

# Ειδικά Συστήματα Ελέγχου Πλοίου

ΣΝΜΜ 2019

## Μάθημα 3B: Νευρωνικά Δίκτυα - Υπολογιστικά Εργαλεία

**Γεώργιος Παπαλάμπρου**

Επίκουρος Καθηγητής ΕΜΠ

[george.papalambrou@lme.ntua.gr](mailto:george.papalambrou@lme.ntua.gr)

Εργαστήριο Ναυτικής Μηχανολογίας (Κτίριο Λ)

Σχολή Ναυπηγών Μηχανολόγων Μηχανικών

Εθνικό Μετσόβιο Πολυτεχνείο

March 19, 2019

# Περιεχόμενα

1 MATLAB

2 Σχεδίαση Νευρωνικού Δικτύου για τον Virtual Sensor

# Περιεχόμενο Μαθήματος

- M3A: Εισαγωγή στην Μηχανική Μάθηση, Νευρωνικά Δίκτυα. Βιβλιογραφία.
- M3B: ΝΔ: Υπολογιστικά εργαλεία
- M3Γ: Εφαρμογή ΝΔ σε μηχανές εσωτερικής καύσης/virtual sensors

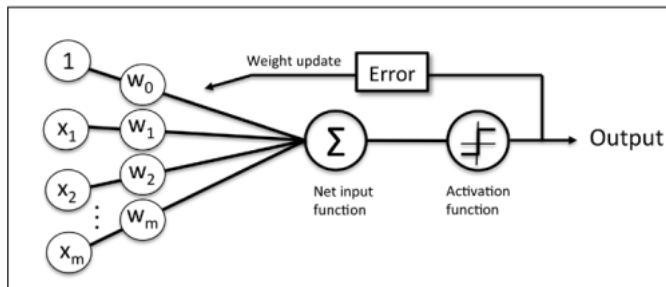
# Σύγχρονα εργαλεία για την σχεδίαση ΝΔ

Βασικές πηγές:

- Τα τελευταία χρόνια η σχεδίαση ενός ΝΔ γίνεται με ευρεία χρήση της γλώσσας Python, του MATLAB, αλλά και με άλλες πλατφόρμες (Tensor Flow, Keras, κλπ).
- Για το μάθημα 2019, η σχεδίαση θα γίνει σε MATLAB.

# Σύγχρονα εργαλεία σε Python

- Βιβλίο Sebastian Raschka: Python Machine Learning, 2nd Edition.
- Γίνεται χρήση Python και βιβλιοθηκών numpy, SciPy, Scikit-learn για ΝΔ single - και multi - layer.
- <https://sebastianraschka.com/books.html>



# MATLAB NN Toolbox


The screenshot displays the MATLAB R2017a App Designer interface. The top navigation bar includes 'HOME', 'PLOTS', and 'APPS' tabs. Below this, there are three main sections of toolboxes:

- FAVORITES:** Contains icons for Curve Fitting, Optimization, PID Tuner, System Identification, Signal Analyzer, Image Acquisition, Instrument Control, MATLAB Coder, and Application Compiler.
- MATH, STATISTICS AND OPTIMIZATION:** Contains icons for Classification Learner, Curve Fitting (marked with a star), Distribution Fitter, Neural Net Clustering, Neural Net Fitting, Neural Net Pattern Recognition, Neural Net Time Series, Optimization (marked with a star), PDE, and Regression Learner.
- CONTROL SYSTEM DESIGN AND ANALYSIS:** Contains icons for Control System Designer, Control System Tuner, Fuzzy Logic Designer, Linear System Analyzer, Model Reducer, MPC Designer, Neuro-Fuzzy Designer, PID Tuner (marked with a star), and System Identification (marked with a star).

On the left side, there is a 'Current Folder' pane showing a file explorer view with folders like 'slprj' and 'untitled\_ert\_rtw', and a file 'applin1.m'. Below that is a 'Details' pane and a 'Workspace' pane. The workspace table is as follows:

Name	Value
E	1x196 double
net	1x1 network
signal	1x201 cell

**Neural Time Series (ntstool)**



## Welcome to the Neural Network Time Series Tool.

Solve a nonlinear time series problem with a dynamic neural network.

