KTH ROYAL INSTITUTE
OF TECHNOLOGY

Lecture 15
Artificial Neural Networks

On the Shoulder of Giants

Much of the material in this slide set is based upon:

”Automated Learning techniques in Power Systems”
by L. Wehenkel, Université Liege

”Probability based learning” Josephine Sullivan, KTH

”Entropy and Information Gain” by F.Aiolli,
University of Padova

Contents

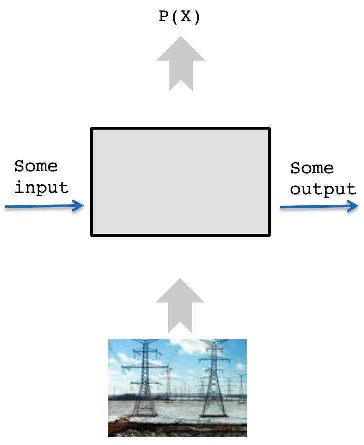
Repeating from last time

Artificial Neural Networks

Power Systems Analysis – An automated learning approach

Understanding states in the power system is established through observation of inputs and outputs without regard to the physical electrotechnical relations between the states.

Adding knowledge about the electrotechnical rules means adding heuristics to the learning.



Given a set of examples (the learning set (LS)) of associated input/output pairs, derive a general rule representing the underlying input/output relationship, which may be used to explain the observed pairs and/or predict output values for any new unseen input.

Contents

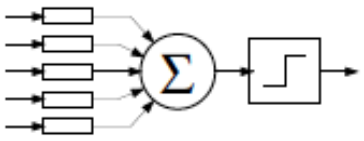
Repeating from last time

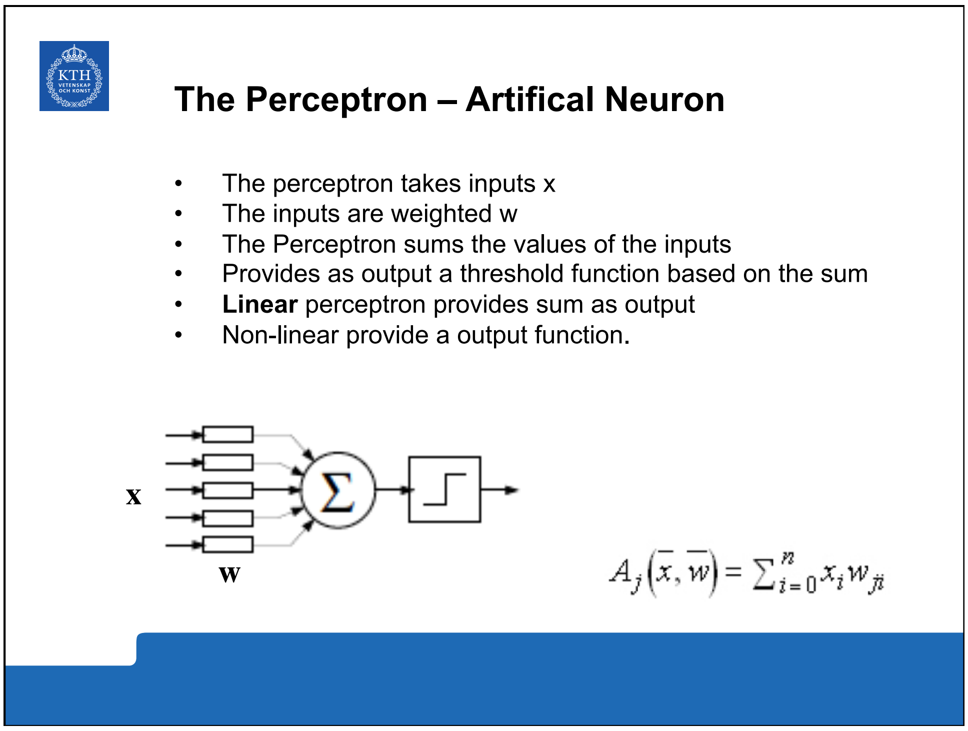
Artificial Neural Networks

Artificial Neural Networks - Introduction

Inspired by the Human Nerve system

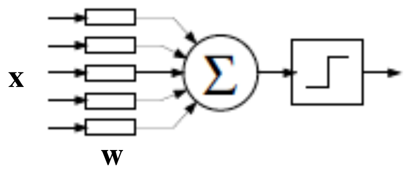
A close resemblance ?



