenen unitats (nodes o neurones) no observables per a l'usuari. El valor de cada unitat oculta és el d'una funció, què depèn, en part, del tipus de xarxa neuronal (IBM 2021). Les SN tenen com a punts positius per sobre de les DN que necessiten molts menys recursos computacionals, són més fàcils d'interpretar i són més resistents al sobre entrenament o *overfitting* (condició que apareix quan un model de xarxa neuronal dona un resultat significativament millor per a les dades d'entrenament del qual dona per a les dades noves) (Anònim, 2021) Les SN són també menys precises que les DN a l'hora de representar dades complexes i menys eficients d'entrenar, per tant, no són òptimes per a tasques grans (Rita. 2022). Com en aquesta investigació es necessita un procés simple de comparació, s'ha triat fer servir una SN en comptes d'una DN.

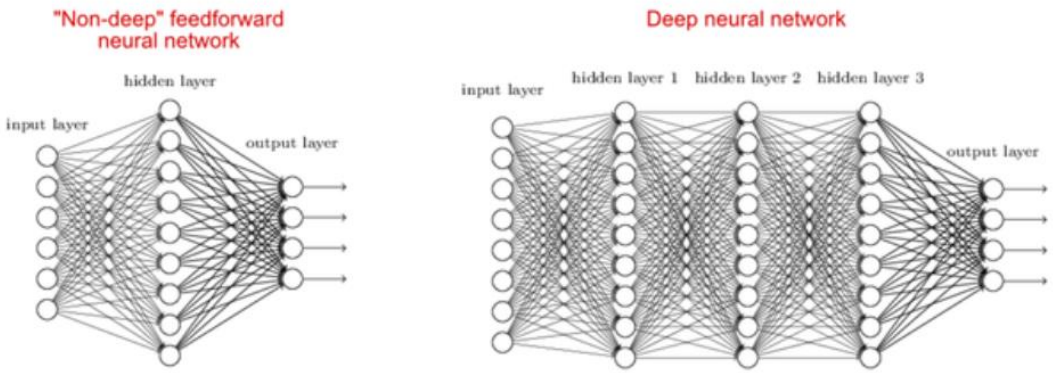


Figura 1. Comparació de capes ocultes de SN amb DN. Imatge extreta de (Rita. 2022).

Aquests models de normalitat permetrien obtenir senyals del comportament d'un AG, mitjançant informació obtinguda de la resta de la població, i programar un senyal d'alarma en cas d'un canvi sobtat en el seu comportament que pogués malmetre el seu funcionament, o d'una possible entrada gradual en una zona de perill per risc de fallida, etc., arribant a poder-se fer servir la informació creuada entre AG com a nou mètode de funcionament predictiu en un futur.

Aprofundir en aquest tipus de mètodes innovadors i poc estudiats pot arribar a aportar noves visions de millora dels mètodes ja existent de Manteniment Predictiu i preventiu, i, alhora pot obrir noves zones inexplorades de coneixement que podrien arribar a generar dades més importants que els mètodes actuals i a reduir els costos de generació d'energia a causa d'una possible reducció en la quantitat de sensors a fer servir, entre d'altres.

Interès per fer el treball

La idea del treball surt de l'estudi previ realitzat a (Marti-Puig et al. 2022). En aquest estudi es van adonar que, quan els AG funcionen correctament, les magnituds de cada màquina, observades de forma síncrona en el temps, evolucionen de forma conjunta seguint el mateix patró de variació (Figura 2).

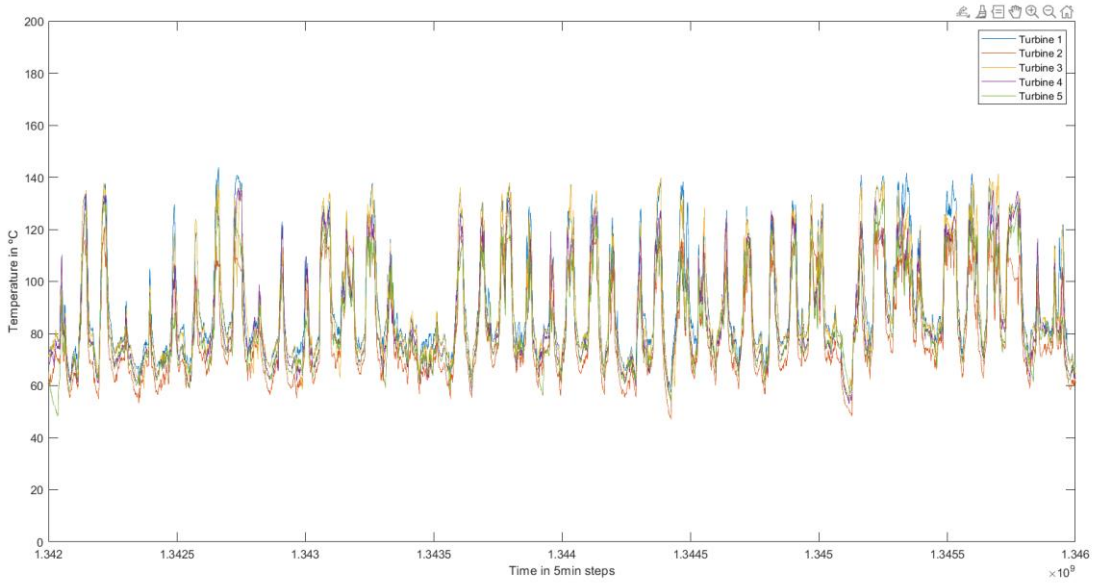


Figura 2. Representació del senyal dels cinc AG en un període de 44 dies.

Les dades visualitzades a la Figura 2 representen les oscil·lacions de la temperatura dels diferents AG durant un període de quaranta-quatre dies. En aquest període, totes les turbines presenten un comportament molt similar, pràcticament idèntic. Analitzant en una finestra de temps més petita, tal com es pot observar a la Figura 3 per a tenir una imatge més clara de la similitud de les dades, es pot veure que tot i existir diferències entre elles, aquestes són pràcticament constants, sent la turbina 1 la de més temperatura i la dos la de menys.

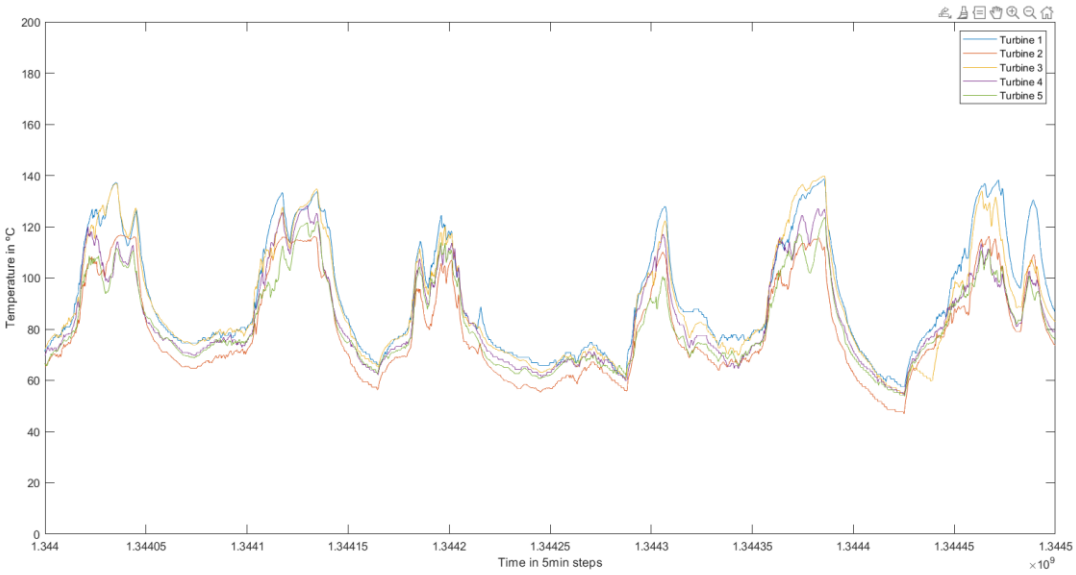


Figura 3. Representació dels senyals dels AG en un període de 5 dies.

A causa d'aquesta similitud entre els cinc AG es pot imaginar que mitjançant la informació creuada es pot arribar a predir el comportament que tindran els AG de manera representativa amb un grau alt d'èxit.

2. Objectius

La finalitat d'aquest treball és estudiar la possibilitat de predir el comportament d'un AG dintre d'una població controlada tal com es proposa a (Marti-Puig et al. 2022), analitzant les dades dels AG veïns.

Per a executar aquesta tasca es plantegen els següents objectius.

I. **Netejar les dades.** Dintre de un grup de dades gran poden existir errors per possibles mal funcionaments dels sistemes sensorials o per a comportament anòmals dels individus analitzats (outliers i dades faltants). En cas d'existir, cal eliminar aquests errors per a obtenir una predicció tan correcta com sigui possible d'un règim de normalitat.
Hipòtesi 1 (H1): Existiran comportaments anòmals en els individus que es consideraran com errors i caldran ser eliminats.

II. **Trobar una xarxa neuronal òptima.** Trobar una xarxa neuronal de dimensions adequades (amb el menor error possible) és molt important pel correcte processament de les dades. Tant una xarxa massa petita com una massa gran portarà a errors a l'hora de manipular les dades.

1- *Hipòtesi 2 (H2): Tenint en compte el que es comenta a (*Sandhya Krishnan. 2021) (Ahmed Gad. 2018) *es creu que el nombre de nòduls ocults correspondrà entre 2 i 4 nodes pel nombre d'inputs (dos) que s'usaran.*

III. **Predicció de comportament mitjançant models de normalitat.** Amb els resultats assolits dels dos objectius anteriors es vol provar d'analitzar si fent servir models de normalitat es pot predir el comportament dels AG de la població d'estudi.
Hipòtesi 3 (H3): Es podrà predir quan un AG començarà a funcionar amb irregularitat.

3. Metodologia

Les dades d'aquest estudi han estat proporcionades pel Departament de Ciències i Tecnologies de la Universitat de Vic i seran processades amb el llenguatge de programació Matlab, versió R2022a. Alguns exemples importants del codi seran afegits en els annexos.

3.1. Àrea d'estudi

Aquest treball pertany al camp de l'enginyeria mecatrònica i estudia la relació de funcionaments que té un grup d'aerogeneradors Fuhrlander per a realitzar prediccions de comportament amb informació creuada.

3.2. Població sobre la qual s'ha fet l'estudi

La base de dades complerta conté la informació, guardada al SCADA durant tres anys, de cinc AG de 2,5 MW Fuhrlaender, model FL2500. Aquest model de turbina consta de 10 sistemes diferents: xarxa, transmissió, generador, convertidor, góndola, sistema hidràulic, rotor, meteorològic, turbina i torre.

Els senyals de les turbines també estan organitzats en 10 grups depenent del sistema al qual pertanyen i els seus indicadors estadístics es registren cada cinc minuts.

Les variables s'emmagatzemen en subtensors dins d'un tensor amb un nom que simbolitza el subsistema i el tipus de variable, separades per un guió baix; el primer terme és el sistema físic principal, per exemple, generador = WGEN, caixa de canvis = WTRM, góndola = WNAC, etc.

Un tensor és un objecte algebraic que descriu una relació multilineal entre conjunts d'objectes algebraics relacionats amb un espai vectorial. Es pot entendre com a una matriu en tres dimensions que conté matrius de dues o tres dimensions en el seu interior.

La representació tensorial de les dades del sistema d'AG es pot observar en les figures 4, 5 i 6.

A la figura 4 es representa el sistema d'AG sencer en forma de tensor format per deu subtensors. Cada subtensor es correspon amb un subsistema. Mirant el tensor frontalment i imaginant que es tracta d'una matriu en dues dimensions (ignorant el gruix dels subtensors) es pot entendre que cada columna d'aquesta matriu correspon a un subsistema. Dels deu subsistemes només es fan servir dades del subsistema 1 o WGEN (blau) (seguint l'ordre d'exemple de la figura 2).

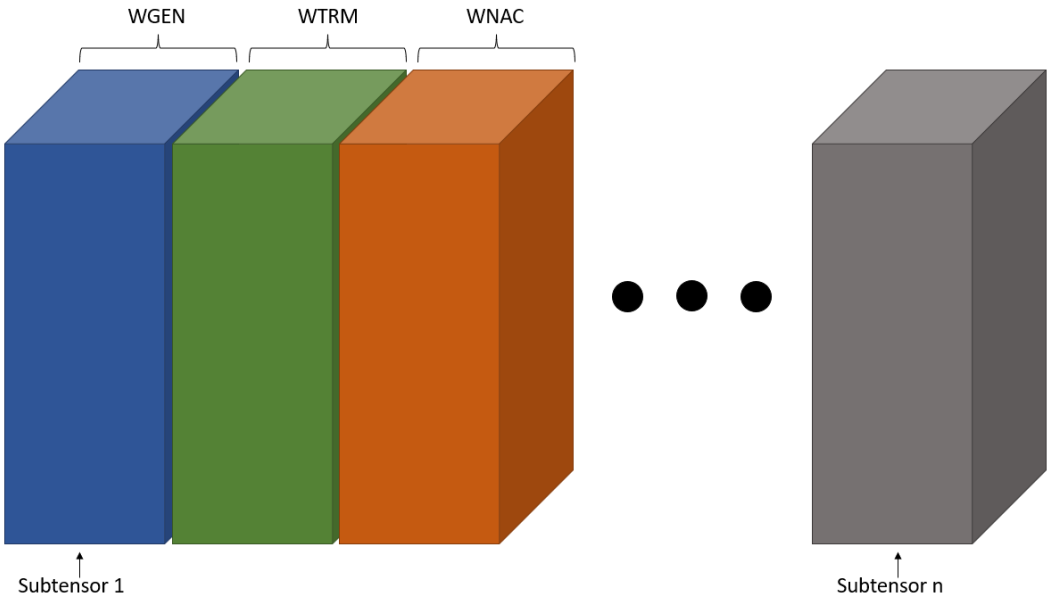


Figura 4. Estructuració de les dades del sistema d'AG en un tensor.

Dins dels subtensor 1 també hi ha informació dividides en seccions tal com es pot veure en la figura 5. Mirant el subtensor 1 frontalment de nou i dividint-lo en sis llesques, s'obté una llesca de dades de cada una de les sis variables del subtensor 1 que contenen informació sobre diferents variables del generador de cada un dels AG. En aquest treball només es farà servir la variable 1 del subtensor WGEN (llesca groga), que correspon a la temperatura de la fase 1 del generador.

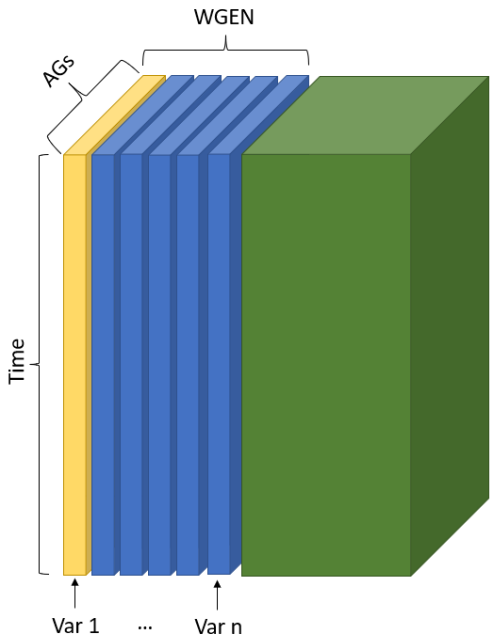


Figura 5. Desglossament del subtensor 1 o WGEN.

Cada una de les llesques extretes del subtensor 1 correspon a una matriu de dimensions 200000x5x1 aproximadament (Files x Columnes). A la figura 6 es pot observar el desglossament de la informació de la variable 1 usada en aquest treball. Rotant la matriu, per a poder tenir una vista frontal més entenedora, es pot visualitzar com la primera columna (taronja) de dades conté informació sobre les temperatures de la fase A de la turbina 1 (AG80) classificada en ordre ascendent (començant a mirar des de l'extrem superior) segons el temps de mesura. La primera mesura es troba a l'extrem superior de la columna i l'última a l'inferior.

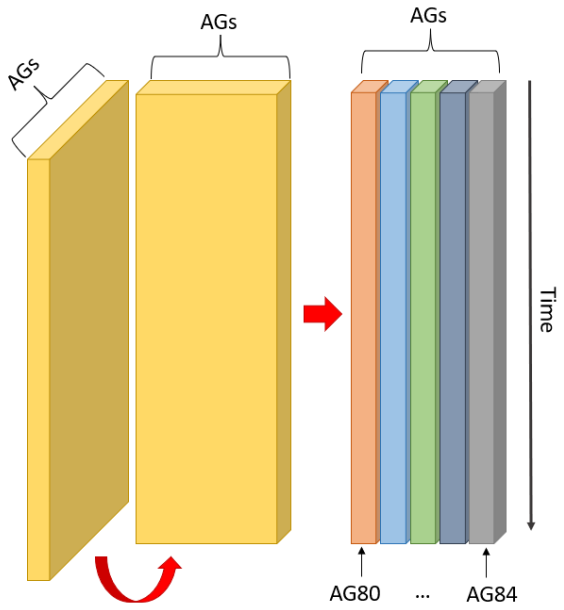


Figura 6. Matriu temperatura de fase A dels AG.