### Introduction

Prediction is a kind of dynamic filtering, in which past values of one or more time series are used to predict future values. Dynamic neural networks, which include tapped delay lines are used for nonlinear filtering and prediction.


There are many applications for prediction. For example, a financial analyst might want to predict the future value of a stock, bond or other financial instrument. An engineer might want to predict the impending failure of a jet engine.

Predictive models are also used for system identification (or dynamic modelling), in which you build dynamic models of physical systems. These dynamic models are important for analysis, simulation, monitoring and control of a variety of systems, including manufacturing systems, chemical processes, robotics and aerospace systems.

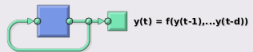
This tool allows you to solve three kinds of nonlinear time series problems shown in the right panel. Choose one and click [Next].

### Select a Problem

Nonlinear Autoregressive with External (Exogenous) Input (NARX)  
 Predict series  $y(t)$  given  $d$  past values of  $y(t)$  and another series  $x(t)$ .

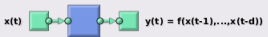



Nonlinear Autoregressive (NAR)  
 Predict series  $y(t)$  given  $d$  past values of  $y(t)$ .





Nonlinear Input-Output  
 Predict series  $y(t)$  given  $d$  past values of series  $x(t)$ .


**Important Note:** NARX solutions are more accurate than this solution. Only use this solution if past values of  $y(t)$  will not be available when deployed.





 Choose a problem, then click [Next].

 Neural Network Start

 Welcome

 Back

 Next

 Cancel

ΓΠ-Εισαγωγή στα Νευρωνικά Δίκτυα


ΣΝΜΜ 2019

March 19, 2019

8 / 22



Neural Fitting (nftool)



## Welcome to the Neural Fitting app.

Solve an input-output fitting problem with a two-layer feed-forward neural network.

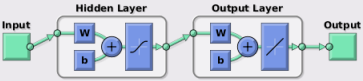
### Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating engine emission levels based on measurements of fuel consumption and speed (`engine_dataset`) or predicting a patient's bodyfat level based on body measurements (`bodyfat_dataset`).

The Neural Fitting app will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

### Neural Network



A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (`fitnet`), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network will be trained with Levenberg-Marquardt backpropagation algorithm (`trainlm`), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (`trainscg`) will be used.

➔ To continue, click [Next].

Neural Network Start

Welcome

Back

Next

Cancel

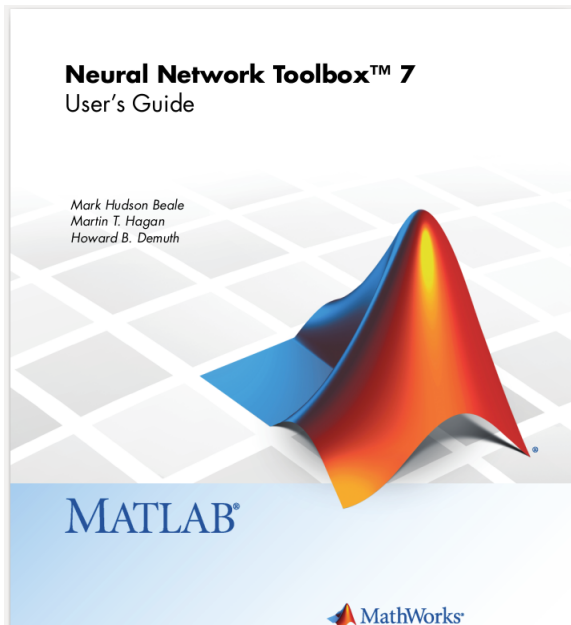
ΓΠ-Εισαγωγή στα Νευρωνικά Δίκτυα

ΣΝΜΜ 2019

March 19, 2019

9 / 22

# MATLAB, User's Guide, ver. 7 (pdf)



- Ανοιχτή πλατφόρμα για σχεδίαση ΝΔ, με Python, MATLAB.