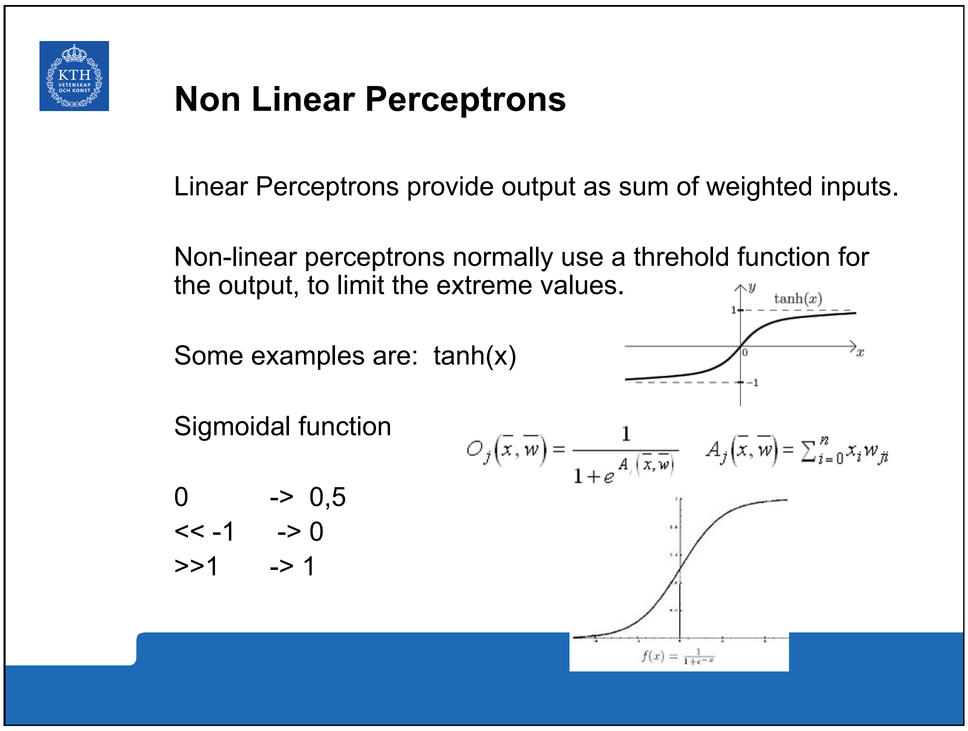


The Perceptron – Artifical Neuron

- The perceptron takes inputs x
- The inputs are weighted w
- The Perceptron sums the values of the inputs
- Provides as output a threshold function based on the sum
- **Linear** perceptron provides sum as output
- Non-linear provide a output function.



$$A_j(\bar{x}, \bar{w}) = \sum_{i=0}^{n} x_i w_{ji}$$

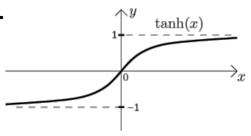


Non Linear Perceptrons

Linear Perceptrons provide output as sum of weighted inputs.

Non-linear perceptrons normally use a threhold function for the output, to limit the extreme values.

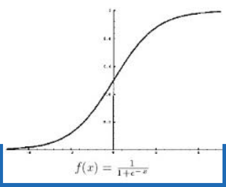
Some examples are: tanh(x)



Sigmoidal function

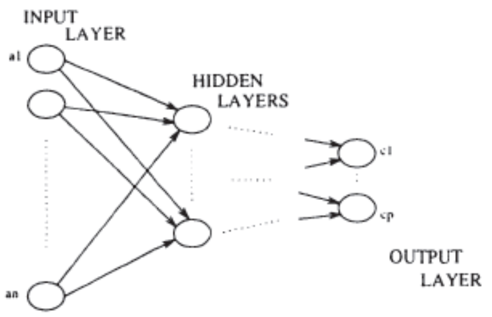
$$O_j(\bar{x}, \bar{w}) = \frac{1}{1+e^{A_j(\bar{x}, \bar{w})}} \qquad A_j(\bar{x}, \bar{w}) = \sum_{i=0}^{n} x_i w_{ji}$$

0	-> 0,5
<< -1	-> 0
>>1	-> 1



Artificial Neural Network

Multi Layer Perceptrons (MLP)
A network of interconnected Perceptrons in several layers
First layer recives input, forwards to second layer etc.
Normally one hidden layer is sufficient to create good mappings