Com s'ha explicat a la introducció del treball a (Marti-Puig et al. 2022) s'observa que només cal analitzar una de les variables per a trobar un comportament anòmal del sistema sencer quan la fallada és molt important. Per aquest motiu el treball es centra a fer servir la variable 1 dins del subsistema WGEN (Taula 1). No obstant això, realitzar l'estudi per a la resta de variables significa reproduir el mateix procediment.

3.3. Entorn

Tot el processament de les dades s'ha fet entre dos ordinadors. Per als treballs més lleugers s'ha fet servir un ordinador portàtil Honor MagicBook 16 amb un processador AMD Ryzen 5 5600H de 6 nuclis a una velocitat interna de refresc de 3,3 GHz i una memòria RAM de 16GB DDR4, en el sistema operatiu Windows 11 Pro 64 bits. Per a les tasques més pesades com el processament de les SN, s'ha utilitzat un ordinador de sobretaula amb un processador AMD Ryzen 2700X de vuit nuclis a una velocitat de refresc interna de 4 GHz i una memòria RAM instal·lada de 16 GB DDR4 amb un sistema operatiu Windows 10 Pro 64 bits.

3.4. Intervencions

Objectiu 1: Netejar les dades

Es busca filtrar i eliminar els errors que puguin existir en les dades abans de fer el seu estudi.

El primer pas per assolir l'objectiu 1 és dividir les dades de cada tensor temperatura de fase A en dos subtensors tal com s'explica a (Martí-Puig et al. 2018) El vector amb les dades corresponents a la primera meitat es fa servir per a entrenar les xarxes neuronals en l'objectiu 2. Aquestes dades han de correspondre a un comportament normal dels AG, de manera que si existeixen comportaments anòmals o errors hauran de ser eliminats. La segona meitat de les dades es reservarà per a testejar les xarxes entrenades.

Per a saber si la primera meitat de les dades conté errors, es representaren les dades en una gràfica i es realitzarà una estimació visual on s'avalua com es comporten els diferents AG i, si existeixen zones amb valors diferents del comportament determinat com a normal, s'eliminaran aquestes dades de tots els AG. En cas d'existir aquests comportaments anòmals o errors de funcionament s'establirà de manera arbitrària una temperatura llindar inferior i/o una temperatura llindar superior que delimitaran el rang de temperatures establertes com a comportament normal i s'eliminaran les temperatures fora d'aquest rang.

Les dades de temperatura del generador, fase A es guardaran en deu vectors verticals, dos per a cada AG a analitzar (primera i segona meitat) i es recorreran els cinc vectors corresponents a la primera meitat de les dades (vectors primera meitat) buscant valors inferiors i superiors als llindars establerts. En cas que es trobin es guardarà l'índex de posició dels corresponents valors en un vector per a cada AG (vectors d'índex).

Es crearan cinc vectors de dimensions iguals als vectors primera meitat (vectors candidats) i es modificaran a 1 els valors dels índexs corresponents a les temperatures exteriors als llindars establerts, guardades en els vectors d'índex. Un cop estiguin tots els valors modificats, es recorreran els cinc vectors candidats i es buscarà si tenen un valor d'1 a la mateixa posició. En cas de tots tenir un 1 a la mateixa posició, s'entendrà que el comportament dels AG, tot i ser fora dels valors llindar, no és anòmal sinó forçat per causes externes i es canviaran els valors a 0 de nou, tornant-se a considerar aquest comportament com a normal. Un exemple d'aquesta situació pot ser una parada dels cinc AG per a evitar una sobrecàrrega del sistema elèctric o per evitar ruptura dels alternadors a causa de vents massa forts.

Quan els valors dels vectors candidats siguin normals, aquests vectors s'uniran en un de sol (vector errors) amb un criteri de porta lògica or. Si un dels vectors té un 1 en una posició, el valor del vector final passa a ser 1. Això es farà perquè les xarxes neuronals del objectiu 2 necessiten que tots els vectors que es facin servir tinguin el mateix nombre de valors.

Per a finalitzar aquest objectiu es buscaran els índexs dels vectors primera meitat corresponents als índexs amb valor 0 del vector errors (índex amb valors de funcionament normal) i es guardaran els valors dels vectors primera meitat en cinc nou vectors amb només valors de comportament normal (vectors comportament normal). Aquests cinc vectors comportament normal seran els que es faran servir per a entrenar les xarxes neuronals SN en l'objectiu 2.

Objectiu 2: Trobar una xarxa neuronal òptima.

Es busca trobar les millors dimensions per a la capa interna de la xarxa neuronal per a reduir errors.

Per a crear les xarxes neuronals es farà servir el model de Matlab feedforwardnet. Caldrà establir la quantitat òptima de nodes o neurones que ha de tenir la capa interna o *hidden layer* tant per a evitar l'*overfitting* com per a maximitzar la velocitat de processament de les xarxes i disminuir el temps d'espera.

El primer pas serà crear cinc configuracions de dades per l'entrenament de les xarxes. Per a fer aquestes configuracions es seleccionaran de l'AG 80 a l'AG 83 i es guardaran els seus vectors comportament normal en una llista. Aquesta llista serà els valors d'entrenament de la SN encarregada de predir el comportament de l'AG 84, i el vector configuració normal faltant el corresponent a l'AG 84, que serà l'objectiu o *target* a predir. Les cinc configuracions resultants es poden veure a la Taula 1.

Taula 1. Configuracions d'entrenament de les SN

Configuració	Dades d'entrenament	*Target*
c.1	T(nets(AG 81 – 84))	T(nets(AG 80))
c.2	T(nets(AG 80, 82 – 84))	T(nets(AG 81))
c.3	T(nets(AG 80, 81, 83, 84))	T(nets(AG 82))
c.4	T(nets(AG 80 - 82, 84))	T(nets(AG 83))
c.5	T(nets(AG 80 – 83))	T(nets(AG 84))

T* Indica que els vectors han de ser transposats.

Com les *feedforwardnet* funcionen amb les dades en forma de fila en comptes de columna, i tant els vectors configuració normal com els *target* estan disposats en forma de columna, es transposaran abans d'entrar-los a les SN.

De la mateixa manera es prepararan les dades per a testejar el resultat de l'entrenament de les SN amb la segona meitat de les dades dels AG, les que no s'han netejat. Aquesta vegada només s'hauran de transposar les llistes ja que els *target* es faran servir d'una manera diferent (Taula 2).

Taula 2. Configuracions de test de les SN

Configuració	Dades d'entrenament	*Target*
ct.1	T(AG 81 – 84)	AG 80
ct.2	T(AG 80, 82 – 82)	AG 81
ct.3	T(AG 80, 81, 83, 84)	AG 82
ct.4	T(AG 80 - 82, 84)	AG 83
ct.5	T(AG 80 – 83)	AG 84

T* Indica que els vectors han de ser transposats.

Les SN, tot i ser més ràpides que les DN, segueixen consumint una gran quantitat de recursos de l'ordinador i triguen molt temps a processar-se. Per a reduir aquest temps es farà una primera exploració del funcionament de les SN amb quatre quantitats de nodes diferents: 5, 10, 15 i 20. Per a cada una de les configuracions de nodes es crearan deu variables "cel·la" buida de dimensions 30x1, una per a cada configuració d'entrenament i test per a poder guardar els valors resultants dels errors de les SN. El motiu d'utilitzar les dimensions 30x1, serà realitzar un estudi de repetibilitat amb els resultats obtinguts de les SN.

Un estudi de repetibilitat necessita un mínim de vint-i-cinc mostres per a poder ser considerat efectiu, però per la possibilitat de que les SN es desconfigurin en una o més de les repeticions i

donin com a resultat un valor exageradament gran o petit, s'eliminaran els valors màxim i mínim dels trenta valors d'error finals. El número trenta s'ha triat per a poder deixar un marge per a eliminar aquests valors i continuar tenint una quantitat estadísticament fiable.

El procés sencer per a obtenir els errors de les SN constarà de cinc passos: entrenar les SN, testejar-les, repetir aquests dos passos trenta vegades per l'estudi de repetibilitat, trobar l'error i, finalment, analitzar els resultats.

Per l'entrenament de les SN es crearà la SN especificant el nombre de nodes que tindrà la capa interna. Un cop creada, es cridarà i s'introduiran, les dades d'entrenament corresponents com a dades de treball i les dades *target* com a objectiu a comparar.

A l'hora de testejar les SN amb les configuracions de test, es cridarà la SN ja entrenada, se li introduiran les dades de test i s'assoliran els resultats estimats.

Una SN té tres paràmetres de configuració, el paràmetre d'entrenament (que determina quin percentatge de les dades es farà servir per a entrenar la SN), el de validació (que serveix per a avaluar el seu rendiment) i el de test (per a provar que la SN funciona correctament), entre els que s'han de distribuir totes les dades. La distribució final consistirà en:

- Paràmetre d'entrenament: 100% de les dades.
- Paràmetre de validació: 0% de les dades.
- Paràmetre de test: 0% de les dades.

Com en aquest cas les dades ja s'hauran dividit en dades d'entrenament i dades de test abans d'entrenar les SN, no caldrà designar cap percentatge de dades al paràmetre de test. De la mateixa manera, el que es buscarà en aquest objectiu serà trobar els errors generats per cada SN amb diferent configuració de nodes, per a saber quina serà la millor configuració, implicant que no caldrà fer servir el paràmetre de validació per a aconseguir informació dels rendiments.

Quan les xarxes acabin de processar es guardaran els valors corresponents als errors de les SN de cada una de les trenta iteracions a cada variable cel·la, i s'eliminaran els valors màxims i mínims. Amb aquestes dades es farà la Mitjana i es trobarà la Desviació Estàndard dels errors que servirà per a fer una gràfica de tipus *box-plot*, realitzar un examen visual dels resultats, i decidir en quin interval de quantitat de nodes aprofundir. Per exemple, si en les gràfiques surt que el valor d'error més baix correspon als 10 nodes i els 5 nodes tenen el següent error més baix, es revisarà entre les quantitats 6, 7, 8 i 9 nodes, comparant els resultats amb els 5 i 10 nodes per a veure si realment el valor més petit continua sent el corresponent a 10 nodes, o si pertany a un altra quantitat. Per a revisar els valors 6, 7, 8, 9 de nodes es seguirà exactament el mateix procediment anterior:

1. Entrenar les xarxes restants amb els paràmetres mencionats amb anterioritat.
2. Testejar les SN amb els valors de la segona meitat.
3. Repetir 30 vegades aquests dos passos.
4. Trobar l'error i la desviació Estàndard
5. Analitzar els resultats en forma de gràfica per a trobar el error més petit.

El criteri per a decidir quina quantitat de nodes és el òptim serà comparar en cada SN quin nombre de nodes correspon al valor més petit i triar el més recorrent com a millor resultat. En cas de repetir-se i no haver-hi una clara decisió, es decidirà arbitràriament entre els candidats a millor resultat el que es considerarà com a òptim per a poder prosseguir amb el treball.

Objectiu 3: Predicció de comportament mitjançant models de normalitat.

Es pretén predir el comportament de la població d'estudi mitjançant models de normalitat.

Per a generar els diferents models de normalitat es compararan punt a punt els resultats obtinguts de les SN amb els funcionaments reals dels AG per a crear una diferència. Aquesta diferència hauria de mostrar amb claredat el comportament de l'AG corresponent, però a causa de la naturalesa de les dades, és molt probable que al fer el gràfic per l'anàlisi visual d'aquest comportament hi hagi molt soroll. Per a eliminar aquest soroll i aconseguir una visió més clara dels comportaments de les turbines es farà un *smooting* de les dades aconseguides amb un procediment de *moving average,* o mitjana mòbil, de tipus *sliding box*.

El *sliding box* es correspon amb un mètode de la funció movAVG de Matlab que serveix per a calcular la mitjana mòbil. Aquest mètode realitza la mitjana de tots els valors dins de la finestra de dimensions determinades. Cada finestra es correspon amb un període de la fórmula estadística. La funció *sliding box* dona el mateix pes a cada una de les finestres o períodes, corresponent la *mooving average* de tipus *sliding box* del Matlab amb una mitjana mòbil simple.

Es calcularan les Desviacions Estàndard mòbils superiors e inferiors corresponents (mateix principi que la mitjana mòbil) per a tenir una visualització més clara de la variació que poden tenir els comportaments dels AG.

3.5. Anàlisi estadístic

Les matemàtiques fetes servir per a aquest treball són, en la seva majoria, molt simples i directes. En elles s'inclou trobar l'error quadràtic mig de les SN, calcular la Mitjana i les Desviacions Estàndard dels errors per a trobar la quantitat de nodes que proporcionen un millor funcionament de les xarxes i elaborar la mitjana mòbil simple per a suavitzar els errors resultants i trobar comportaments anòmals.

Les SN han estat entrenades mitjançant una funció d'entrenament que es basa en l'algoritme Levenberg-Marquardt que s'explicarà juntament amb els conceptes anteriors.

Error absolut:

L'error absolut (EA) és la diferència total entre el valor real i el valor aproximat.

$$EA = \Delta x = x_0 - x$$

On: x_0 = valor mesurat; x = valor real.

Error quadràtic mig:

L'error quadràtic mig (EMC) és un mètode estimador que mesura la mitjana dels errors al quadrat, és a dir, la diferencia entre l'estimador i el que s'estima.

La fórmula de l'EMC és la següent:

$$EMC = \frac{1}{n}\,(\widehat{\boldsymbol{Y}} - \boldsymbol{Y})^T * (\widehat{\boldsymbol{Y}} - \boldsymbol{Y})^T$$

On: n = Nombre de mostres; $\widehat{\boldsymbol{Y}}$ = Vector estimador (vector predicció); **Y** = Vector real (vector *target).*

Mitjana i Desviació Estàndard:

Donat un grup de *n* números, la mitjana o mitjana aritmètica es defineix com la suma de tots aquests números dividits entre *n.*

Es representa amb la fórmula:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

On: x = valors dels quals fer la mitjana; n = nombre total de valors.

La Desviació Estàndard és defineix com l'arrel quadrada de la variància de X, sent X una variable aleatòria. S'utilitza per a quantificar la variació o dispersió d'un conjunt numèric de dades.

La seva representació és:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

On: μ = mitjana; xi = valor del qual calcular la desviació; n = nombre total de valors.

Mitjana mòbil simple:

La mitjana mòbil simple (SMA) (Jason Fernand et al. 2022) es calcula prenent la mitjana d'un rang de valors en un període específic. Aquesta operació es repeteix fins a cobrir totes les dades i diferents períodes. Finalment, es sumen les mitjanes i es divideixen entre el nombre de períodes total.

$$SMA_n = \frac{p_{n-k+1} + p_{n-k+2} + \cdots + p_n}{k} = \frac{1}{k}\sum_{i=n-k+1}^{n} p_i$$

On: p = Mitjana en un període*; k* = Nombre de períodes.

Desviació Estàndard mòbil:

La Desviació Estàndard mòbil es calcula trobant la Desviació Estàndard d'un rang de valors en un període específic (Anònim, 2023). El concepte és el mateix que el de la SMA.

$$DSM_n = \frac{\sigma_{n-k+1} + \sigma_{n-k+2} + \cdots + \sigma_n}{k} = \frac{1}{k} \sum_{i=n-k+1}^{n} \sigma_i$$

On: σ = Desviació Estàndard en un període; *n* = Nombre de períodes.

Mètode Levenberg-Marquardt:

El mètode Levenberg-Marquardt,(LMA) també conegut com a mètode dels límits quadrats esmorteïts s'utilitza per a resoldre problemes de mínims quadrats no lineals que sorgeixen especialment en l'ajustament de corbes de mínims quadrats. El LMA interpola entre l'algoritme de Gauss-Newton i el mètode de descens de gradient.

La representació d'aquest mètode és:

$$S_i = -[H + \lambda I]^{-1} \nabla \mathrm{f}(x_i)$$

On: H = Matriu gaussiana; I = matriu identitat; λ = Valor escalar amb un valor gran a l'inici de l'algoritme que s'altera a cada iteració del algoritme; ∇ = Gradient.

El funcionament del mètode Levenberg-Marquardt es pot resumir en sis passos (OrvizarTV. 2021):

1. Definir:
    a. x_i.
    b. ϵ_1 → Tolerància per a la convergència del mètode.
    c. ϵ_2 → Tolerància pel valor del gradient.

2. Calcular: $S_i = -[H + \lambda I]^{-1} \nabla \mathrm{f}(x_i)$

3. Actualitzar $Xi + 1 = Si + Xi$

4. Comparar:
    a. $f(Xi + 1) < f(Xi) \rightarrow \lambda i + 1 = 1/2\lambda i$
    b. $f(Xi + 1) > f(Xi) \rightarrow \lambda i + 1 = 2\lambda i$

5. Acabar cicle iteratiu comparant amb toleràncies.
    a. $|f(Xi + 1 - f(Xi)| > \epsilon_1$ *o* $|\nabla f(Xi)| > \epsilon_2$ → retornar al punt 2.
    b. $|f(Xi + 1 - f(Xi)| < \epsilon_1$ *i* $|\nabla f(Xi)| < \epsilon_2$ → avançar al punt 6.