Docs » pyrenn: A recurrent neural network toolbox for python and matlab [Edit on GitHub](#)

## pyrenn: A recurrent neural network toolbox for python and matlab

**Maintainer:** Dennis Atabay, <dennis.atabay@tum.de>

**Organization:** Institute for Energy Economy and Application Technology, Technische Universität München

**Version:** 0.1

**Date:** Jun 30, 2018

**Copyright:** This documentation is licensed under a [Creative Commons Attribution 4.0 International license](#).

DOI: [10.5281/zenodo.45022](https://doi.org/10.5281/zenodo.45022)

### Contents

This documentation contains the following pages:

- [Create a neural network](#)
- [Train a neural network](#)
- [Use a trained neural network](#)
- [Save and load a neural network](#)
- [Examples](#)

### Features

- pyrenn allows to create a wide range of (recurrent) neural network configurations
- It is very easy to create, train and use neural networks
- It uses the [Levenberg-Marquardt algorithm](#) (a second-order Quasi-Newton optimization method) for training, which is much faster than first-order methods like [gradient descent](#). In the matlab version additionally the [Broyden-Fletcher-Goldfarb-Shanno algorithm](#) is implemented
- The python version is written in pure python and numpy and the matlab version in pure matlab

# G. Hinton, UTor, Ca

- Στο site [www.cs.toronto.edu/~hinton/](http://www.cs.toronto.edu/~hinton/) υπάρχουν οι δημοσιεύσεις, tutorials, MATLAB codes, κλπ.

← → ↻ 🏠 ⓘ www.cs.toronto.edu/~hinton/ 80% ⋮ ⌘ ⌘ 🔍 Search

## Geoffrey E. Hinton

➡ If you want to make a charitable donation in memory of Jacqueline Ford, please visit <https://www.goodfooddelivery.com/the-charity/>

Department of Computer Science  
**University of Toronto**  
6 King's College Rd.  
Toronto, Ontario

**email:** geoffrey [dot] hinton [at] gmail [dot] com  
**voice:** send email  
**fax:** scan and send email

### Information for prospective students:

I advise interns at Brain team Toronto.  
I also advise some of the residents in the [Google Brain Residents Program](#).  
I will not be taking any more visiting students, summer students or visitors at the University of Toronto. I will not be the graduate students, but I may co-advise a few graduate students with Prof. Roger Grosse or soon to be Prof. Jimmy Ba.

### News

[Results of the 2012 competition to recognize 1000 different types of object](#)  
[How George Dahl won the competition to predict the activity of potential drugs](#)  
[How Vlad Mnih won the competition to predict job salaries from job advertisements](#)  
[How Laurens van der Maaten won the competition to visualize a dataset of potential drugs](#)

[Using big data to make people vote against their own interests](#)  
[A possible motive for making people vote against their own interests](#)

### Basic papers on deep learning

Hinton, G. E., Osindero, S. and Teh, Y. (2006)  
A fast learning algorithm for deep belief nets.  
*Neural Computation*, **18**, pp 1527-1554. [pdf]  
[Movies of the neural network generating and recognizing digits](#)

Hinton, G. E. and Salakhutdinov, R. R. (2006)  
Reducing the dimensionality of data with neural networks.  
*Science*, Vol. 313, no. 5786, pp. 504 - 507, 28 July 2006.  
[full paper.] [supporting online material (pdf).] [Matlab code.]

➡ LeCun, Y., Bengio, Y. and Hinton, G. E. (2015)  
Deep Learning  
*Nature*, Vol. 521, pp 436-444. [pdf]

### Papers on deep learning without much math

Hinton, G. E. (2007)  
To recognize shapes, first learn to generate images  
In P. Cisek, T. Drew and J. Kalaska (Eds.)  
*Computational Neuroscience: Theoretical Insights into Brain Function*. Elsevier. [pdf of final draft]

Hinton, G. E. (2007)  
Learning Multiple Layers of Representation.  
*Trends in Cognitive Sciences*, Vol. 11, pp 428-434. [pdf]