Where is the "learning" in ANN

Given an input vector $a(o)$ (attributes of an object)

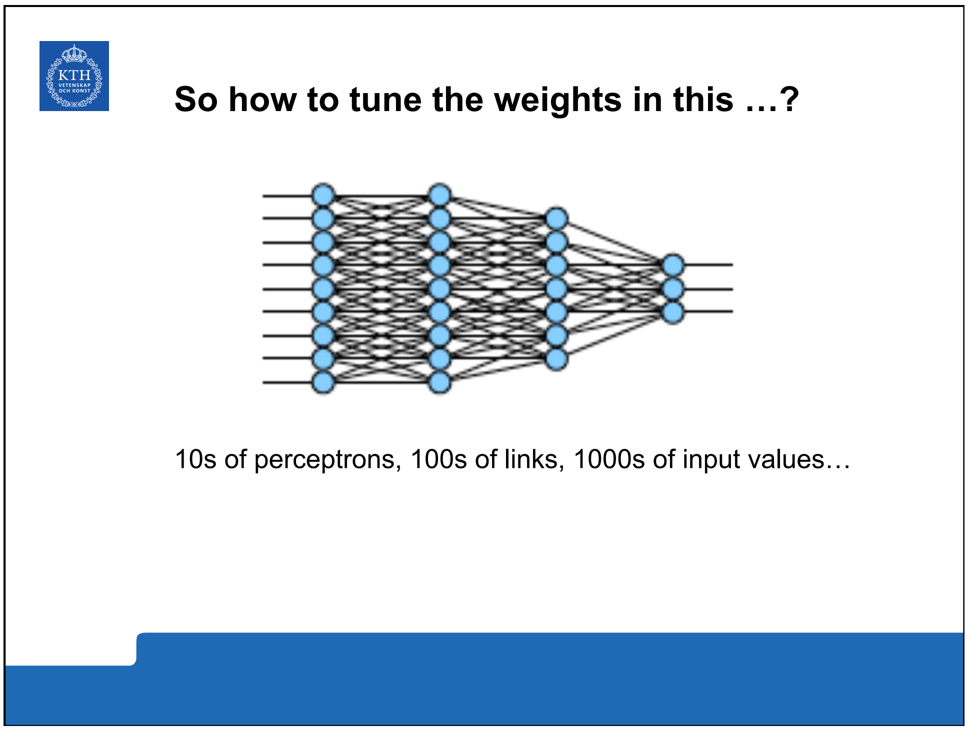
For a classification problem

We want to assign it to a class C_i

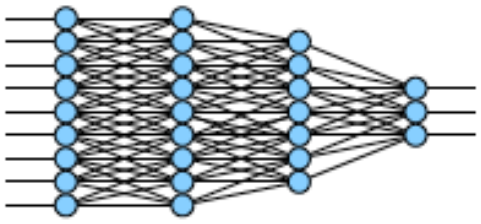
For a regression problem

We want it to approximate a value y

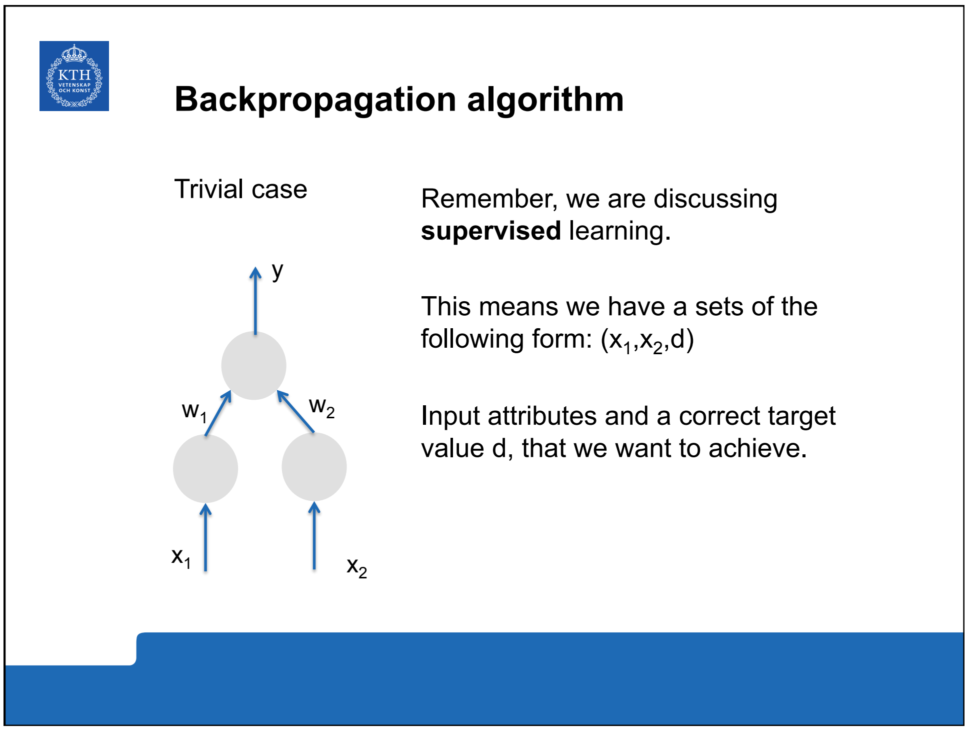
We have to tune the weights of the inputs of the perceptrons