6. Donar el resultat $Xi + 1, f(Xi + 1)$.

4. Resultats

Objectiu 1: Netejar les dades

El primer objectiu del treball era filtrar i eliminar els errors en cas de trobar-ne.

Al representar gràficament la primera meitat dels valors dels senyals de temperatura del generador, fase A Figura 7, es pot observar com apareixen comportaments anòmals en tots els AG. Les zones de comportament anormal o zones d'error poden ser observades en l'última de les gràfiques (error regions), representades per àrees de color vermell. Cadascuna d'aquestes àrees representa una zona d'error en, com a mínim, un dels AG.

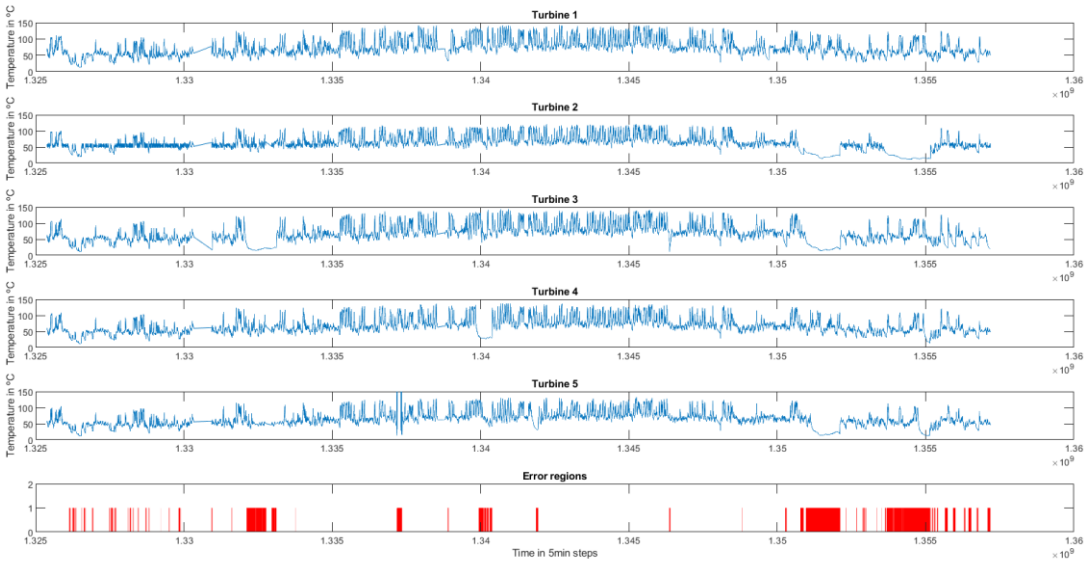


Figura 7. Senyals de temperatura del generador, fase A dels AG sense netejar.

Mitjançant la gràfica de la turbina 5 o AG84 (Figura 8) es pot entendre amb més claredat els conceptes de zona de comportament normal i anòmal. En la regió de comportament normal es pot veure com les temperatures oscil·len en un rang d'entre 40 ºC i 130 ºC aproximadament. Quan la temperatura baixa per sota dels 40 graus es considera que el funcionament és anormalment baix i, per tant, un error de funcionament de la turbina o AG. De la mateixa manera, si la temperatura augmenta dels 130 graus Celsius, aproximadament, es considera una temperatura anormalment alta i, també, un error de funcionament de l'AG.

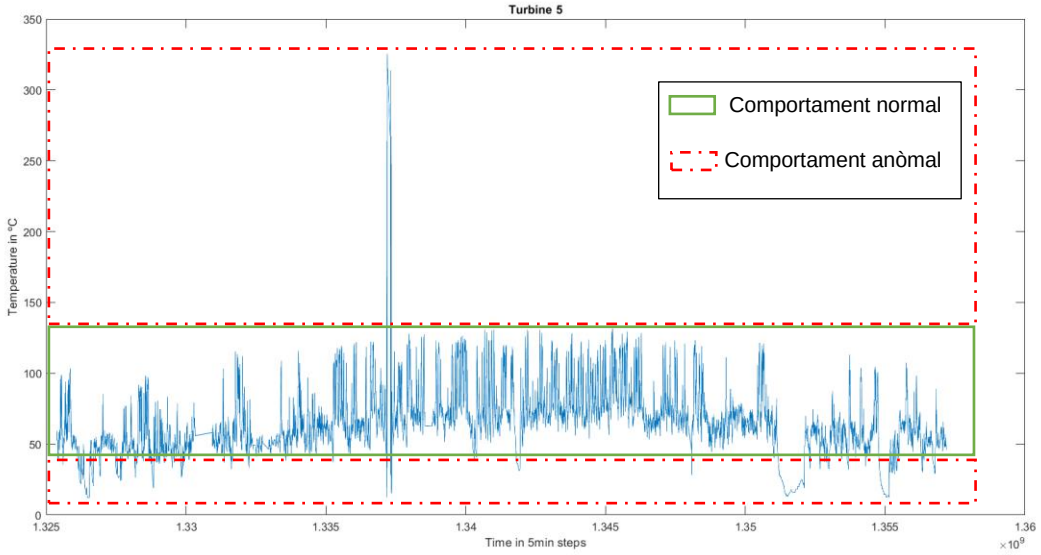


Figura 8. Representació de les zones de funcionament anòmal i normal de la turbina 5 o AG84.

Per tant, observant amb deteniment la figura 9 es pot veure com les zones vermelles corresponen correctament amb una de les zones anòmales, ja sigui per temperatura elevada (corresponent línia groga prima, difícil d'apreciar degut a l'escala de les gràfiques) o per temperatura baixa (línies gruixudes).

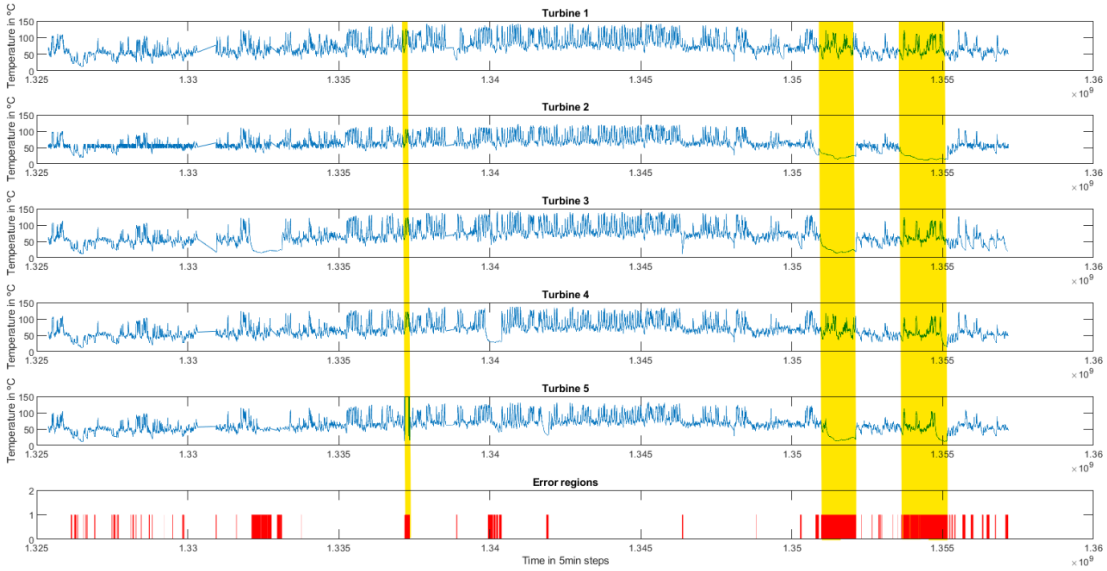


Figura 9. Visualització de les regions de funcionament anòmal a les gràfiques de les turbines.

Com la informació d'aquestes regions d'error no es pot fer servir per a l'entrenament de les SN s'han hagut d'eliminar els valors anòmals, tant els pics de temperatura elevada com les valls de baixa temperatura. Les gràfiques de comportament normal resultants es poden contemplar a la Figura 10. A les regions on abans hi havia comportament anormal, ara hi ha una línia recta que indica la falta de valors eliminats al netejar.

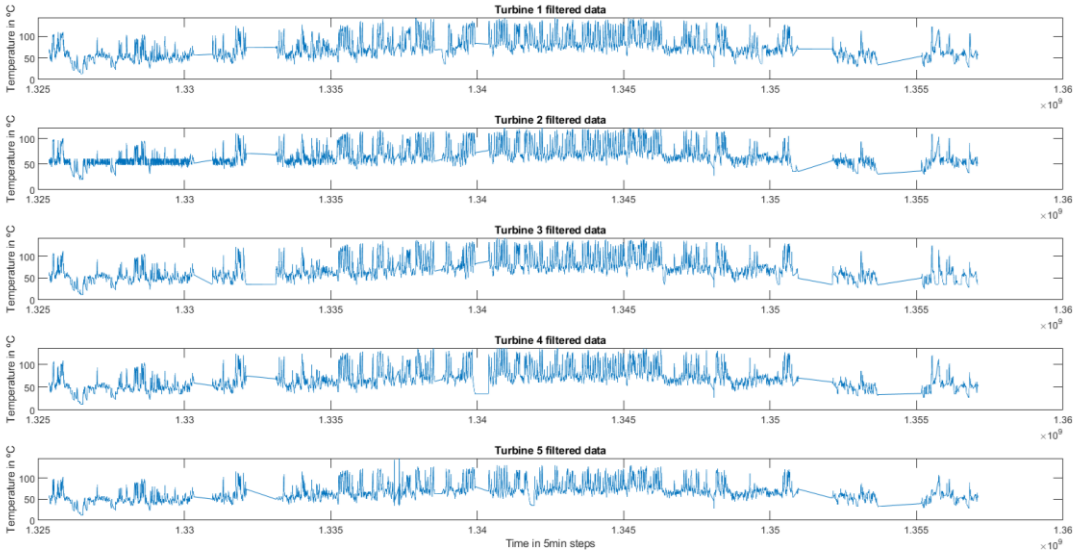


Figura 10. Primera meitat dels senyals de temperatura de la fase A del generador filtrades

El codi fet servir per a netejar i representar les dades es presenta a l'Annex A.

Objectiu 2: Trobar una xarxa neuronal òptima.

L'objectiu 2 buscava trobar les millors dimensions per a la capa interna de la xarxa neuronal per tal de reduir errors.

Amb els resultats obtinguts de l'estudi de repetibilitat s'ha realitzat la mitjana i la desviació Standard per a cada SN i cada configuració d'AG. El codi fet servir per a extreure la Mitjana i la Desviació Estàndard de les quantitats de nodes 5, 10, 15 i 20 es presenta a l'Annex C.

Els valors obtinguts s'han guardat a una taula (Taula 3).

Taula 3 Mitjanes i Desviacions Estàndard obtingudes de les SN per a les diferents configuracions i quantitats de nodes 5, 10, 15 i 20.

Nodes	Resultats entrenament					Resultats test				
SN	c.1	c.2	c.3	c.4	c.5	ct.1	ct.2	ct.3	ct.4	ct.5
5	38.049*	33.921	24.854	21.591	31.200	215.226	171.542	198.163	280.122	903.505
	*(0.322)***	*(0.384)*	*(0.431)*	*(0.271)*	*(0.250)*	*(28.014)*	*(27.028)*	*(57.575)*	*(31.489)*	*(59.836)*
10	36.265	32.421	23.706	20.612	30.074	248.248	212.772	210.820	271.864	907.798
	(0.382)	*(0.328)*	*(0.168)*	*(0.206)*	*(0.179)*	*(44.303)*	*(62.464)*	*(123.771)*	*(42.262)*	*(59.105)*
15	35.295	31.615	23.270	20.095	29.322	285.431	299.040	338.974	274.351	929.211
	(0.324)	*(0.223)*	*(0.169)*	*(0.221)*	*(0.238)*	*(127.713)*	*(187.727)*	*(504.566)*	*(34.593)*	*(57.735)*
20	34.509	31.040	22.856	19.657	28.7872	325.640	325.952	374.895	303.280	950.874
	(0.322)	*(0.238)*	*(0.198)*	*(0.266)*	*(0.271)*	*(241.170)*	*(149.329)*	*(563.157)*	*(86.690)*	*(95.907)*

***Mitjana, **Desviació Estàndard.**

Els errors dels entrenaments són més baixos que els errors de test de les SN i, analitzant els resultats del test corresponents a la Mitjana, es pot veure com a les configuracions ct.1, ct.2, ct.3 i ct.5 la quantitat de nodes amb un error més baix és 5 mentre que a ct.4 és 10.

Com les millors configuracions de nodes s'han trobat entre els valors 5 i 10, s'ha realitzat també la Mitjana i la Desviació Estàndard de les quantitats de nodes 6, 7, 8, i 9. Els resultats obtinguts es poden veure a la Taula 4.

Taula 4. Mitjanes i Desviacions Estàndard obtingudes de les SN per a les diferents configuracions quantitats de nodes 6, 7, 8 i 9.

Nodes	Resultats entrenament					Resultats test				
SN	c.1	c.2	c.3	c.4	c.5	ct.1	ct.2	ct.3	ct.4	ct.5
6	37.522*	33.449	24.284	21.284	30.894	221.604	176.670	178.792	268.574	920.632
	*(0.441)***	*(0.384)*	*(0.491)*	*(0.256)*	*(0.234)*	*(35.550)*	*(27.028)*	*(29.001)*	*(26.161)*	*(62.924)*
7	37.097	33.159	24.460	21.070	30.575	220.604	186.664	209.647	284.486	903.149
	(0.453)	*(0.373)*	*(0.462)*	*(0.210)*	*(0.204)*	*(33.999)*	*(32.491)*	*(87.305)*	*(57.723)*	*(67.530)*
8	36.446	32.819	24.862	20.928	30.375	242.300	182.941	215.289	262.867	920.900
	(0.356)	*(0.390)*	*(0.382)*	*(0.190)*	*(0.386)*	*(83.885)*	*(39.248)*	*(50.962)*	*(33.196)*	*(59.642)*
9	36.265	32.611	25.198	20.769	30.283	233.786	203.509	196.211	263.143	911.270
	(0.357)	*(0.258)*	*(0.443)*	*(0.189)*	*(0.241)*	*(40.242)*	*(41.968)*	*(60.674)*	*(38.255)*	*(52.762)*

***Mitjana, **Desviació Standard.**

Comparant tots els valors de Mitjanes obtinguts es pot veure que les millors quantitats de nodes es corresponen amb 5 nodes per ct.1, 5 per ct.2, 6 per ct.3, 8 per ct.4 i 7 per ct.5 (Taula 5).

Taula 5. Mitjanes i Desviacions Estàndard obtingudes de les SN per a les diferents configuracions quantitats de nodes 5, 6, 7, 8, 9 i 10.

Nodes	Resultats entrenament					Resultats test				
SN	c.1	c.2	c.3	c.4	c.5	ct.1	ct.2	ct.3	ct.4	ct.5
5	38.049*	33.921	24.854	21.591	31.200	215.226	171.542	198.163	280.122	903.505
	*(0.322)***	*(0.384)*	*(0.431)*	*(0.271)*	*(0.250)*	*(28.014)*	*(27.028)*	*(57.575)*	*(31.489)*	*(59.836)*
6	37.522*	33.449	24.284	21.284	30.894	221.604	176.670	178.792	268.574	920.632
	*(0.441)***	*(0.384)*	*(0.491)*	*(0.256)*	*(0.234)*	*(35.550)*	*(27.028)*	*(29.001)*	*(26.161)*	*(62.924)*
7	37.097	33.159	24.460	21.070	30.575	220.604	186.664	209.647	284.486	903.149
	(0.453)	*(0.373)*	*(0.462)*	*(0.210)*	*(0.204)*	*(33.999)*	*(32.491)*	*(87.305)*	*(57.723)*	*(67.530)*
8	36.446	32.819	24.862	20.928	30.375	242.300	182.941	215.289	262.867	920.900
	(0.356)	*(0.390)*	*(0.382)*	*(0.190)*	*(0.386)*	*(83.885)*	*(39.248)*	*(50.962)*	*(33.196)*	*(59.642)*
9	36.265	32.611	25.198	20.769	30.283	233.786	203.509	196.211	263.143	911.270
	(0.357)	*(0.258)*	*(0.443)*	*(0.189)*	*(0.241)*	*(40.242)*	*(41.968)*	*(60.674)*	*(38.255)*	*(52.762)*
10	36.265	32.421	23.706	20.612	30.074	248.248	212.772	210.820	271.864	907.798
	(0.382)	*(0.328)*	*(0.168)*	*(0.206)*	*(0.179)*	*(44.303)*	*(62.464)*	*(123.771)*	*(42.262)*	*(59.105)*

***Mitjana, **Desviació Standard.**

La irregularitat dels resultats no permet triar amb precisió una quantitat de nodes que serveixi per a tots els senyals dels AG per igual, així que s'ha triat la quantitat de nodes 6 com la més bona per ser aquest el valor central.

Objectiu 3: Predicció de comportament mitjançant models de normalitat.

L'objectiu 3 buscava predir els comportaments dels AG mitjançant models de normalitat.