Hinton, G. E. (2014)

### BIOGRAPHICAL INFORMATION

— Curriculum Vitae [pdf]  
— Biographical sketch  
— Brief Bio  
— Photographs (1.jpg)  
— Photographs (2.jpg)

### PUBLICATIONS

— Publications by year  
— Slides of public talks

### 2012 COURSE/EA COURSE LECTURES:

Neural Networks for Machine Learning  
— Lecture1 (mp4)  
— Lecture 5 (slides) (pdf or .pdf)

### OLD UNIVERSITY OF TORONTO COURSES

— cs-371 Spring 2013 (undergrad)  
— cs-433-5 Spring 2013 (graduate)

### VIDEO TALKS & TUTORIALS

— YouTube (2012) *Rein. Sec. and Machine Learning (1hr)*  
— YouTube (2012) *The Next Generation of Neural Networks (1hr)*  
— YouTube (2010) *Recent Developments in Deep Learning (1hr)*  
— Interview on CBC radio "Quirks and Quarks" Feb 11 2011  
— Interview on CBC radio "The Current" Mar 3 2013  
— Tutorial (2008) *Deep Belief Nets (short pdf and mp3)*  
— Workshop Talk (2007) *How to do backpropagation in a brain*  
(20mins) [pdf] [mp3] [mp4] [mp7] [mp10] [mp11] [mp12]

### OLD TUTORIAL SLIDES

— 2011 NIPS workshop talk [pdf] [mp3]  
— paper on "Transforming Autoencoders"  
— 2007 NIPS tutorial (short pdf and mp3)  
— *Reinforcement Learning* tutorial  
— CPAR Summer School 2007  
— CPAR Summer School 2006  
— CPAR Summer School 2005  
— List of Past Tutorials

### MOVIES

— generating digits  
— generating walks (Graham Taylor)  
— speaking with a silicon (5 min) (Peta)

### MATLAB CODE

— Matlab for Science paper  
— L3NE software  
— *lectures from motor program*  
— Ink Press lecture tour

# Tensor Flow

- TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google...(Wikipedia)

TensorFlow

Get Started with TensorFlow

TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. See the sections below to get started.

### Learn and use ML

The high-level TensorFlow API provides building blocks to create and train deep learning models. Start with these beginner-friendly notebook examples, then read the TensorFlow Keras guide.

- Basic classification
- Text classification
- Regression
- Clustering and understanding
- Time and text

[View TensorFlow guide](#)

### Research and experimentation

Eager execution provides an imperative, definition-by-usage interface for advanced operations, deep custom layers, training pipelines, and training loops with auto-differentiation. Start with these notebooks, then read the eager execution guide.

- Eager execution basics
- Automatic differentiation and gradient tape
- Custom training loops
- Custom layers
- Custom training with logging

### ML at production scale

Estimators can train large models on multiple machines in a production environment. TensorFlow provides a collection of pre-made Estimators to implement common ML algorithms. See the Estimators guide.

- Build a linear model with Estimators
- ML and deep learning with Estimators
- Boosted trees
- How to build a simple but classifier with TF.nn
- Build a Convolutional Neural Network using Estimators

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, y_train, x_test, y_test = x_train / 255.0, y_train, x_test / 255.0, y_test

model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

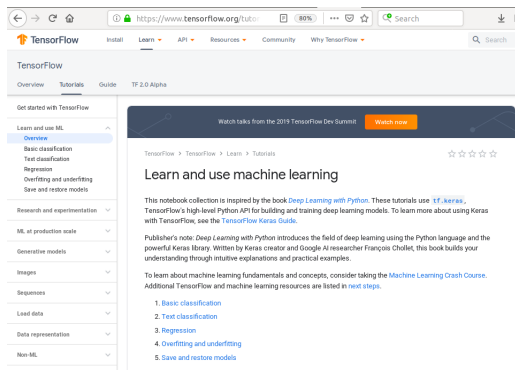
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

[Run code now](#) Try Google's interactive notebook

# Tensor Flow

- Το βασικό interface είναι σε γλώσσα Python, αλλά για λόγους ταχύτητας είναι γραμμένο σε C++.