So how to tune the weights in this …?

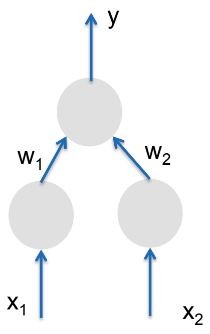


10s of perceptrons, 100s of links, 1000s of input values…



Backpropagation algorithm

Trivial case



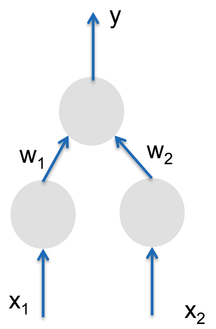
Remember, we are discussing **supervised** learning.

This means we have a sets of the following form: (x_1,x_2,d)

Input attributes and a correct target value d, that we want to achieve.

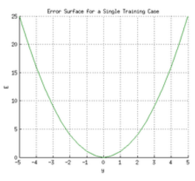
Backpropagation algorithm

Trivial case



The Least Squares Error

$$E = (d - y)^2$$



For a linear Perceptron

$$y = x_1 w_1 + x_2 w_2$$

Find minima of $E(y)$ w.r.t (w_1, w_2)

Finding minima - gradient descent

In the general case, we want to find the minima of the Error function with regards to the weights

$$E_j(\bar{x}, \bar{w}, d) = \left(O_j(\bar{x}, \bar{w}) - d_j\right)^2$$

If we can find the derivative of the error function, we can use that to find a suitable adjustment of the weights.

………..

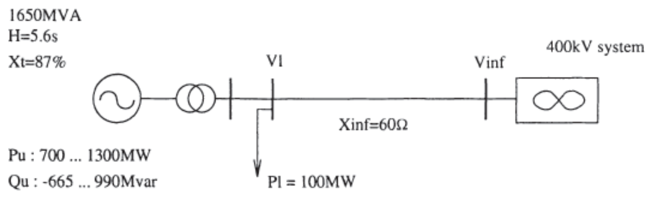
$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

Where are we now?

1. We have created a ANN with a ”suitable number of layers” and perceptrons
2. We have chosen a threshold function for the perceptrons
3. We have allocated random weights to all links
4. We use our Training set to tune the weights using the Backdrop algorithm.
5. In the Backdrop algorithm we had to determine the minima of the error function and derived a formula on how to adjust weights to reach the minima.
6. We adjust the weights, try a new test set and run the whole process again.

And this was for a ANN with one layer…..

Example from Automatic Learning techniques in Power Systems



One Machine Infinite Bus (OMIB) system

- Assuming a fault close to the Generator will be cleared within 155 ms by protection relays
- We need to identify situations in which this clearing time is sufficient and when it is not
- Under certain loading situations, 155 ms may be too slow.

Source: Automatic Learning techniques in Power Systems, L. Wehenkel

OMIB – further information

In the OMIB system the following parameters influence security

- Amount of active and reactive power of the generator (
- Amount of load nearby the generator (Pl)
- Voltage magnitudes at the load bus and at the infinite bus Short-circuit reactance Xinf, representing the effect of variable topology in the large system represented by the infinite bus.