En totes les gràfiques que es presenten en els resultats de l'objectiu 3 es troben representades només valors de la segona meitat de dades dels AG, ja que la primera meitat s'ha utilitzat per a l'entrenament de les SN i, per tant, no s'ha pogut fer servir per als models de normalitat.

Després de determinar 6 com a millor quantitat de nodes per a la capa interna de les SN en l'objectiu anterior, s'han creat cinc models de normalitat, un per a cada AG, calculant l'error absolut, punt a punt, dels processos de test de la SN de 6 nodes. Els resultats obtinguts es poden veure representats en la figura 11.

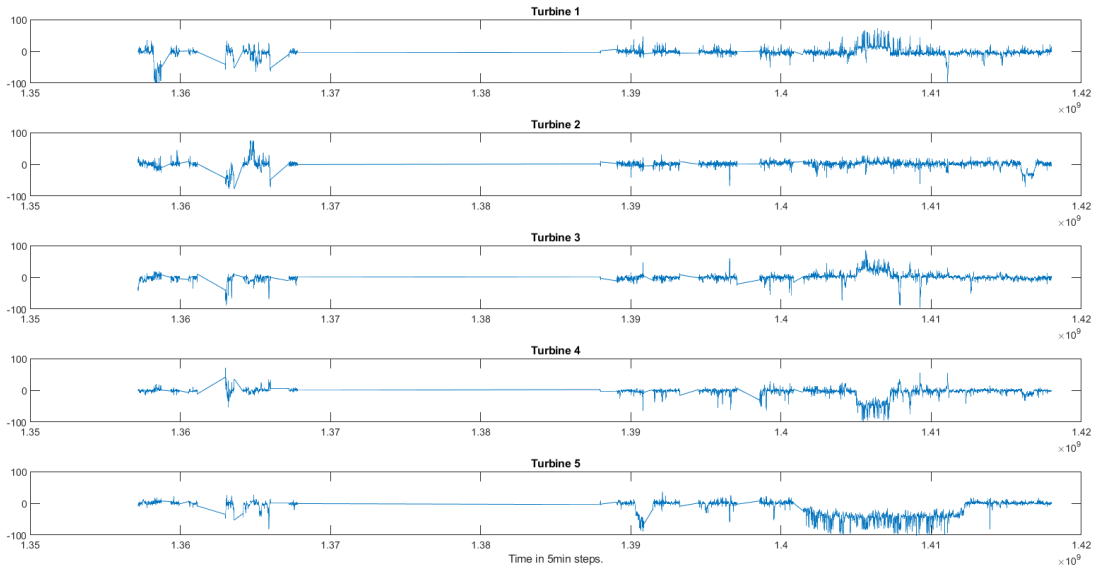


Figura 11. Representació dels errors absoluts dels senyals dels AG.

Aquests senyals d'error no permeten veure els comportaments dels diferents AG amb molta claredat per culpa del soroll (oscil·lacions del senyal) que presenten. Una representació més clara d'aquests models es pot trobar a la figura 12, on es representen els mateixos valors després de realitzar-se una mitjana mòbil simple (MMS) per a suavitzar els senyals.

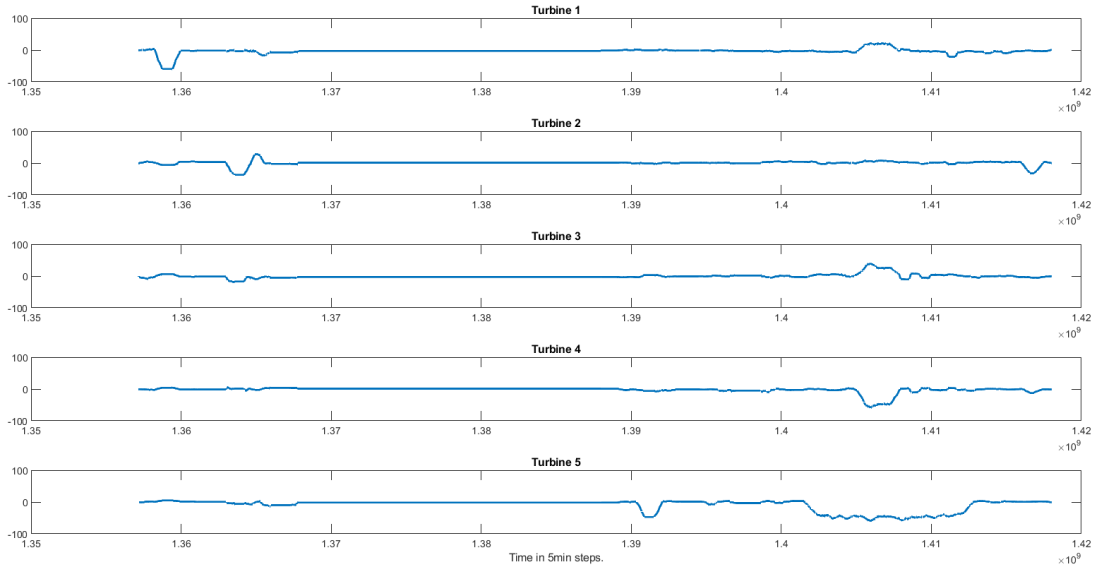


Figura 12. Mitjana mòbil simple dels models de normalitat.

En aquestes gràfiques s'ha fet servir un període de 2000 mostres on cada mostra s'ha pres en un interval de temps de 5 minuts. Això vol dir que el període de mostreig correspon a set dies. En la figura 13 es pot observar una comparació dels models de normalitat sense suavitzar (blau) amb les representacions de les MMS (vermell). Centrant-se en la gràfica de la turbina 2 (Figura 14) es pot contemplar més fàcilment el senyal suavitzat resultant d'aplicar una MMS sobre un senyal d'error.

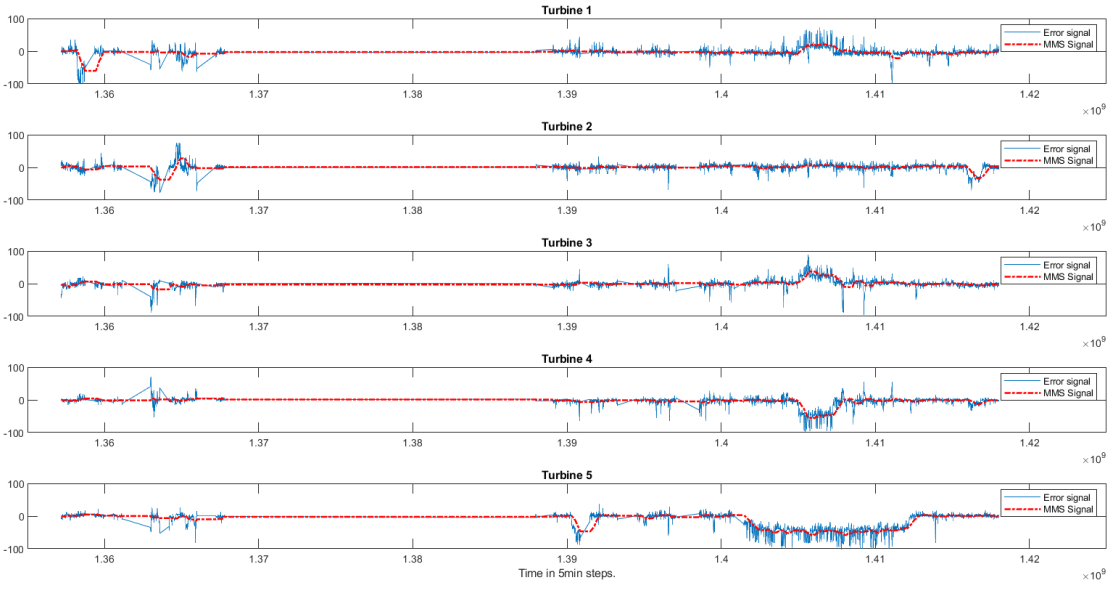


Figura 13. Comparació dels models de normalitat i les senyals suavitzades de les MMS.

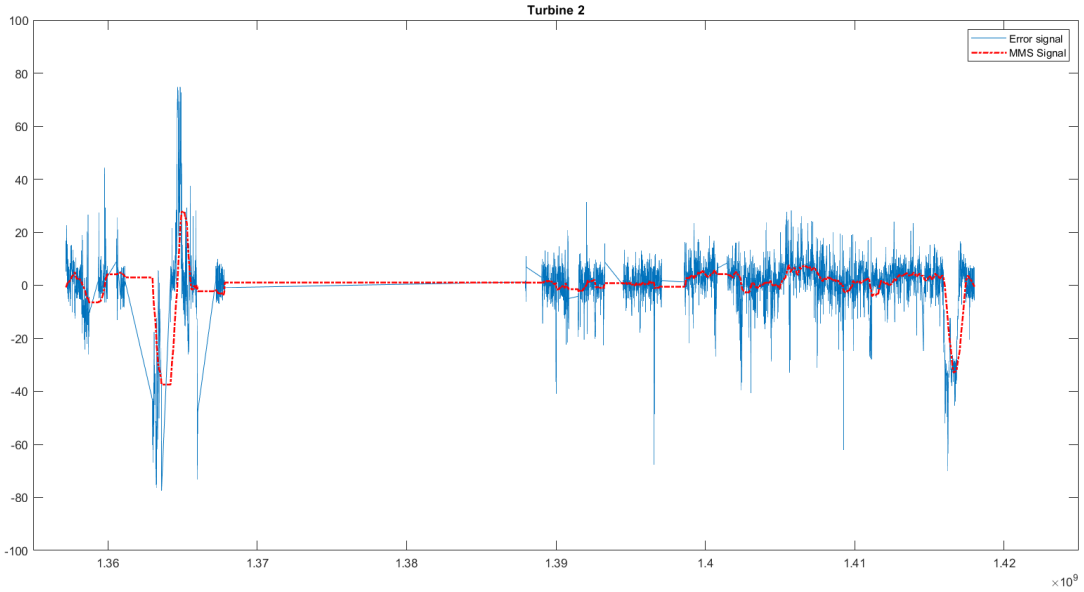


Figura 14. Comparació del senyal del model de normalitat amb el suavitzat de la MMS de la turbina 2.

Representant els valors de temperatures dels AG i comparant-los amb les dades obtingudes gràcies a les MMS (Figures 15 i 16) es pot veure com, si els AG tenen unes temperatures dintre dels rangs de comportament normal, el senyal d'error oscil·la entre valors al voltant del 0. Quan els senyals comencen a deteriorar-se i a presentar comportaments anormals els senyals dels comportaments normals suavitzats augmenten o redueixen el seu valor depenent de les anomalies.

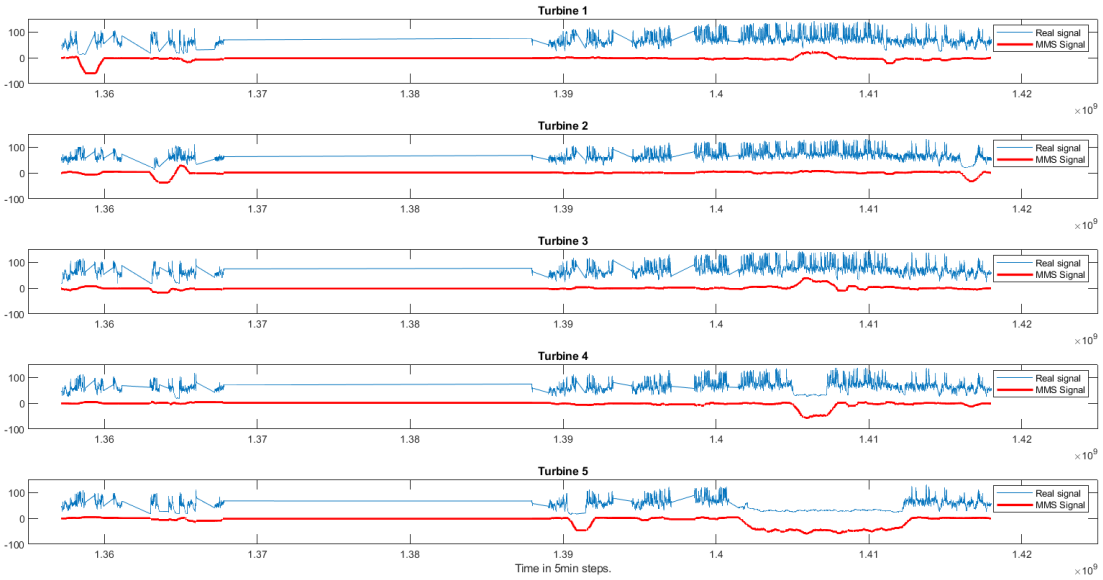


Figura 15. Comparació dels valors de temperatures dels AG amb els seus senyals d'error suavitzat.

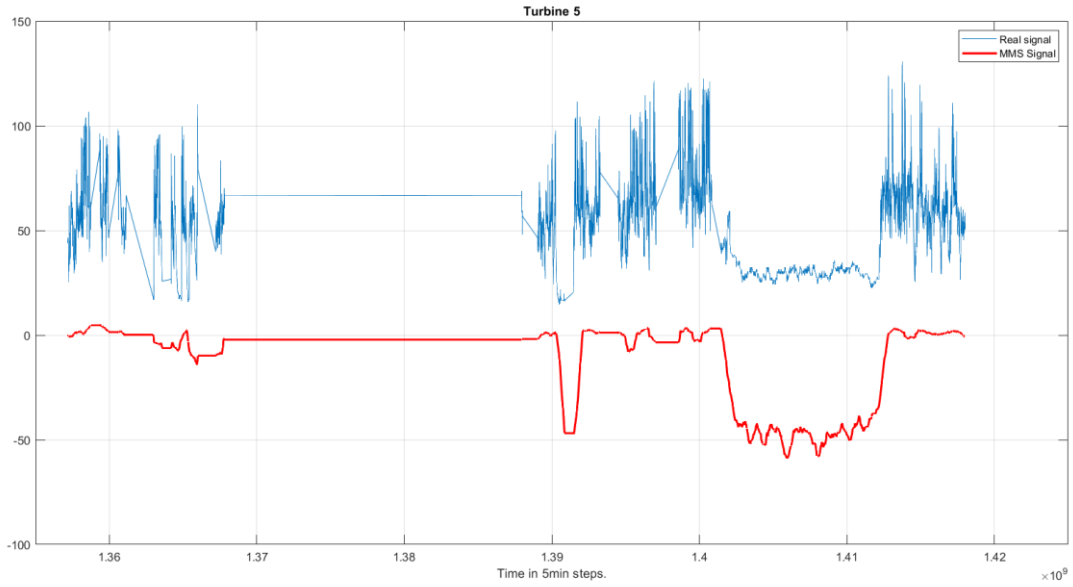


Figura 16. Comparació dels valors de temperatura de la turbina 5 amb els seu senyal d'error suavitzat.

En la regió d'error de comunicació, on no es presenten valors de dades, el valor de l'error suavitzat és representat constantment amb un 0 (franja groga) (Figura 17).

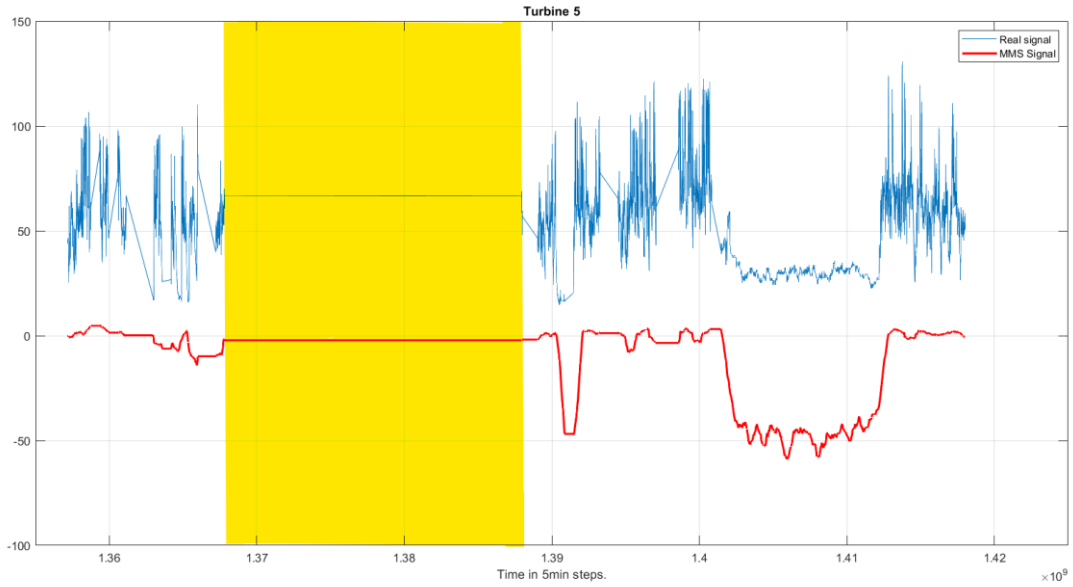


Figura 17. Zona d'error de comunicació de la turbina 5.

Aquests valors dels comportaments de normalitat suavitzats poden presentar variacions (Figures 18 i 19). Els valors corresponents a les desviacions superiors s'han representat en groc i els inferiors en taronja.