The screenshot shows the TensorFlow website's 'Learn' section. The browser address bar displays 'https://www.tensorflow.org/tutorials'. The page title is 'Learn and use machine learning'. The main content area includes a 'Watch now' button, a breadcrumb trail 'TensorFlow > TensorFlow > Learn > Tutorials', and a list of five tutorial topics: 1. Basic classification, 2. Text classification, 3. Regression, 4. Overfitting and underfitting, and 5. Save and restore models. The left sidebar contains a navigation menu with categories like 'Get started with TensorFlow', 'Learn and use ML', 'Research and experimentation', 'ML at production scale', 'Generative models', 'Images', 'Sequences', 'Lead data', 'Data representation', and 'Non-ML'.

# Tensor Flow

- Συνήθως το TF τρέχει ως η βασική μηχανή και σε υψηλότερο επίπεδο υπάρχουν (απλούστερα) APIs όπως το Keras κα.

The screenshot shows the TensorFlow website's 'Overview' page. At the top, there's a navigation bar with 'TensorFlow' and links for 'Install', 'Learn', 'API', 'Resources', 'Community', and 'Why TensorFlow'. Below this, there are tabs for 'Overview', 'Tutorials', 'Guide', and 'TF 2.0 Alpha'. The main content area is divided into two columns. The left column is titled 'For beginners' and describes the Sequential API. The right column is titled 'For experts' and describes the subclassing API. Both columns include code snippets and 'Run code now' buttons. An illustration of a server rack is visible in the background of the right column.

TensorFlow makes it easy for beginners and experts to create machine learning models. See the sections below to get started.

[See the tutorials](#) [See the guide](#)

Tutorials show you how to use TensorFlow with complete, end-to-end examples. Guides explain the concepts and components of TensorFlow.

### For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the "Hello World" example below, then visit the [tutorials](#) to learn more.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