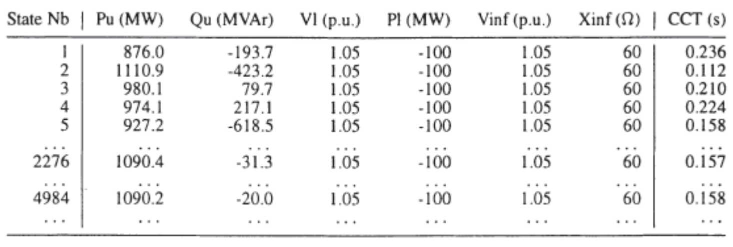
In the example, Voltages at generator and Infinite bus are assumed similar and constant for simplicity

Source: Automatic Learning techniques in Power Systems, L. Wehenkel

Our database of objects with attributes

In a simulator, we randomly sample values for P_u and Q_u creating a database with 5000 samples (objects) and for each object we have a set of attributes $(P_u, Q_u, V_1, P_1, V_{inf}, X_{inf}, CCT)$ as per below.

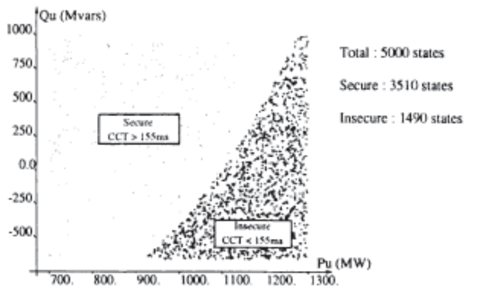
Table 1.1. Sample of OMIB operating states

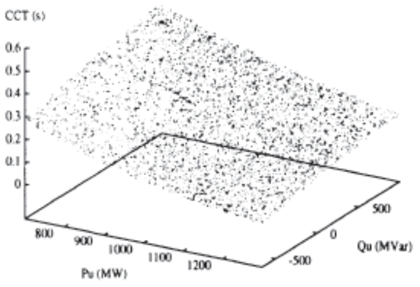


State Nb	Pu (MW)	Qu (MVAr)	Vl (p.u.)	Pl (MW)	Vinf (p.u.)	Xinf (Ω)	CCT (s)
1	876.0	-193.7	1.05	-100	1.05	60	0.236
2	1110.9	-423.2	1.05	-100	1.05	60	0.112
3	980.1	79.7	1.05	-100	1.05	60	0.210
4	974.1	217.1	1.05	-100	1.05	60	0.224
5	927.2	-618.5	1.05	-100	1.05	60	0.158
...	...	...	...	...	...	...	...
2276	1090.4	-31.3	1.05	-100	1.05	60	0.157
...	...	...	...	...	...	...	...
4984	1090.2	-20.0	1.05	-100	1.05	60	0.158
...	...	...	...	...	...	...	...

Source: Automatic Learning techniques in Power Systems, L. Wehenkel

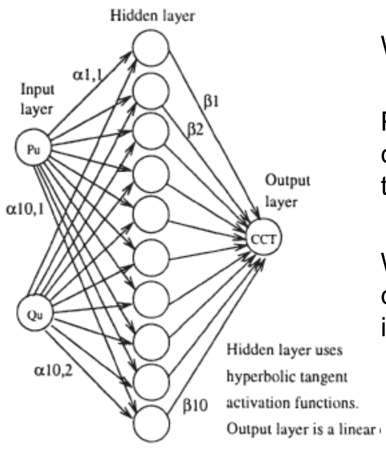
Plot of database content





Source: Automatic Learning techniques in Power Systems, L. Wehenkel

Initial ANN for the OMIB problem



Weights are random

Perceptrons use linear combination of inputs and tanh function

We want to calculate the clearing time (CCT), i.e. This is a **Regression** problem

$$\text{Output}_i(\text{state}) = \tanh(\alpha_{i,1} Pu(\text{state}) + \alpha_{i,2} Qu(\text{state}) + \theta_i),$$

Output and Error function

The Output function is:

$$\mathrm{CCT}_{\mathrm{MLP}}(\mathrm{state}) = \sum_{i=1\ldots 10} \beta_i \tanh(\alpha_{i,1} Pu(\mathrm{state}) + \alpha_{i,2} Qu(\mathrm{state}) + \theta_i),$$