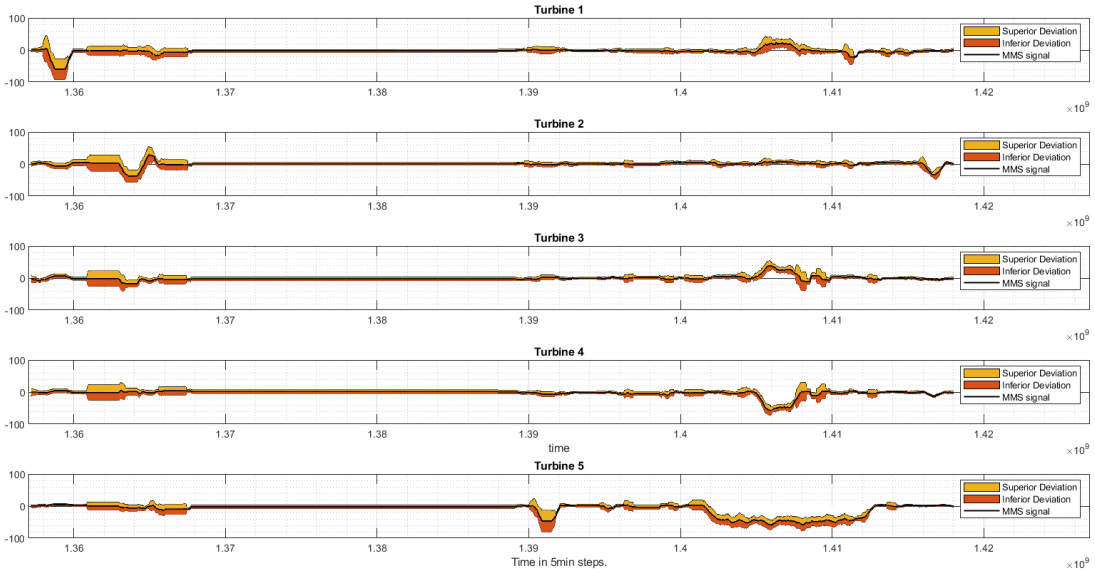


Figura 18. Representació de la distribució amb una variació de σ.

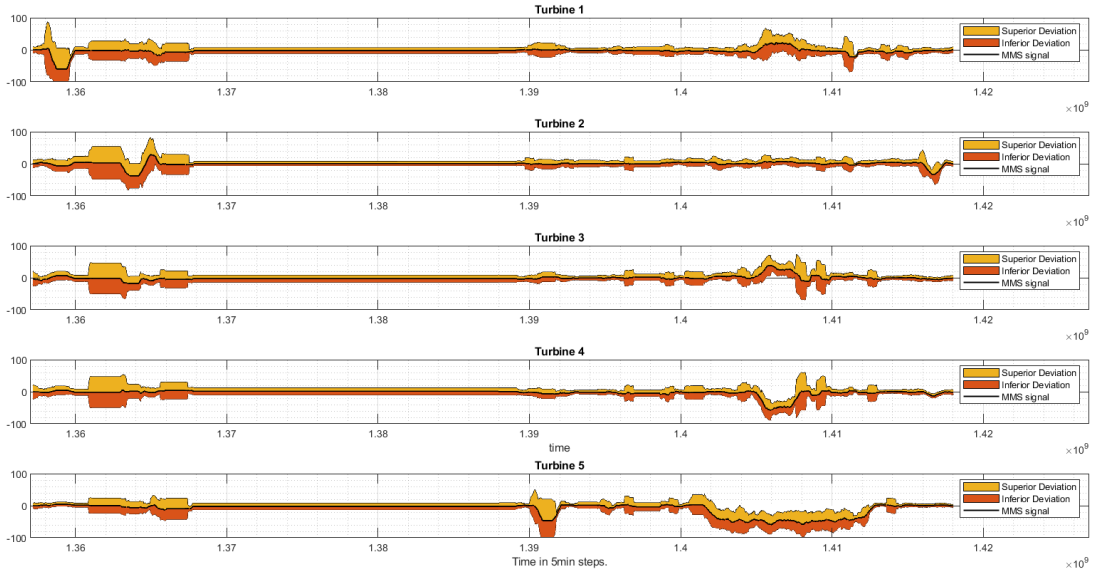


Figura 19. Representació de la distribució amb una variació de 2σ.

Com es pot observar en les figures 18 i 19 quan el comportament dels AG està dintre del rang de normalitat, i per tant presenta un error petit, la variació que pot tenir és també petita, però en les regions on els errors augmenta la volatilitat de la Desviació pateix també un gran increment fent possible afirmar que les zones d'error presenten una Desviació Estàndard inferior i superior més gran que les zones de normalitat.

Una distribució de variació σ engloba el 68,2% dels valors que pot assolir, en aquest cas, el comportament de normalitat de les SN (Figura 13) mentre que la distribució 2σ engloba el 95,6% dels valors.

El codi amb el qual s'han obtingut aquests resultats es pot trobar a l'annex 3.

5. Discussió

La finalitat d'aquest treball era estudiar la possibilitat de predir el comportament d'un AG dintre d'una població controlada. Per això s'havien plantejat els objectius de netejar les dades, trobar una xarxa neural òptima i la predicció de comportament mitjançant models de normalitat.

En l'objectiu de netejar les dades s'havia predit que existiran comportaments anòmals considerats com a errors i es podran tractar (H1)

Les dades s'han separat en dues meitats per a tractar-les més endavant seguint el que es proposa a (Martí-Puig et al. 2018). Aquest sistema permet conservar una meitat de les dades intactes, cosa molt útil per a la representació de models de normalitat, mentre que el procés més habitual, que consistiria en no netejar cap de les dades o netejar-les totes per a ser separades després internament per la SN, no tindria el contrast que permet determinar el comportament normal dels AG i comparar-lo amb el real.

En la primera meitat de les dades han aparegut comportaments anormals esperats per temperatures més baixes que el rang de comportament normal i també per algun pic elevat de temperatura exageradament alt. Al començament es va considerar buscar els pics superiors e inferiors de les temperatures amb la funció *findpeaks* de Matlab, que busca els màxims relatius d'un vector. Invertint el senyal i aplicant de nou la funció es pretenia trobar també els mínims relatius. Amb aquests mínims i màxims i calculant ambdues mitjanes es volien trobar els límits superior i inferior que delimitarien la zona de comportament normal.

El problema que tenia aquest mètode era que davant d'una zona d'error mínim molt gran, com és el cas de la gràfica 5 en la segona part de les dades, els mínims de la zona de comportament anormal guanyaven molt de pes a l'hora de fer la mitjana, degut a una menor quantitat de dades bones en proporció a les dolentes i quedaven uns valors no representatius. Tot i que només s'havia de netejar la primera part de les dades. Es va considerar que aquest mètode no era el més eficient en tots els casos i es va descartar, substituint-lo definint els límits inferiors i superiors després d'un anàlisi visual.

Tot i que es van buscar documents on es parlés de la millor forma de netejar aquestes dades, no es va trobar cap que prendre com a refèrencia. Aquest factor ha sigut un més pel qual es va decidir realitzar la inspecció visual per a delimitar les regions de comportament.

Al final els errors deguts a comportaments anormals han pogut ser netejats satisfactòriament, combinant una inspecció visual juntament amb un estudi empíric per a establir els límits, i s'ha pogut fer servir de manera correcta el senyal resultant per a l'entrenament de les dades de l'objectiu 2.

En objectiu de trobar una xarxa neural òptima es plantejava que El nombre de nòduls de la capa oculta de les SN correspondrà entre 2 i 4 (H2)**.**

Els autors de *(*Sandhya Krishnan. 2021) i (Ahmed Gad. 2018) comenten es pot trobar el nombre de nodes que hauria de tenir la capa oculta d'una xarxa neuronal seguint tres paràmetres:

1. La quantitat de nodes ocults ha de trobar-se entre el número de nodes d'*input* i *output*.
2. El nombre de nodes de la capa oculta hauria de ser 2/3 de les dimensions de la capa *input* + les dimensions la *output*
3. El nombre de nodes ocults hauria de ser menys que dues vegades les dimensions de la capa d'*input*.

Tenint en compte aquests tres punts, sent 2 el nombre de nodes d'entrada i 1 el de sortida de la SN de Matlab feta servir, els nodes interns haurien de ser 2 per al punt 1, 3, arrodonint, per al punt 2 i com a molt 4 segons el punt 3.

Aquests resultats són diferents dels obtinguts mitjançant el procés empíric d'anàlisi fet servir. A causa de les existents limitacions computacionals es va optar per a fer una exploració inicial i així veure en quina zona residia la quantitat ideal de nodes, per a aprofundir després en ella i trobar el valor òptim. Aquesta zona va resultar ser diferent de les quantitats de nodes resultants dels paràmetres plantejats a *(*Sandhya Krishnan. 2021) i (Ahmed Gad. 2018) i van ser descartades sense ser comprovades per procediment. Al final la quantitat de nodes feta servir ha sigut 6, que no coincideix amb cap dels paràmetres proposats a *(*Sandhya Krishnan. 2021) i (Ahmed Gad. 2018) però que s'ha considerat com a millor pels motius explicats anteriorment.

En l'objectiu de predir el comportament dels AG mitjançant models de normalitat amb informació creuada es plantejava que si seria possible.

Al representar les gràfiques obtingudes a partir dels models de normalitat es pot observar com es poden extreure moltes dades útils per a la predicció de comportaments. Les gràfiques d'error absolut són molt sorolloses i és necessari fer un suavitzat calculant la mitjana mòbil simple per a poder fer servir les dades de manera més eficient.

El suavitzat de la MMS s'ha fet considerant un període de set dies. Aquest suavitzat permet veure les fluctuacions de temperatura que tenen una certa durada de temps ignorant els que duren poc. Es pot debatre sobre si aquestes dimensions del període són els més apropiats ja que l'acumulació de petits errors ignorats per a aquesta manera de tractar les dades podria acabar degenerant en una fallada no detectada pel sistema. Una possible manera de corregir això seria determinar quins petits errors són considerats significatius a llarg termini i reduïr el període de la MMS perquè arribin a ser representats. A l'hora de realitzar aquest treball no s'ha considerat contabilitzar els possibles errors de curta durada sinó trobar els més significatius. Objectiu que amb aquest període s'aconsegueix correctament.

Al mirar les representacions de les MMS amb el funcionament real dels AG, es pot observar que els senyals obtinguts de les MMS segueixen el comportament dels AG de manera correcta.

Centrant-se en les gràfiques de Desviació Estàndard mòbil es pot observar les zones d'error tenen una variació més gran que les zones de normalitat. Això pot ser degut a que en una zona de normalitat tots els AG tenen un comportament pràcticament idèntic mentre que en una zona d'error, el comportament dels AG es reparteix entre anòmal i normal creant una diferència molt gran entre les dades de comportament normal i les de comportament anormal que es veuen reflectides finalment en aquestes representacions.

Amb les gràfiques de Desviació Estàndard mòbil es poden generar senyals d'alarma d'una manera similar a com es proposa a (Marti-Puig et al. 2021) delimitant uns llindars d'error. Quan la variació del senyal de les MMS superi aquests llindars l'alarma saltaria i avisaria d'un error en el sistema.

6. Conclusió

L'estudi realitzat sobre la predicció de comportament d'aerogeneradors mitjançant informació creuada permet concloure que els senyals de temperatura dels AG presenten rangs de temperatura associats a un comportament anormal que s'han d'eliminar establint uns llindars mínims i màxims per a poder extreure les dades de comportament normal.

Després d'analitzar els errors quadrats de les SN amb diferents quantitats de nodes a la capa interna es conclou que la millor quantitat de nodes ocults que pot tenir una SN per a fer aquest treball és 6.

Estudiant les diferents gràfiques d'error extretes mitjançant els models de normalitat es pot afirmar que és possible predir el comportament dels AG mitjançant informació creuada dels AG veïns.

6.1. Limitacions i millores a realitzar en projectes futurs

Una de les limitacions més grans d'aquest projecte era la falta de recursos computacionals de la que es disposava. El processament de les xarxes neuronals trigava diverses hores en l'ordinador de sobretaula i dies en el portàtil. Per a cada xarxa, una iteració d'entrenament i test de les trenta que s'havien de fer, resultava, aproximadament, en uns 30 minuts de processament pel portàtil. Tenint en compte que s'han analitzat vuit xarxes diferents (5, 6, 7, 8, 9, 10, 15 i 20) i que cada una trigava unes 15 hores a processar-se, el temps de processament total en el portàtil hauria sigut de 120 hores. Amb l'ordinador de sobretaula, tot aquest procés es reduïa a 12 hores seguides de processament. Això sense comptar el procediment d'aprenentatge per a fer servir les SN.

Per culpa d'aquestes limitacions de temps es va optar per a fer una exploració inicial fent servir 20 com a valor màxim i després centrar-se a aprofundir entre els resultats 5 i 10 ignorant totes les altres quantitats de nodes possibles. Havent-se disposat d'un equip de processament més potent s'haurien pogut analitzar totes les configuracions de quantitats de nodes individualment.

Com a punt a millorar en el futur analitzar com a mínim les configuracions de quantitats de nodes 2, 3 i 4, ja que aquestes són les que s'havien plantejat en la hipòtesi com a millors, però que s'han hagut de descartar després de l'exploració inicial.

Tot i que es va rebutjar aquest procés per a netejar les dades a la hipòtesi 1, podria ser bona idea inspeccionar amb més detall la proposta de la funció *findingpeaks* si per a futurs projectes s'ha d'automatitzar el procés netejar dades a partir de la temperatura i/o si cal regular les dades en funció de variacions de temperatura, ja que si la temperatura del sistema general augmenta o disminueix sistemàticament, els llindars establerts manualment podrien no ser els adequats.

De cara a la representació d'errors es podria explorar la possibilitat d'afegir senyals d'alarma a partir de les gràfiques de la Desviació Estàndard mòbil. La implementació d'un codi que permeti trobar en quin moment la desviació comença a créixer o que incrementa uns certs percentatges respecte a les mesures anteriors pot resultar molt útil per ajudar a definir quan un error cal ser revisat perquè comença a ser perillós, implementant diferents rangs d'alarma per a diferents graus d'error i ajudar a planificar el millor moment per parar l'AG durant un temps més llarg per a un manteniment a més gran escala.

7. Bibliografia

Appavou, Fabiani, Brown, Adam, E, B., 2016. Renewables 2016 Global Status Report. Tech. rep., Renewable energy policy network for the 21ST century .

EUROSTAT. Energy Balance Sheets 2016 Data. 2018

Niklas Hellberg. (2019) Crece la generación de electricidad con fuentes renovables en 2019 (unep.org)

David Jones. (2020) La Energía Verde en Europa Supera por primera vez a los combustibles fósiles (avatarenergia.com). Ember

SAWIN, Janet L., et al. Renewables 2017 global status report. 2016.

Marti-Puig, P.; Cusidó, J.; Lozano, F.J.; Serra-Serra, M.; Caiafa, C.F.; Solé-Casals, J. Detection of Wind Turbine Failures through Cross-Information between Neighbouring Turbines. Appl. Sci. 2022, 12, 9491. https://doi.org/ 10.3390/app12199491

Hongzhou Wang, A survey of maintenance policies of deteriorating systems, European Journal of Operational Research, Volume 139, Issue 3, 2002, Pages 469-489 https://doi.org/10.1016/S0377-2217(01)00197-7

Jesús María Pinar Pérez, Fausto Pedro García Márquez, Andrew Tobias, Mayorkinos Papaelias, Wind turbine reliability analysis, Renewable and Sustainable Energy Reviews, Volume 23, 2013. https://doi.org/10.1016/j.rser.2013.03.018.

Zhang, P.; Lu, D. A survey of condition monitoring and fault diagnosis toward integrated O&M for wind turbines. Energies 2019, 12, 2801. https://doi.org/10.3390/en12142801

Sandhya Krishnan. (2021) How to determine the number of layers and neurons in the hidden layer? | by Sandhya Krishnan | Geek Culture | Medium | Geek Culture

Ahmed Gad. (2018) https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e

Iberdrola (2023) Mantenimiento Predictivo: Qué es, características y ejemplos - Iberdrola

Marti-Puig, P.; Blanco-M., A.; Serra-Serra, M.; Solé-Casals, J. Wind Turbine Prognosis Models Based on SCADA Data and Extreme Learning Machines. Appl. Sci. 2021, 11, 590. http://doi.org/ 10.3390/app11020590

Julio (2018) ¿Qué es la Energía Distribuida? - Conciencia Eco

Rita. (2022) Deep Neural Networks Vs Shallower Neural Networks: Advantages And Disadvantages – Surfactants

IBM (2021) Conceptos básicos (redes neuronales) - Documentación de IBM

Anònim (2022) What is Overfitting? - Definition from Techopedia

Jason Fernando, Chip Stapleton, Katrina Munichiello. (2022) Moving Average (MA): Purpose, Uses, Formula, and Examples (investopedia.com)

OrvizarTV. (2021) LEVENBERG MARQUARDT | Optimización multidimensional - YouTube

Pere Marti-Puig, Alejandro Blanco-M, Juan José Cárdenas, Jordi Cusidó, Jordi Solé-Casals, Effects of the pre-processing algorithms in fault diagnosis of wind turbines, Environmental Modelling & Software, Volume 110, 2018, Pages 119-128, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2018.05.002.