[Run code now](#) Try in Google's interactive notebook

### For experts

The subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the "Hello World" example below, then visit the [tutorials](#) to learn more.

```
class MyModel(tf.keras.Model):
    def __init__(self):
        super(MyModel, self).__init__()
        self.conv1 = Conv2D(32, 3, activation='relu')
        self.flatten = Flatten()
        self.d1 = Dense(128, activation='relu')
        self.d2 = Dense(10, activation='softmax')

    def call(self, x):
        x = self.conv1(x)
        x = self.flatten(x)
        x = self.d1(x)
        return self.d2(x)

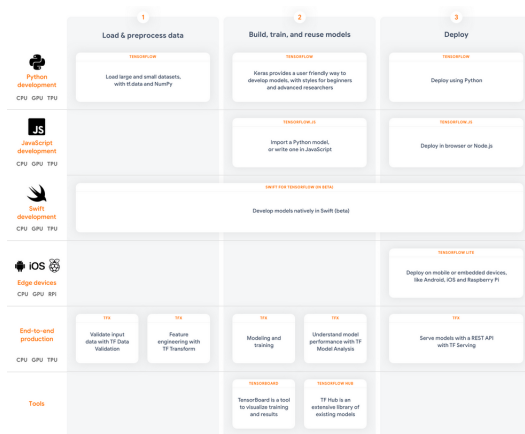
model = MyModel()

with tf.GradientTape() as tape:
    logits = model(images)
    loss_value = loss(logits, labels)
    grads = tape.gradient(loss_value, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

[Run code now](#) Try in Google's interactive notebook

# Tensor Flow

- Το TF παρέχει μια συλλογή workflows για την ανάπτυξη και εκπαίδευση μοντέλων που χρησιμοποιούν Python, JavaScript για εύκολη ανάπτυξη σε cloud, browser ή συσκευές (πχ RPi), ανεξάρτητα από τη γλώσσα που χρησιμοποιείτε.





# Tensor Flow

- Coral: quad-core NXP i.MX 8M system-on-chip paired with integrated GC7000 Lite Graphics, 1GB of LPDDR4 RAM, and 8GB of eMMC storage. It has wireless chip that supports Wi-Fi 2.4/5GHz and Bluetooth, a 3.5mm audio jack, and a full-size HDMI 2.0a port, plus USB 2.0/3.0 ports, a 40-pin GPIO expansion header, and a Gigabit Ethernet port (150\$).



# Virtual Sensors

- Μέσω 2 ΝΔ θα γίνεται εκτίμηση παραμέτρων  $\lambda$ ,  $\text{NO}_x$ , για έλεγχο κλειστού βρόχου.
- Γίνεται έτσι εκτίμηση παραμέτρων μηχανής παράλληλα με τα αισθητήρια (=Virtual Sensors).

**Development of recurrent neural networks for virtual sensing of  $\text{NO}_x$  emissions in internal combustion engines**

Ivan Arsis, Cesare Pianese, Marco Sorrentino  
Department of Industrial Engineering, University of Palermo

Copyright © 2009 Society of Automotive Engineers, Inc.

**ABSTRACT**

The paper focuses on the experimental identification and validation of recurrent neural networks (RNN) for virtual sensing of  $\text{NO}_x$  emissions in internal combustion engines (ICE). Exact learning procedures and experimental tests are proposed. To improve RNN prediction capabilities in predicting  $\text{NO}_x$  emission dynamics, the electrical control system (ECS) engine was tested in terms of an optimized system of hardware and software tools for engine and automation and control strategies prototyping. A fast response analyzer was used to measure  $\text{NO}_x$  emissions at the exhaust valve. The accuracy of the developed RNN model is assessed by comparing simulated and experimental responses for a wide range of operating scenarios. The results evidence that optimized virtual  $\text{NO}_x$  sensor act other significant opportunities for implementing on-board feedback and feedback control strategies aimed at improving the performance of after-treatment devices.

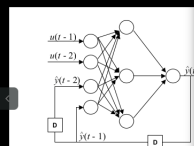
**INTRODUCTION**

Automotive engines and control systems are more and more sophisticated due to increasingly restrictive environmental regulations. Particularly in Diesel and ICE engines, complex after-treatment devices (e.g. SCR and DPF) have been introduced to meet the imposed  $\text{NO}_x$  and particulate emission limits. Furthermore, on-board diagnostics for both Diesel and spark ignition (SI) engines is becoming tighter and tighter in fact engine faults could lead to performance degradation, which may cause higher emissions. More recently, multivariable control systems are required to identify the actual fuel composition used to adjust the stoichiometric ratio.

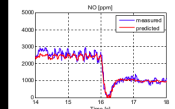
