CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING

MASTER'S THESIS

Neural Network Methods for Nonlinear Dynamical Systems with Sinusoidal Nonlinearity

2018

MOHAMED G. A. GHITA

SUPERVISOR: DOC. ING. IVO BUKOVSKÝ, PH.D.

Annotation List

Author name:	Mohamed G. A. Ghita
Name of Master's Thesis:	Neural Network Methods for Nonlinear Dynamical Systems with Sinusoidal Nonlinearity
Year:	2018
Field of Study:	Instrumentation and Control Engineering
Department:	Department of Instrumentation and Control Engineering
Supervisor:	Doc. Ing. Ivo Bukovský, Ph.D.
Bibliographic data:	Number of pages 50 Number of figures 31 Number of tables 1 Number of attachments 1
Keywords	HONU, recurrent HONU, dynamic HONU, static HONU, Neural Networks, Supervised learning, complex-valued, CV-HONU, Adam Optimizer, Gradient Descent, Lavenburg Marquardt, Conjugate Gradient, Pendulum, manipulator, DOF, Python, sinusoidal nonlinearity

Statement

I declare that I have worked on this thesis independently assuming that the results of thesis can be also used at discretion of the supervisor of the thesis as its co-author. I also agree with the potential of publication of the results of the thesis or its substantial part, provided I will be listed as the co-author.

In Prague _____ Signature: _____

Acknowledgment

My thesis supervisor ***DOC. ING. IVO BUKOVSKÝ*** kept his door open whenever I ran into trouble or needed help. The first time, I showed him a piece of code, he took my laptop and helped debugging it. I would like to thank you specially for the time you spent on improving my writing.

I finished my thesis and master studies while working full-time and I would not have been able to continue my studies or pass stressful moments without the support of my colleagues and managers in ***Radalytica s.r.o.*** and ***Advacam s.r.o.***. Thank you all!

Abstract

First, the thesis reviews the recent work in adaptive approximation of nonlinear dynamic systems with sinusoidal nonlinearities, including fundamental approaches of high-order neural units (HONU) and their supervised learning techniques. Two different learning techniques are studied: sample-by-sample and batch learning adaptation. Then, the thesis proposes and compares the performance of general neural architectures with complex-valued neural architectures for approximation of the given nonlinear systems (with sinusoidal nonlinearities). Furthermore, the work develops the learning rules of the newly proposed complex-valued high-order neural units. Finally, a software toolbox is developed for all different neural architectures and dynamic systems, analyzed in this thesis. The toolbox is used to generate all the simulation results during the thesis.

Table of Contents

List of Figures

List of Tables

1 Introduction

The number of robotics applications in industry and research laboratories is increasing as industrial robots are becoming more reliable [1, 2]. However, more effort is needed to integrate other technologies with robotics. Myself, I have been recently involved in development of industrial x-ray imaging robotic system[1] ,Figure 1, that uses several robots, and I would like to introduce my thesis by drawing the connotation between the topic of my thesis and the current needs in robotic applications according to my recent experience.

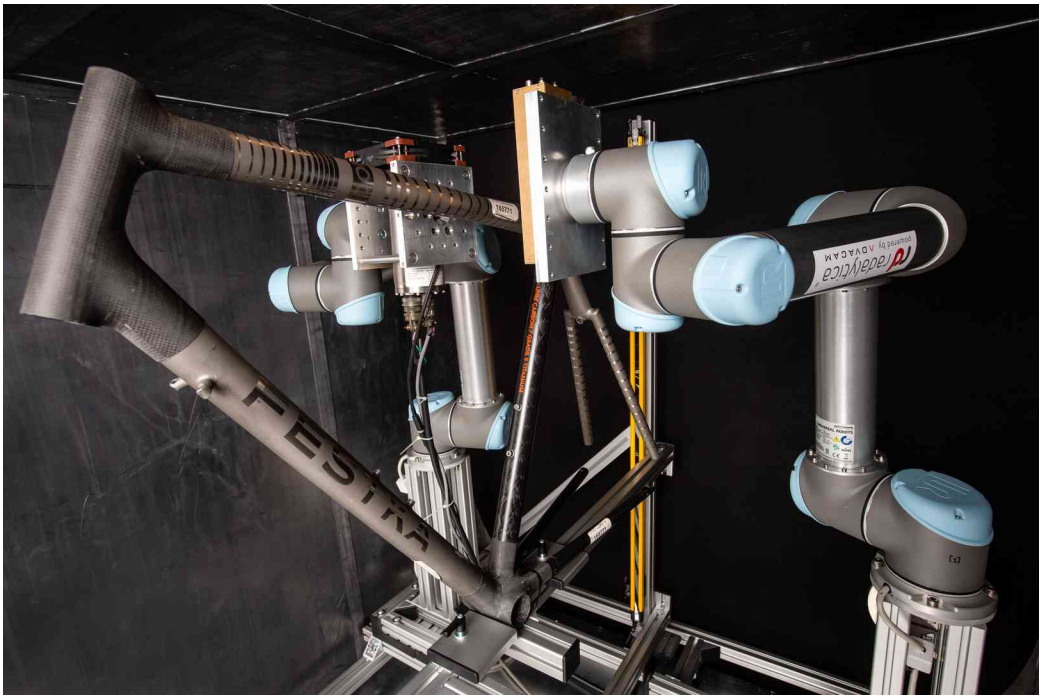

Figure 1: Dynamic system with sinusoidal nonlinearity and varying payload. The robots switch between different tools. Also, the robot on the left carries x-ray tube with a high voltage cable which changes the payload significantly during the motion. X-Ray Imaging Robotic System (credit Radalytica s.r.o.)[1]

Considering the x-ray robotic systems Figure 1; however not limiting to, one of the manipulators switches between several tools which have different payloads. In some cases, the operator can set the payload or center of mass of these tools inaccurately which slightly increases the manipulator oscillation. Also, the x-ray tube is connected to a high voltage cable which changes the payload mass and center of mass significantly during the robot motion. Then for different payloads, the robot control is not optimal due to unknown or varying payload, which raises the need for data-driven (adaptive) approaches.