Anònim (2023) Moving Standard Deviation | Zaner: Commodities, Futures, Forex and Cash Metals Brokers

Annex A

En aquest annex es pot trobar un exemple del codi escrit en Matlab fet servir per a la representació i neteja de les dades de temperatura dels generadors, fase A inicials.

```
load WGEN.mat


% Guardar los valores en variables
m1_raw = WGEN(:, 1, 1);
m2_raw = WGEN(:, 1, 2);
m3_raw = WGEN(:, 1, 3);
m4_raw = WGEN(:, 1, 4);
m5_raw = WGEN(:, 1, 5);


%% Dividir valores en dos mitades
% valores mitad inicial
m1_medio = m1_raw(1: length(m1_raw)/2);
m2_medio = m2_raw(1: length(m2_raw)/2);
m3_medio = m3_raw(1: length(m3_raw)/2);
m4_medio = m4_raw(1: length(m4_raw)/2);
m5_medio = m5_raw(1: length(m5_raw)/2);
t_medio = t_(1: length(t_)/2);


% Valores de mitad a final


m1_medio2 = m1_raw(length(m1_raw)/2: length(m1_raw));
m2_medio2 = m2_raw(length(m2_raw)/2: length(m2_raw));
m3_medio2 = m3_raw(length(m3_raw)/2: length(m3_raw));
m4_medio2 = m4_raw(length(m4_raw)/2: length(m4_raw));
m5_medio2 = m5_raw(length(m5_raw)/2: length(m5_raw));
t_medio2 = t_(length(t_)/2: length(t_));


% crear matrices cero de dimensiones maquinas
m_zero = zeros(size(m1_medio));


m_zero1 = m_zero;
m_zero2 = m_zero;
m_zero3 = m_zero;
m_zero4 = m_zero;
m_zero5 = m_zero;


% matriz ceros resultado
m_zerot_result = m_zero;
```

```
% buscar indices con valores t<40


t = 35;
t2 = 150
row = find(m1_medio < t | m1_medio > t2);
row2 = find(m2_medio < t | m2_medio > t2);
row3 = find(m3_medio < t | m3_medio > t2);
row4 = find(m4_medio < t | m4_medio > t2);
row5 = find(m5_medio < t | m5_medio > t2);


% reemplazar valor indices en matrices 0 para comparar


m_zero1(row,:) = 1;
m_zero2(row2,:) = 1;
m_zero3(row3,:) = 1;
m_zero4(row4,:) = 1;
m_zero5(row5,:) = 1;


% recorrer matrices para ver cuando valores indices coinciden
% y cambiar valor indices por 0.


for i = 1:length(m_zero)
    if (m_zero1(i) == 1) && (m_zero2(i) == 1) && (m_zero3(i) == 1) &&
(m_zero4(i) == 1) && (m_zero5(i) == 1)


        m_zero1(i,:) = 0;
        m_zero2(i,:) = 0;
        m_zero3(i,:) = 0;
        m_zero4(i,:) = 0;
        m_zero5(i,:) = 0;


    end
end


% juntar todos los valores en una matriz 0 nueva


for i = 1:length(m_zero)
    if (m_zero1(i) == 1) || (m_zero2(i) == 1) || (m_zero3(i) == 1) ||
(m_zero4(i) == 1) || (m_zero5(i) == 1)

        m_zerot_result(i,:) = 1; % matriz de 0s
    end
end
```

```
valores = find(m_zerot_result == 0);


% prueba 2 --> Eliminar ceros


m_tt1 = m1_medio(valores);
m_tt2 = m2_medio(valores);
m_tt3 = m3_medio(valores);
m_tt4 = m4_medio(valores);
m_tt5 = m5_medio(valores);
m_time = t_(valores);


% Representar sin 0


figure;
subplot(5, 1, 1)
plot(m_time, m_tt1)
title("Turbine 1 filtered data")
ylabel("Temperature in ºC")
subplot(5, 1, 2)
plot(m_time, m_tt2)
title("Turbine 2 filtered data")
ylabel("Temperature in ºC")
subplot(5, 1, 3)
plot(m_time, m_tt3)
title("Turbine 3 filtered data")
ylabel("Temperature in ºC")
subplot(5, 1, 4)
plot(m_time, m_tt4)
title("Turbine 4 filtered data")
ylabel("Temperature in ºC")
subplot(5, 1, 5)
plot(m_time, m_tt5)
title("Turbine 5 filtered data")
xlabel("Time in 5min steps")
ylabel("Temperature in ºC")


% figure;
figure;
subplot(6, 1, 1)
plot(t_medio, m1_medio)
title("Turbine 1")
ylabel("Temperature in ºC")
axis([1.325*10^9 1.36*10^9 0 150])


subplot(6, 1, 2)
plot(t_medio, m2_medio)
```

```
title("Turbine 2")
ylabel("Temperature in ºC")
axis([1.325*10^9 1.36*10^9 0 150])


subplot(6, 1, 3)
plot(t_medio, m3_medio)
title("Turbine 3")
ylabel("Temperature in ºC")
axis([1.325*10^9 1.36*10^9 0 150])


subplot(6, 1, 4)
plot(t_medio, m4_medio)
title("Turbine 4")
ylabel("Temperature in ºC")
axis([1.325*10^9 1.36*10^9 0 150])


subplot(6, 1, 5)
plot(t_medio, m5_medio)
title("Turbine 5")
ylabel("Temperature in ºC")
axis([1.325*10^9 1.36*10^9 0 150])


subplot(6, 1, 6)
baseLine = 0.0;
bar(t_medio, m_zerot_result, 1.5, "r", 'EdgeColor',"none")
% hold on
% fill(t_medio, m_zerot_result, 'r')
title("Error regions")
xlabel("Time in 5min steps")
axis([1.325*10^9 1.36*10^9 0 2])
```

Annex B

En aquest annex es pot trobar un exemple del codi escrit en Matlab fet servir per a la creació, entrenament i test de les SN. En aquest cas amb configuració de nodes 5, 10, 15 i 20.

Creació de les SN.

```
m_tt1 = m1_medio(valores);
m_tt2 = m2_medio(valores);
m_tt3 = m3_medio(valores);
m_tt4 = m4_medio(valores);
m_tt5 = m5_medio(valores);
% Call the Shallow network


% Definimos los parámetros que usar en la red


x1 = [m_tt1 m_tt2 m_tt3 m_tt4]; % primera configuració de inputs
x2 = [m_tt1 m_tt2 m_tt3 m_tt5]; % segona configuració de inputs
x3 = [m_tt1 m_tt2 m_tt4 m_tt5]; % tercera
x4 = [m_tt1 m_tt3 m_tt4 m_tt5]; % quarta
x5 = [m_tt2 m_tt3 m_tt4 m_tt5]; % cinquena


t1 = m_tt5; % primera configuració de target
t2 = m_tt4; % segona
t3 = m_tt3; % tercera
t4 = m_tt2; % quarta
t5 = m_tt1; % cinquena


m1 = [m1_medio2 m2_medio2 m3_medio2 m4_medio2]; % primera configuració de
inputs
m2 = [m1_medio2 m2_medio2 m3_medio2 m5_medio2]; % segona configuració de
inputs
m3 = [m1_medio2 m2_medio2 m4_medio2 m5_medio2]; % tercera
m4 = [m1_medio2 m3_medio2 m4_medio2 m5_medio2]; % quarta
m5 = [m2_medio2 m3_medio2 m4_medio2 m5_medio2]; % cinquena


mt1 = m5_medio2; % primera configuració de target
mt2 = m4_medio2; % segona
mt3 = m3_medio2; % tercera
mt4 = m2_medio2; % quarta
mt5 = m1_medio2; % cinquena


% transponer valores
x1_t = x1';
x2_t = x2';
x3_t = x3';
```

```
x4_t = x4';
x5_t = x5';


t1_t = t1';
t2_t = t2';
t3_t = t3';
t4_t = t4';
t5_t = t5';


m1_t = m1';
m2_t = m2';
m3_t = m3';
m4_t = m4';
m5_t = m5';

% CREATING THE EMPTY LISTS
% LISTS 5
N=30;


Data51=cell(N,1);
Data51_test=cell(N,1);


Data52=cell(N,1);
Data52_test=cell(N,1);


Data5=cell(N,1);
Data5_test=cell(N,1);


Data54=cell(N,1);
Data54_test=cell(N,1);


Data55=cell(N,1);
Data55_test=cell(N,1);


% LISTS 10


Data101=cell(N,1);
Data101_test=cell(N,1);


Data102=cell(N,1);
Data102_test=cell(N,1);
```

```
Data10=cell(N,1);
Data10_test=cell(N,1);


Data104=cell(N,1);
Data104_test=cell(N,1);


Data105=cell(N,1);
Data105_test=cell(N,1);


% LISTS 15


Data151=cell(N,1);
Data151_test=cell(N,1);


Data152=cell(N,1);
Data152_test=cell(N,1);


Data15=cell(N,1);
Data15_test=cell(N,1);


Data154=cell(N,1);
Data154_test=cell(N,1);


Data155=cell(N,1);
Data155_test=cell(N,1);


% LISTS 20


Data201=cell(N,1);
Data201_test=cell(N,1);


Data202=cell(N,1);
Data202_test=cell(N,1);


Data20=cell(N,1);
Data20_test=cell(N,1);


Data204=cell(N,1);
Data204_test=cell(N,1);
```

```
Data205=cell(N,1);
Data205_test=cell(N,1);


% CREATING THE NETWORKS


net_51 = fitnet(5);
net_101 = fitnet(10);
net_151 = fitnet(15);
net_201 = fitnet(20);


net_52 = fitnet(5);
net_102 = fitnet(10);
net_152 = fitnet(15);
net_202 = fitnet(20);


net_5 = fitnet(5);
net_10 = fitnet(10);
net_15 = fitnet(15);
net_20 = fitnet(20);


net_54 = fitnet(5);
net_104 = fitnet(10);
net_154 = fitnet(15);
net_204 = fitnet(20);


net_55 = fitnet(5);
net_105 = fitnet(10);
net_155 = fitnet(15);
net_205 = fitnet(20);
```

Training the networks

M1

NETWORK 5 NODES

```
for i=1:N

    net51 = train(net_51, x1_t, t1_t);


    % network parameters, all data is being used to train the network

    net51.divideParam.trainRatio = 100/100;

    net51.divideParam.valRatio = 0/100;
```

```
net51.divideParam.testRatio = 0/100;


y51 = net51(x1_t);


% NETWORK 10 NODES
net101 = train(net_101, x1_t, t1_t);


% network parameters, all data is being used to train the network
net101.divideParam.trainRatio = 100/100;
net101.divideParam.valRatio = 0/100;
net101.divideParam.testRatio = 0/100;


y101 = net101(x1_t);
% NETWORK 15 NODES
net151 = train(net_151, x1_t, t1_t);


% network parameters, all data is being used to train the network
net151.divideParam.trainRatio = 100/100;
net151.divideParam.valRatio = 0/100;
net151.divideParam.testRatio = 0/100;


y151 = net151(x1_t);
%NETWORK 20 NODES
net201 = train(net_201, x1_t, t1_t);


% network parameters, all data is being used to train the network
net201.divideParam.trainRatio = 100/100;
net201.divideParam.valRatio = 0/100;
net201.divideParam.testRatio = 0/100;
```

```
y201 = net201(x1_t);

%M2 NETWORK 5 NODES
net52 = train(net_52, x2_t, t2_t);

% network parameters, all data is being used to train the network
net52.divideParam.trainRatio = 100/100;
net52.divideParam.valRatio = 0/100;
net52.divideParam.testRatio = 0/100;

y52 = net52(x2_t);
% NETWORK 10 NODES
net102 = train(net_102, x2_t, t2_t);

% network parameters, all data is being used to train the network
net102.divideParam.trainRatio = 100/100;
net102.divideParam.valRatio = 0/100;
net102.divideParam.testRatio = 0/100;

y102 = net102(x2_t);
% NETWORK 15 NODES
net152 = train(net_152, x2_t, t2_t);

% network parameters, all data is being used to train the network
net152.divideParam.trainRatio = 100/100;
net152.divideParam.valRatio = 0/100;
net152.divideParam.testRatio = 0/100;

y152 = net152(x2_t);
%NETWORK 20 NODES
```

```
net202 = train(net_202, x2_t, t2_t);

% network parameters, all data is being used to train the network
net202.divideParam.trainRatio = 100/100;
net202.divideParam.valRatio = 0/100;
net202.divideParam.testRatio = 0/100;

y202 = net202(x2_t);
%M4 NETWORK 5 NODES
net54 = train(net_54, x4_t, t4_t);

% network parameters, all data is being used to train the network
net54.divideParam.trainRatio = 100/100;
net54.divideParam.valRatio = 0/100;
net54.divideParam.testRatio = 0/100;

y54 = net54(x4_t);
%NETWORK 10 NODES
net104 = train(net_104, x4_t, t4_t);

% network parameters, all data is being used to train the network
net104.divideParam.trainRatio = 100/100;
net104.divideParam.valRatio = 0/100;
net104.divideParam.testRatio = 0/100;

y104 = net104(x4_t);
% NETWORK 15 NODES
net154 = train(net_154, x4_t, t4_t);

% network parameters, all data is being used to train the network
```

```
net154.divideParam.trainRatio = 100/100;
net154.divideParam.valRatio = 0/100;
net154.divideParam.testRatio = 0/100;

y154 = net154(x4_t);

%NETWORK 20 NODES
net204 = train(net_204, x4_t, t4_t);

% network parameters, all data is being used to train the network
net204.divideParam.trainRatio = 100/100;
net204.divideParam.valRatio = 0/100;
net204.divideParam.testRatio = 0/100;

y204 = net204(x4_t);
%M5 NETWORK 5 NODES
net55 = train(net_55, x5_t, t5_t);

% network parameters, all data is being used to train the network
net55.divideParam.trainRatio = 100/100;
net55.divideParam.valRatio = 0/100;
net55.divideParam.testRatio = 0/100;

y55 = net55(x5_t);
% NETWORK 10 NODES
net105 = train(net_105, x5_t, t5_t);

% network parameters, all data is being used to train the network
net105.divideParam.trainRatio = 100/100;
net105.divideParam.valRatio = 0/100;
```

```
net105.divideParam.testRatio = 0/100;


y105 = net105(x5_t);
% NETWORK 15 NODES
net155 = train(net_155, x5_t, t5_t);

% network parameters, all data is being used to train the network
net155.divideParam.trainRatio = 100/100;
net155.divideParam.valRatio = 0/100;
net155.divideParam.testRatio = 0/100;

y155 = net155(x5_t);
% NETWORK 20 NODES
net205 = train(net_205, x5_t, t5_t);

% network parameters, all data is being used to train the network
net205.divideParam.trainRatio = 100/100;
net205.divideParam.valRatio = 0/100;
net205.divideParam.testRatio = 0/100;

y205 = net205(x5_t); % ERROR FINDING M1
error_51 = y51' - t1;
error_101 = y101' - t1;
error_151 = y151' - t1;
error_201 = y201' - t1;

% Get the error.
r_error51 = 1/length(error_51) * error_51';
result51 = dot(r_error51, error_51);
```

```
r_error101 = 1/length(error_101) * error_101';
result101 = dot(r_error101, error_101);

r_error151 = 1/length(error_151) * error_151';
result151 = dot(r_error151, error_151);

r_error201 = 1/length(error_201) * error_201';
result201 = dot(r_error201, error_201);

Data51{contador} = result51;
Data101{contador} = result101;
Data151{contador} = result151;
Data201{contador} = result201;

ym51 = net51(m1_t);
ym101 = net101(m1_t);
ym151 = net151(m1_t);
ym201 = net201(m1_t);

error_5t1 = ym51' - mt1;
error_10t1 = ym101' - mt1;
error_15t1 = ym151' - mt1;
error_20t1 = ym201' - mt1;

r_error5t1 = 1/length(error_5t1) * error_5t1';
result5t1 = dot(r_error5t1, error_5t1);

r_error10t1 = 1/length(error_10t1) * error_10t1';
result10t1 = dot(r_error10t1, error_10t1);
```

```
r_error15t1 = 1/length(error_15t1) * error_15t1';
result15t1 = dot(r_error15t1, error_15t1);

r_error20t1 = 1/length(error_20t1) * error_20t1';
result20t1 = dot(r_error20t1, error_20t1);

Data51_test{contador} = result5t1;
Data101_test{contador} = result10t1;
Data151_test{contador} = result15t1;
Data201_test{contador} = result20t1; %M2
error_52 = y52' - t2;
error_102 = y102' - t2;
error_152 = y152' - t2;
error_202 = y202' - t2;

% Get the error.
r_error52 = 1/length(error_52) * error_52';
result52 = dot(r_error52, error_52);

r_error102 = 1/length(error_102) * error_102';
result102 = dot(r_error102, error_102);

r_error152 = 1/length(error_152) * error_152';
result152 = dot(r_error152, error_152);

r_error202 = 1/length(error_202) * error_202';
result202 = dot(r_error202, error_202);

Data52{contador} = result52;
Data102{contador} = result102;
```

```
Data152{contador} = result152;
Data202{contador} = result202;


ym52 = net52(m2_t);
ym102 = net102(m2_t);
ym152 = net152(m2_t);
ym202 = net202(m2_t);


error_5t2 = ym52' - mt2;
error_10t2 = ym102' - mt2;
error_15t2 = ym152' - mt2;
error_20t2 = ym202' - mt2;


r_error5t2 = 1/length(error_5t2) * error_5t2';
result5t2 = dot(r_error5t2, error_5t2);


r_error10t2 = 1/length(error_10t2) * error_10t2';
result10t2 = dot(r_error10t2, error_10t2);


r_error15t2 = 1/length(error_15t2) * error_15t2';
result15t2 = dot(r_error15t2, error_15t2);


r_error20t2 = 1/length(error_20t2) * error_20t2';
result20t2 = dot(r_error20t2, error_20t2);


Data52_test{contador} = result5t2;
Data102_test{contador} = result10t2;
Data152_test{contador} = result15t2;


%M3
```

```
net5 = train(net_5, x3_t, t3_t);

% network parameters, all data is being used to train the network
net5.divideParam.trainRatio = 100/100;
net5.divideParam.valRatio = 0/100;
net5.divideParam.testRatio = 0/100;

y5 = net5(x3_t);
%NETWORK 10 NODES
net10 = train(net_10, x3_t, t3_t);

% network parameters, all data is being used to train the network
net10.divideParam.trainRatio = 100/100;
net10.divideParam.valRatio = 0/100;
net10.divideParam.testRatio = 0/100;

y10 = net10(x3_t);
% NETWORK 15 NODES
net15 = train(net_15, x3_t, t3_t);

% network parameters, all data is being used to train the network
net15.divideParam.trainRatio = 100/100;
net15.divideParam.valRatio = 0/100;
net15.divideParam.testRatio = 0/100;

y15 = net15(x3_t);
% NETWORK 20 NODES
net20 = train(net_20, x3_t, t3_t);

% network parameters, all data is being used to train the network
```

```
net20.divideParam.trainRatio = 100/100;
net20.divideParam.valRatio = 0/100;
net20.divideParam.testRatio = 0/100;

y20 = net20(x3_t);

error_5 = y5' - t3;
error_10 = y10' - t3;
error_15 = y15' - t3;
error_20 = y20' - t3;

% Get the error.
r_error5 = 1/length(error_5) * error_5';
result5 = dot(r_error5, error_5);

r_error10 = 1/length(error_10) * error_10';
result10 = dot(r_error10, error_10);

r_error15 = 1/length(error_15) * error_15';
result15 = dot(r_error15, error_15);

r_error20 = 1/length(error_20) * error_20';
result20 = dot(r_error20, error_20);

Data5{contador} = result5;
Data10{contador} = result10;
Data15{contador} = result15;
Data20{contador} = result20;

ym5 = net5(m3_t);
```

```
ym10 = net10(m3_t);
ym15 = net15(m3_t);
ym20 = net20(m3_t);

error_5t = ym5' - mt3;
error_10t = ym10' - mt3;
error_15t = ym15' - mt3;
error_20t = ym20' - mt3;

r_error5t = 1/length(error_5t) * error_5t';
result5t = dot(r_error5t, error_5t);

r_error10t = 1/length(error_10t) * error_10t';
result10t = dot(r_error10t, error_10t);

r_error15t = 1/length(error_15t) * error_15t';
result15t = dot(r_error15t, error_15t);

r_error20t = 1/length(error_20t) * error_20t';
result20t = dot(r_error20t, error_20t);

Data5_test{contador} = result5t;
Data10_test{contador} = result10t;
Data15_test{contador} = result15t;
Data20_test{contador} = result20t;
Data202_test{contador} = result20t2; %M4
error_54 = y54' - t4;
error_104 = y104' - t4;
error_154 = y154' - t4;
error_204 = y204' - t4;
```

```
% Get the error.
r_error54 = 1/length(error_54) * error_54';
result54 = dot(r_error54, error_54);

r_error104 = 1/length(error_104) * error_104';
result104 = dot(r_error104, error_104);

r_error154 = 1/length(error_154) * error_154';
result154 = dot(r_error154, error_154);

r_error204 = 1/length(error_204) * error_204';
result204 = dot(r_error204, error_204);

Data54{contador} = result54;
Data104{contador} = result104;
Data154{contador} = result154;
Data204{contador} = result204;

ym54 = net54(m4_t);
ym104 = net104(m4_t);
ym154 = net154(m4_t);
ym204 = net204(m4_t);

error_5t4 = ym54' - mt4;
error_10t4 = ym104' - mt4;
error_15t4 = ym154' - mt4;
error_20t4 = ym204' - mt4;

r_error5t4 = 1/length(error_5t4) * error_5t4';
```

```
result5t4 = dot(r_error5t4, error_5t4);


r_error10t4 = 1/length(error_10t4) * error_10t4';
result10t4 = dot(r_error10t4, error_10t4);


r_error15t4 = 1/length(error_15t4) * error_15t4';
result15t4 = dot(r_error15t4, error_15t4);


r_error20t4 = 1/length(error_20t4) * error_20t4';
result20t4 = dot(r_error20t4, error_20t4);


Data54_test{contador} = result5t4;
Data104_test{contador} = result10t4;
Data154_test{contador} = result15t4;
Data204_test{contador} = result20t4; %M5
error_55 = y55' - t5;
error_105 = y105' - t5;
error_155 = y155' - t5;
error_205 = y205' - t5;


% Get the error.
r_error55 = 1/length(error_55) * error_55';
result55 = dot(r_error55, error_55);


r_error105 = 1/length(error_105) * error_105';
result105 = dot(r_error105, error_105);


r_error155 = 1/length(error_155) * error_155';
result155 = dot(r_error155, error_155);
```

```
r_error205 = 1/length(error_205) * error_205';
result205 = dot(r_error205, error_205);

Data55{contador} = result55;
Data105{contador} = result105;
Data155{contador} = result155;
Data205{contador} = result205;

ym55 = net55(m5_t);
ym105 = net105(m5_t);
ym155 = net155(m5_t);
ym205 = net205(m5_t);

error_5t5 = ym55' - mt5;
error_10t5 = ym105' - mt5;
error_15t5 = ym155' - mt5;
error_20t5 = ym205' - mt5;

r_error5t5 = 1/length(error_5t5) * error_5t5';
result5t5 = dot(r_error5t5, error_5t5);

r_error10t5 = 1/length(error_10t5) * error_10t5';
result10t5 = dot(r_error10t5, error_10t5);

r_error15t5 = 1/length(error_15t5) * error_15t5';
result15t5 = dot(r_error15t5, error_15t5);

r_error20t5 = 1/length(error_20t5) * error_20t5';
result20t5 = dot(r_error20t5, error_20t5);
```

```
    Data55_test{contador} = result5t5;

    Data105_test{contador} = result10t5;

    Data155_test{contador} = result15t5;

    Data205_test{contador} = result20t5;

    contador = contador + 1;
end
```