To cope with the new issues associated with novel complete hardware and to improve both conversion performance and after-treatment efficiency, engine design methods and control and diagnostic analyses have to be revised. A key role is played by diagnostic systems that are able to identify the engine operating condition. This is played by on-board implementation of dedicated models, either as predictors or emulators in model-based controllers or in diagnostic schemes. Virtual sensors (VS) can be used as substitutes of real sensors providing useful information about the actual process. In this scenario, virtual sensors are an effective alternative methodology to solve these challenging control and diagnostic topics.

In the last decade virtual sensors have been developed for many applications (e.g. emissions, drivetrain, electrical systems, environmental monitoring systems, to mention just few cases (Sorrentino, 1999)). VS can be especially useful when: (i) the direct measurement are impossible or costly, such as toxic hazardous chemical or nuclear plants (Guala and Pavia, 1992); (ii) the emissions are highly variable and non-linear, such as the starting characteristics (i.e. accuracy, dynamic performance) of the sensors; (iii) expensive sensors are not available in the operating location. It is also evident that VS can be used as feedback signal generators and their use provides opportunities to implement innovative control schemes and diagnostic. Actually, VS may be designed in such a way as to simulate time-dependent processes and guarantee accuracy, stability and short computational time. In this work it will be shown that the proposed virtual sensor also allows a limited recourse to experiments, thus reducing costs and developing time.

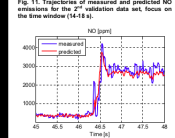
Automotive engines are an open system to be controlled with great precision and modeled with high accuracy. The modeling of engine control, monitoring and diagnostic algorithms must be fast enough to be implemented on-board (VS may support the development of new control and diagnostic systems by acting on the engine target, sensor monitoring and control as the model emulates both to be incorporated in an engine. This is mainly due to the complexity of simulating a process that exhibits advanced non-linearity and fast-dynamics within models and controllers. The state of the art mainly relies on hardware applications where the process knowledge is stored in a large experimental database implemented on maps. A successful attempt to develop a steady-state first



**Fig. 1 – Structure of NOE RNN with variable, one output, one hidden layer output delays D.**



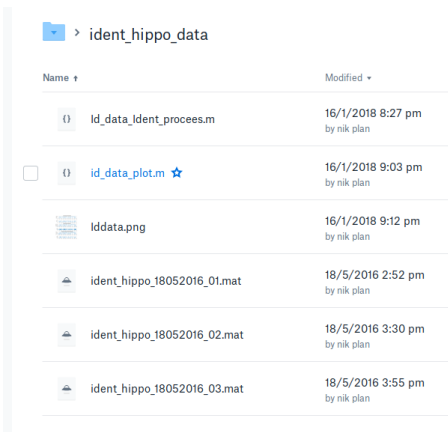
**Fig. 11. Trajectories of measured and predicted  $\text{NO}$  emissions for the 2<sup>nd</sup> validation data set, focus on the time window (14-16 s).**









**Fig. 12. Trajectories of measured and predicted  $\text{NO}$  emissions for the 12<sup>th</sup> validation data set, focus on the time window (45-46 s).**

# Δεδομένα για το ΝΔ

- Υπάρχουν διαθέσιμα στο Dropbox του μαθήματος (link στο [www.lme.ntua.gr](http://www.lme.ntua.gr)) τα δεδομένα για την σχεδίαση του ΝΔ.

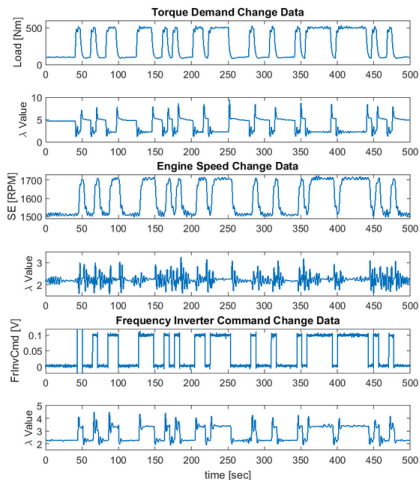


The screenshot shows a Dropbox interface for a folder named 'ident\_hippo\_data'. The files are listed in a table with columns for Name and Modified. The file 'id\_data\_plot.m' is highlighted with a blue star icon.

Name ↑	Modified ↓
 Id_data_Ident_procees.m	16/1/2018 8:27 pm by nik plan
<input type="checkbox"/>  id_data_plot.m ☆	16/1/2018 9:03 pm by nik plan
 lddata.png	16/1/2018 9:12 pm by nik plan
 ident_hippo_18052016_01.mat	18/5/2016 2:52 pm by nik plan
 ident_hippo_18052016_02.mat	18/5/2016 3:30 pm by nik plan
 ident_hippo_18052016_03.mat	18/5/2016 3:55 pm by nik plan

## Δεδομένα για το ΝΔ -2

- Με τα δεδομένα αυτά θα γίνει η σχεδίαση ΝΔ για εκτίμηση των παραμέτρων  $\lambda$  και  $\text{NO}_x$  της μηχανής.



## Δεδομένα για το ΝΔ -2

- Στο ίδιο link θα βρείτε τις 3 βιβλιογραφικές αναφορές (papers) για την σχεδίαση των ΝΔ (Arsie et al.)

# TO DO !

Μελετήστε από το:

Neural Networks: A Comprehensive Foundation, Simon Haykin,  
Macmillan, 1994

το κέφ. 1.