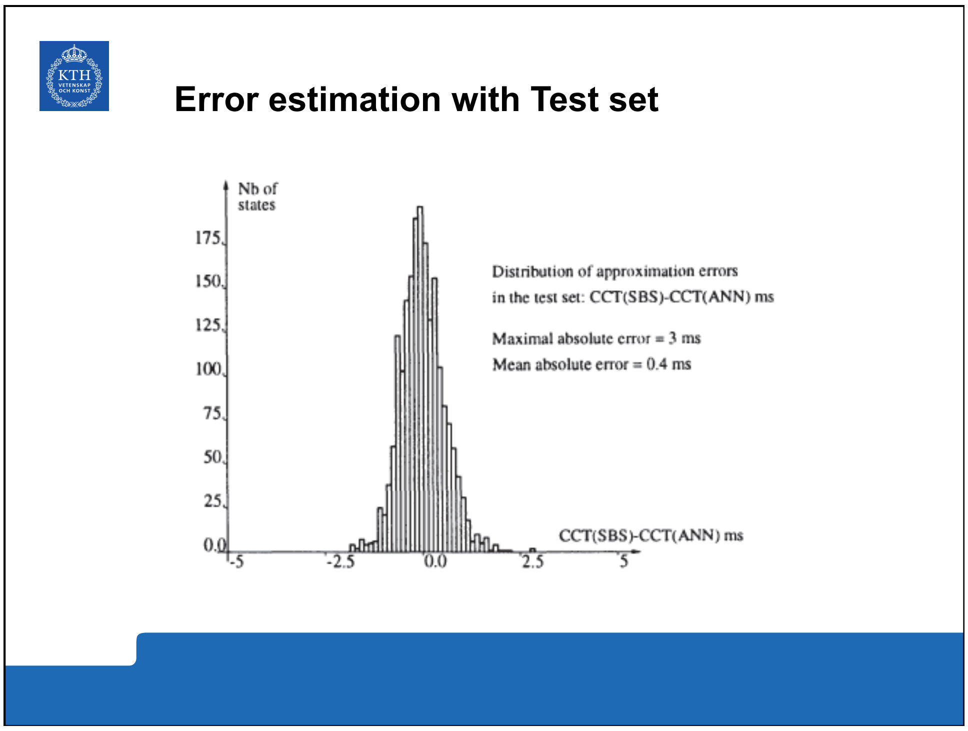
The error function is:

$$SE = N^{-1} \sum_{\mathrm{state} \in \boldsymbol{LS}} |\mathrm{CCT}(\mathrm{state}) - \mathrm{CCT}_{\mathrm{MLP}}(\mathrm{state})|^2,$$

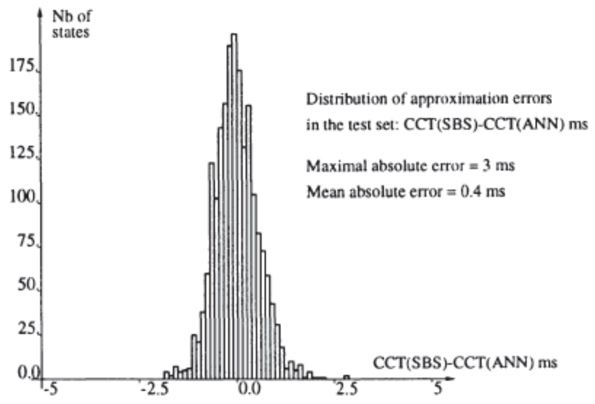
The final ANN structure is

After 46 iterations

$$\begin{aligned}
\mathrm{CCT}_{\mathrm{MLP}} = \; & -0.602710 \tanh(0.000194 Pu - 0.00034 Qu - 0.93219) \\
& -0.401320 \tanh(0.000822 Pu - 0.00020 Qu - 0.76681) \\
& +0.318249 \tanh(0.000239 Pu - 0.00050 Qu - 0.29351) \\
& -0.287230 \tanh(0.002004 Pu - 0.00034 Qu - 1.20080) \\
& +0.184522 \tanh(0.000131 Pu - 0.00057 Qu - 0.03152) \\
& +0.177701 \tanh(0.001799 Pu - 0.00011 Qu - 2.08190) \\
& -0.150720 \tanh(0.001530 Pu - 0.00056 Qu - 1.68040) \\
& +0.142678 \tanh(0.002152 Pu - 0.00046 Qu - 1.72280) \\
& -0.067897 \tanh(0.001910 Pu - 0.00051 Qu - 1.71343) \\
& -0.056020 \tanh(0.000202 Pu - 0.00085 Qu - 0.39876)
\end{aligned}$$



Error estimation with Test set


Nb of states
CCT(SBS)-CCT(ANN) ms
Distribution of approximation errors in the test set: CCT(SBS)-CCT(ANN) ms
Maximal absolute error = 3 ms
Mean absolute error = 0.4 ms