Annex C

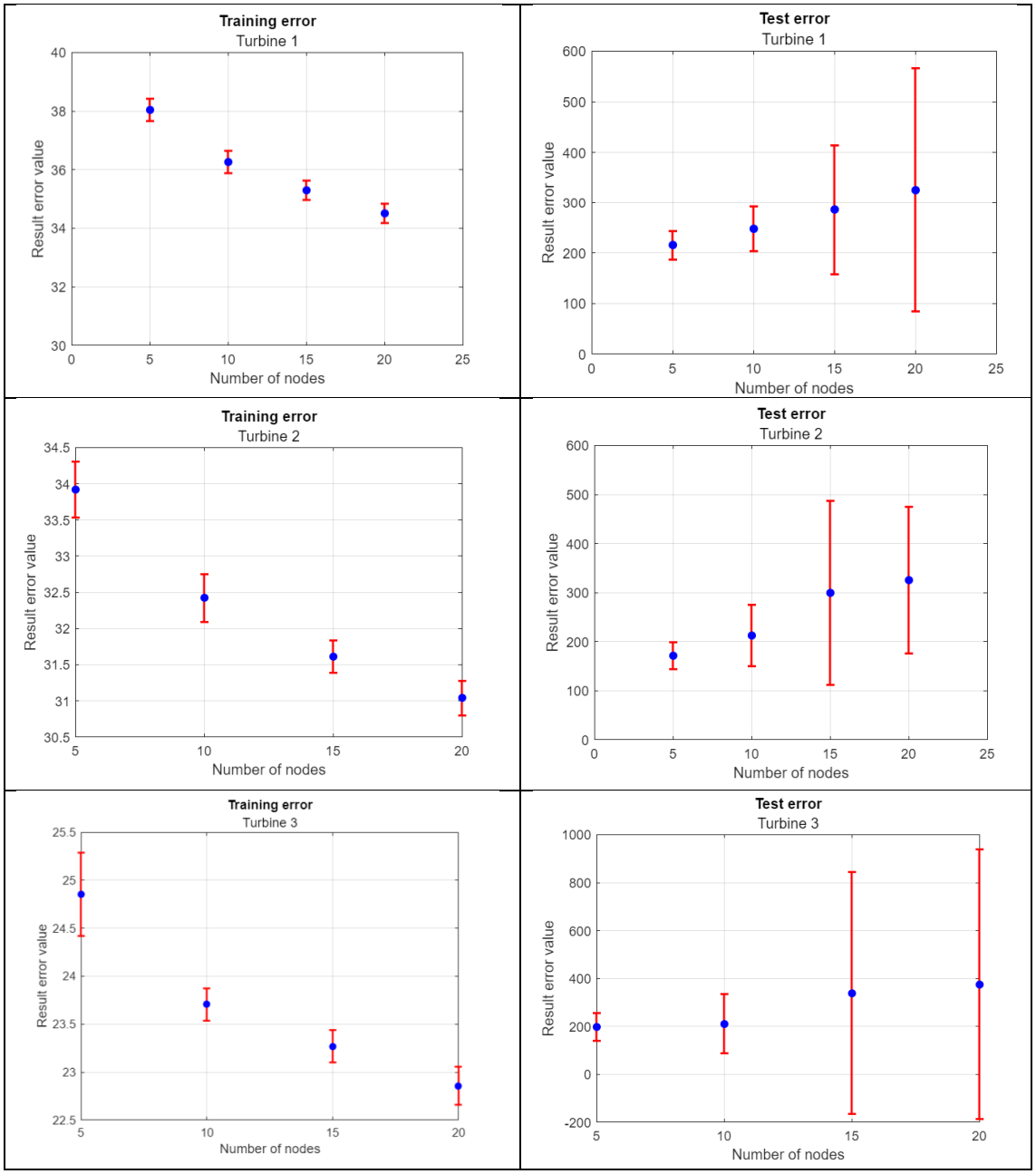
En aquest annex es pot trobar un exemple del codi escrit en Matlab fet servir per a la representació dels *box-plot* de les Mitjanes d'error amb les seves Desviacions Standard corresponents a les dades de la turbina 1. El procediment per a la representació dels *box-plot* de les Mitjanes d'error amb les seves Desviacions Standard de la resta de turbines és el mateix.

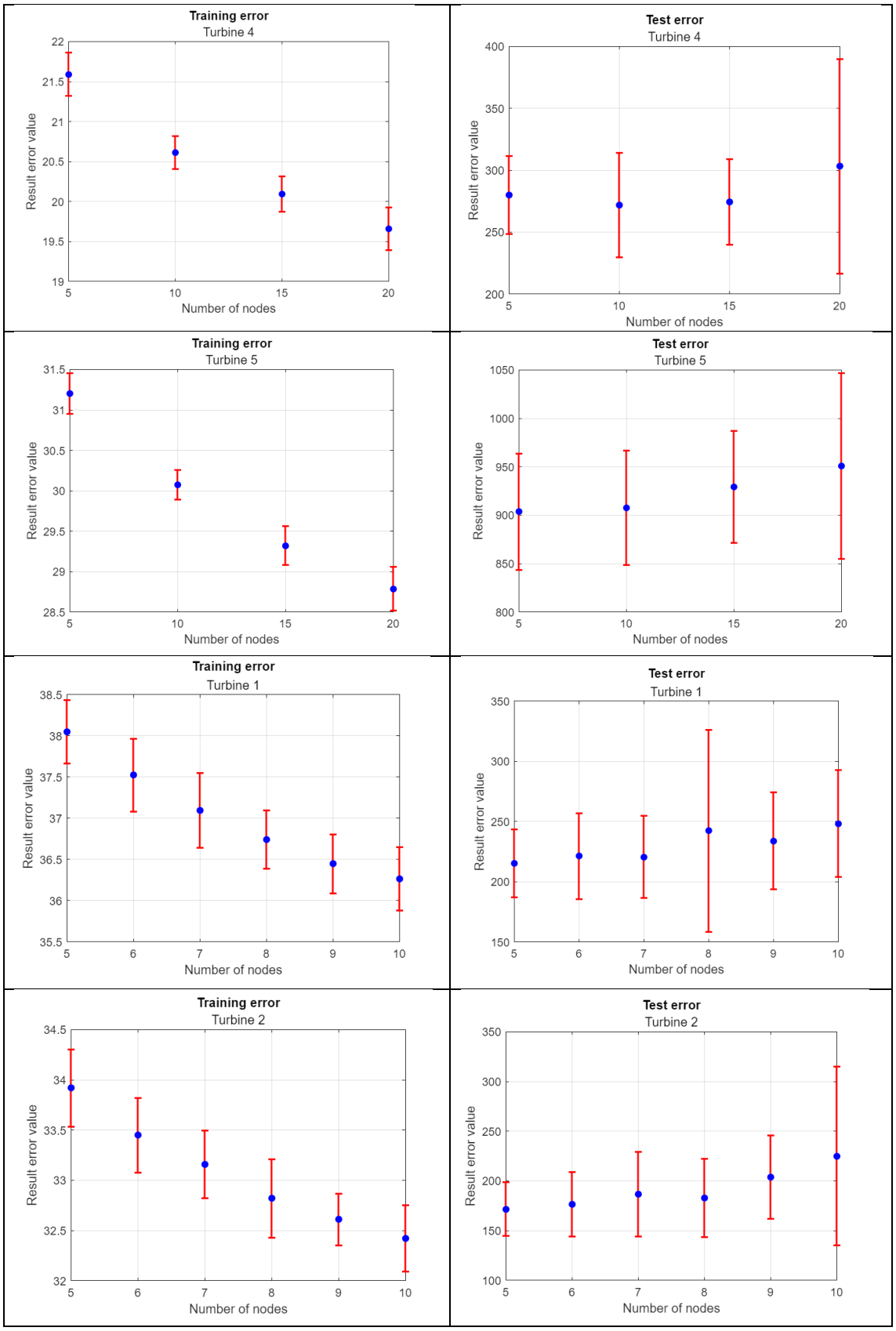
```
data5_m5 = Data55(1:30,:);
data10_m5 = Data105(1:30,:);
data15_m5 = Data155(1:30,:);
data20_m5 = Data205(1:30,:);
data5t_m5 = Data55_test(1:30,:);
data10t_m5 = Data105_test(1:30,:);
data15t_m5 = Data155_test(1:30,:);
data20t_m5 = Data205_test(1:30,:);

S55 = sprintf('%s*', data5_m5{:});
N55 = sscanf(S55, '%f*');


S105 = sprintf('%s*', data10_m5{:});
N105 = sscanf(S105, '%f*');


S155 = sprintf('%s*', data15_m5{:});
N155 = sscanf(S155, '%f*');


S205 = sprintf('%s*', data20_m5{:});
N205 = sscanf(S205, '%f*');


S5t5 = sprintf('%s*', data5t_m5{:});
N5t5 = sscanf(S5t5, '%f*');


S10t5 = sprintf('%s*', data10t_m5{:});
N10t5 = sscanf(S10t5, '%f*');


S15t5 = sprintf('%s*', data15t_m5{:});
N15t5 = sscanf(S15t5, '%f*');


S20t5 = sprintf('%s*', data20t_m5{:});
N20t5 = sscanf(S20t5, '%f*');


m5_m5 = mean(N55)
m10_m5 = mean(N105)
m15_m5 = mean(N155)
m20_m5 = mean(N205)
```

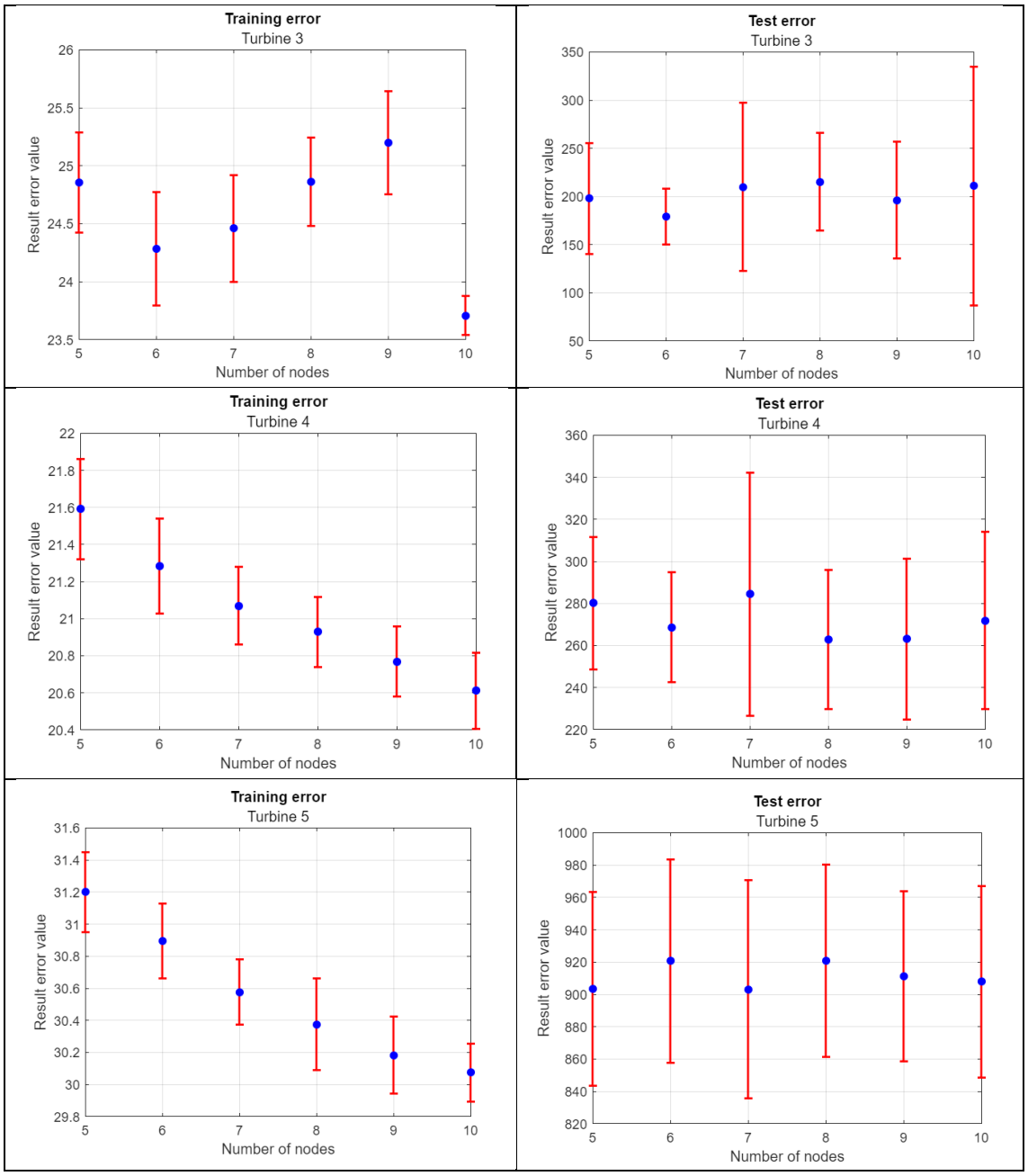
```
m5test_m5 = mean(N5t5)
m10test_m5 = mean(N10t5)
m15test_m5 = mean(N15t5)
m20test_m5 = mean(N20t5)


dv5_m5 = std(N55)
dv10_m5 = std(N105)
dv15_m5 = std(N155)
dv20_m5 = std(N205)
dv5test_m5 = std(N5t5)
dv10test_m5 = std(N10t5)
dv15test_m5 = std(N15t5)
dv20test_m5 = std(N20t5)


x5 = [5 10 15 20];
y5 = [m5_m5 m10_m5 m15_m5 m20_m5];
yt5 = [m5test_m5 m10test_m5 m15test_m5 m20test_m5];


errmat_m5 = [dv5_m5 dv10_m5 dv15_m5 dv20_m5];
errmat2_m5 = [dv5test_m5 dv10test_m5 dv15test_m5 dv20test_m5];


figure;


errorbar(x5, y5, errmat_m5,"red", "LineStyle","none", LineWidth=1.5)
hold on
scatter(x5, y5, "blue", "filled")
axis([0 25 30 40])
title("Training error", "Turbine 1")
xlabel("Number of nodes")
ylabel("Result error value")
grid()


figure;


errorbar(x5, yt5, errmat2_m5,"red", "LineStyle","none", LineWidth=1.5)
hold on
scatter(x5, yt5, "blue", "filled")
axis([0 25 0 600])
title("Test error", "Turbine 1")
xlabel("Number of nodes")
ylabel("Result error value")


grid()
```

Annex D

En aquest annex es poden trobar les representacions gràfiques dels *box-plot* de les Mitjanes d'error amb les seves Desviacions Estàndard de tots els AG. Les primeres deu imatges corresponen a les quantitats de nodes 5, 10, 15 i 20 i les deu següents a les quantitats 5, 6, 7, 8, 9 i 10. A l'esquerra es poden trobar els errors d'entrenament i a la dreta els errors de test.




Training error
Turbine 4
Result error value
Number of nodes
Test error
Turbine 4
Result error value
Number of nodes
Training error
Turbine 5
Result error value
Number of nodes
Test error
Turbine 5
Result error value
Number of nodes
Training error
Turbine 1
Result error value
Number of nodes
Test error
Turbine 1
Result error value
Number of nodes
Training error
Turbine 2
Result error value
Number of nodes
Test error
Turbine 2
Result error value
Number of nodes


Training error
Turbine 3
Result error value
Number of nodes
Test error
Turbine 3
Result error value
Number of nodes
Training error
Turbine 4
Result error value
Number of nodes
Test error
Turbine 4
Result error value
Number of nodes
Training error
Turbine 5
Result error value
Number of nodes
Test error
Turbine 5
Result error value
Number of nodes

Annex E

En aquest annex es troben el codi i els càlculs i realitzats per a les representacions dels models de normalitat i les diferents gràfiques d'error, desviació de l'error i mitjana mòbil simple de l'objectiu 3.

```
t_medio2 = t_(length(t_)/2: length(t_));
% targets machines 6 nodes


% train error
eM1 = t1 - y61';
eM2 = t2 - y62';
eM3 = t3 - y63';
eM4 = t4 - y64';
eM5 = t5 - y65';


% test error
eMt1 = mt1 - ym61';
eMt2 = mt2 - ym62';
eMt3 = mt3 - ym63';
eMt4 = mt4 - ym64';
eMt5 = mt5 - ym65';


% train error representations
figure;
plot(m_time, eM1)
figure;
plot(m_time, eM2)
figure;
plot(m_time, eM3)
figure;
plot(m_time, eM4)
figure;
plot(m_time, eM5)


% test error representations
figure;
plot(t_medio2, eMt1)
figure;
plot(t_medio2, eMt2)
figure;
plot(t_medio2, eMt3)
figure;
plot(t_medio2, eMt4)
figure;
```

```
plot(t_medio2, eMt5)
```

```
% moving average
mAvM1 = abs(mt1) - abs(ym61');
mAvM2 = abs(mt2) - abs(ym62');
mAvM3 = abs(mt3) - abs(ym63');
mAvM4 = abs(mt4) - abs(ym64');
mAvM5 = abs(mt5) - abs(ym65');



movAvg = dsp.MovingAverage(2000, "Method","Sliding window")
mAvResult = movAvg(mAvM1);
mAvResult2 = movAvg(mAvM2);
mAvResult3 = movAvg(mAvM3);
mAvResult4 = movAvg(mAvM4);
mAvResult5 = movAvg(mAvM5);
```

```
figure;
plot(t_medio2, mAvM4)
hold on
plot(t_medio2, mAvResult4, "LineWidth", 1.5, "color", "r", "LineStyle",
"-.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 2")
```

```
%normal deviations x2
figure;
std_dev1 = std(mAvResult5) * 2;
curveS1 = mAvResult5 + std_dev1;
curveI1 = mAvResult5 - std_dev1;


tram_11 = curveI1;
tram_21 = (mAvResult5 - curveI1);
tram_31 = (curveS1 - mAvResult5);
y1 = [tram_11, tram_21, tram_31];
x = t_medio2;



std_dev2 = std(mAvResult4) * 2;
```

```
curveS2 = mAvResult4 + std_dev2;
curveI2 = mAvResult4 - std_dev2;


tram_12 = curveI2;
tram_22 = (mAvResult4 - curveI2);
tram_32 = (curveS2 - mAvResult4);
y2 = [tram_12, tram_22, tram_32];


std_dev3 = std(mAvResult3) * 2;
curveS3 = mAvResult3 + std_dev3;
curveI3 = mAvResult3 - std_dev3;


tram_13 = curveI3;
tram_23 = (mAvResult3 - curveI3);
tram_33 = (curveS3 - mAvResult3);
y3 = [tram_13, tram_23, tram_33];


std_dev4 = std(mAvResult2) * 2;
curveS4 = mAvResult2 + std_dev4;
curveI4 = mAvResult2 - std_dev4;


tram_14 = curveI4;
tram_24 = (mAvResult2 - curveI4);
tram_34 = (curveS4 - mAvResult2);
y4 = [tram_14, tram_24, tram_34];


std_dev5 = std(mAvResult) * 2;
curveS5 = mAvResult + std_dev5;
curveI5 = mAvResult - std_dev5;


tram_15 = curveI5;
tram_25 = (mAvResult - curveI5);
tram_35 = (curveS5 - mAvResult);
y5 = [tram_15, tram_25, tram_35];


% Representació figures
subplot(5, 1, 1)
a = area(x, y1)
a(1).FaceColor = "white";
hold on
p = plot(t_medio2, mAvResult5, "LineWidth", 1.2, "color", "k")
hold off
legend([a(3) a(2) p], {'Superior Deviation', 'Inferior Deviation', 'MMS
signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
```

```
grid on
grid minor
title('Turbine 1')


subplot(5, 1, 2)
a2 = area(x, y2)
a2(1).FaceColor = "white";
hold on
p2 = plot(t_medio2, mAvResult4, "LineWidth", 1.2, "color", "k")
hold off
legend([a2(3) a2(2) p2], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 2')


subplot(5, 1, 3)
a3 = area(x, y3)
a3(1).FaceColor = "white";
hold on
p3 = plot(t_medio2, mAvResult3, "LineWidth", 1.2, "color", "k")
hold off
legend([a3(3) a3(2) p3], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 3')


subplot(5, 1, 4)
a4 = area(x, y4)
a4(1).FaceColor = "white";
hold on
p4 = plot(t_medio2, mAvResult2, "LineWidth", 1.2, "color", "k")
hold off
legend([a4(3) a4(2) p4], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 4')
xlabel('time')


subplot(5, 1, 5)
a5 = area(x, y5)
a5(1).FaceColor = "white";
hold on
```

```
p5 = plot(t_medio2, mAvResult, "LineWidth", 1.2, "color", "k")
hold off
legend([a5(3) a5(2) p5], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 5')
xlabel('Time in 5min steps.')
```

```
%normal deviations
figure;
std_dev1 = std(mAvResult5);
curveS1 = mAvResult5 + std_dev1;
curveI1 = mAvResult5 - std_dev1;


tram_11 = curveI1;
tram_21 = (mAvResult5 - curveI1);
tram_31 = (curveS1 - mAvResult5);
y1 = [tram_11, tram_21, tram_31];
x = t_medio2;




std_dev2 = std(mAvResult4);
curveS2 = mAvResult4 + std_dev2;
curveI2 = mAvResult4 - std_dev2;


tram_12 = curveI2;
tram_22 = (mAvResult4 - curveI2);
tram_32 = (curveS2 - mAvResult4);
y2 = [tram_12, tram_22, tram_32];


std_dev3 = std(mAvResult3);
curveS3 = mAvResult3 + std_dev3;
curveI3 = mAvResult3 - std_dev3;


tram_13 = curveI3;
tram_23 = (mAvResult3 - curveI3);
tram_33 = (curveS3 - mAvResult3);
y3 = [tram_13, tram_23, tram_33];


std_dev4 = std(mAvResult2);
curveS4 = mAvResult2 + std_dev4;
curveI4 = mAvResult2 - std_dev4;
```

```
tram_14 = curveI4;
tram_24 = (mAvResult2 - curveI4);
tram_34 = (curveS4 - mAvResult2);
y4 = [tram_14, tram_24, tram_34];


std_dev5 = std(mAvResult);
curveS5 = mAvResult + std_dev5;
curveI5 = mAvResult - std_dev5;


tram_15 = curveI5;
tram_25 = (mAvResult - curveI5);
tram_35 = (curveS5 - mAvResult);
y5 = [tram_15, tram_25, tram_35];


% Representació figures
subplot(5, 1, 1)
a = area(x, y1)
a(1).FaceColor = "white";
hold on
p = plot(t_medio2, mAvResult5, "LineWidth", 1.2, "color", "k")
hold off
legend([a(3) a(2) p], {'Superior Deviation', 'Inferior Deviation', 'MMS
signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 1')


subplot(5, 1, 2)
a2 = area(x, y2)
a2(1).FaceColor = "white";
hold on
p2 = plot(t_medio2, mAvResult4, "LineWidth", 1.2, "color", "k")
hold off
legend([a2(3) a2(2) p2], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 2')


subplot(5, 1, 3)
a3 = area(x, y3)
a3(1).FaceColor = "white";
hold on
p3 = plot(t_medio2, mAvResult3, "LineWidth", 1.2, "color", "k")
hold off
```

```
legend([a3(3) a3(2) p3], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 3')


subplot(5, 1, 4)
a4 = area(x, y4)
a4(1).FaceColor = "white";
hold on
p4 = plot(t_medio2, mAvResult2, "LineWidth", 1.2, "color", "k")
hold off
legend([a4(3) a4(2) p4], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 4')
xlabel('time')


subplot(5, 1, 5)
a5 = area(x, y5)
a5(1).FaceColor = "white";
hold on
p5 = plot(t_medio2, mAvResult, "LineWidth", 1.2, "color", "k")
hold off
legend([a5(3) a5(2) p5], {'Superior Deviation', 'Inferior Deviation',
'MMS signal'})
axis([1.357*10^9, 1.427*10^9, -100, 100])
grid on
grid minor
title('Turbine 5')
xlabel('Time in 5min steps.')

% predicted behaviour
figure;


subplot(5, 1, 1)
plot(t_medio2, mAvM5)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 1')


subplot(5, 1, 2)
plot(t_medio2, mAvM4)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 2')
```

```
subplot(5, 1,  3)
plot(t_medio2, mAvM3)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 3')


subplot(5, 1, 4)
plot(t_medio2, mAvM2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 4')


subplot(5, 1, 5)
plot(t_medio2, mAvM1)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 5')
xlabel('Time in 5min steps.')
% error smoothing
figure;
subplot(5, 1, 1)
plot(t_medio2, mAvResult5, "LineWidth", 2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 1')
subplot(5, 1, 2)
plot(t_medio2, mAvResult4, "LineWidth", 2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 2')
subplot(5, 1, 3)
plot(t_medio2, mAvResult3, "LineWidth", 2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 3')
subplot(5, 1, 4)
plot(t_medio2, mAvResult2, "LineWidth", 2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 4')
subplot(5, 1, 5)
plot(t_medio2, mAvResult, "LineWidth", 2)
axis([1.35*10^9, 1.42*10^9, -100, 100])
title('Turbine 5')
xlabel('Time in 5min steps.')


figure;
subplot(5, 1, 1)
plot(t_medio2, mAvM5)
hold on
plot(t_medio2, mAvResult5, "LineWidth", 1.5, "color", "r", "LineStyle",
"-.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 1")
```

```
subplot(5, 1, 2)
plot(t_medio2, mAvM4)
hold on
plot(t_medio2, mAvResult4, "LineWidth", 1.5, "color", "r", "LineStyle",
"-.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 2")


subplot(5, 1,  3)
plot(t_medio2, mAvM3)
hold on
plot(t_medio2, mAvResult3, "LineWidth", 1.5, "color", "r", "LineStyle",
"-.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 3")


subplot(5, 1, 4)
plot(t_medio2, mAvM2)
hold on
plot(t_medio2, mAvResult2, "LineWidth", 1.5, "color", "r", "LineStyle",
"-.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 4")


subplot(5, 1, 5)
plot(t_medio2, mAvM1)
hold on
plot(t_medio2, mAvResult, "LineWidth", 1.5, "color", "r", "LineStyle", "-
.")
axis([1.355*10^9, 1.425*10^9, -100, 100])
legend("Error signal", "MMS Signal")
title("Turbine 5")
xlabel('Time in 5min steps.')




% prediction comparison
figure;
subplot(5, 1, 1)
plot(t_medio2, m1_medio2)
hold on
plot(t_medio2, mAvResult5, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
title("Turbine 1")
```

```
subplot(5, 1, 2)
plot(t_medio2, m2_medio2)
hold on
plot(t_medio2, mAvResult4, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
title("Turbine 2")


subplot(5, 1, 3)
plot(t_medio2, m3_medio2)
hold on
plot(t_medio2, mAvResult3, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
title("Turbine 3")


subplot(5, 1, 4)
plot(t_medio2, m4_medio2)
hold on
plot(t_medio2, mAvResult2, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
title("Turbine 4")


subplot(5, 1, 5)
plot(t_medio2, m5_medio2)
hold on
plot(t_medio2, mAvResult, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
title("Turbine 5")
xlabel('Time in 5min steps.')

figure;
plot(t_medio2, m5_medio2)
hold on
plot(t_medio2, mAvResult, "LineWidth", 2, "color", "r")
axis([1.355*10^9, 1.425*10^9, -100, 150])
legend("Real signal", "MMS Signal")
grid on
title("Turbine 5")
xlabel('Time in 5min steps.')
```