The neural network approaches appear as suitable tools for the task of data-driven system identification and potential control. Among these approaches, polynomial neural networks and higher-order neural neural networks, [3–6] that represent nonlinear systems with interesting property of being linear in parameters [7]. Specific research on HONU has been carried out at CTU

1 Radalytica s.r.o., http://www.radalytica.com

in Prague [8–13] . However, there are limitations of these polynomial architectures for their practical use for systems with stronger nonlinearity, such as inverted pendulum [13, 14] that is a typical system with sinusoidal nonlinearity and it is also a typical nonlinearity in robotic dynamics.

Furthermore, there is a quite new trend in signal processing with adaptive filters and neural networks that is potentially suitable to approximate sinusoidal systems, i.e. the complex valued filters and complex valued neural networks [15, 16]. They became recently studied also for approximating dynamical systems [17]; however, this seems to be a rather new topic.

This thesis is an effort to review the neural network approaches for dynamic systems with sinusoidal nonlinearity. Also, Chapter 5 investigates the design and potential use of complex valued HONUs for model approximation from measured data. The purpose is a later robotic system performance optimization, e.g. via identifying the unknown payload or adaptive control optimization (please note, that neither the payload identification nor control are the subject of this thesis and they will be matter of future research)

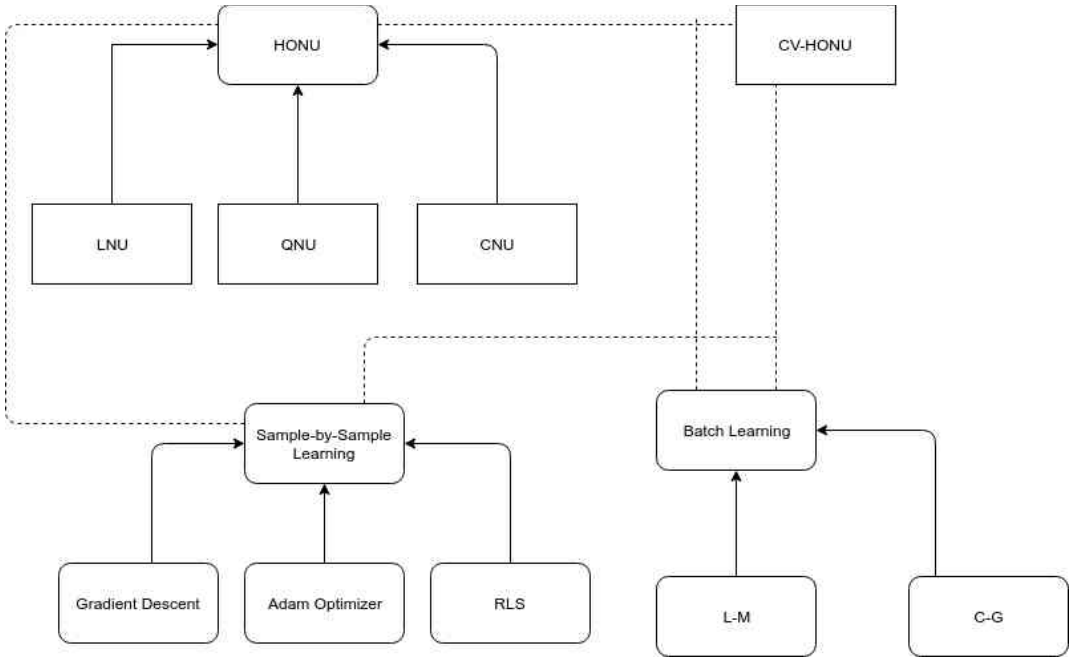

Figure 2: HONU SW toolbox

The study also includes the development of a software toolbox Figure 2 for HONUs, complex-valued HONUs and their learning approaches. The developed software toolbox has two main purposes. First, the toolbox help better understanding HONUs by visualizing their learning process and simulating the output. Secondly, a well-designed toolbox is the mean to integrate this study results with a production level robotic software. Python is a powerful programming language for scientific research and machine learning but, in robotics, usually C++ is preferred in production. However, this study uses Python because it is still in early stages of research. Also, the development of such toolbox is essential to make the results of this work easily used by other colleagues.

Chapter 2 reviews real-valued HONUs and their neuron architecture to achieve different orders. The source of HONUs training and testing data is simulation of several mathematical models, derived in chapter 3. Chapter 4 studies different HONUs learning techniques. The chapter includes two fundamental categories of learning techniques: sample-by-sample and batch learning. Chapter 5 is dedicated to develop a new complex-valued HONU architecture and test its performance in approximating a dynamic system with sinusoidal nonlinearity. Finally, chapter 6 explains the structure of the developed software toolbox for HONU. The conclusion is written in last chapter.

2 Review of Higher Order Neural Units – HONU

Neural Networks is used by connectionists in the field of machine learning to model numerous systems. Connectionists try to model mental phenomena using neural networks. Also, the use of neural networks span vast applications in different fields such as physics, medicine, robotics, economics, control systems [13] etc. The basic building block of neural networks is an artificial neuron. Artificial neurons mimics approximate model of biological neurons, Figure 3 [2].

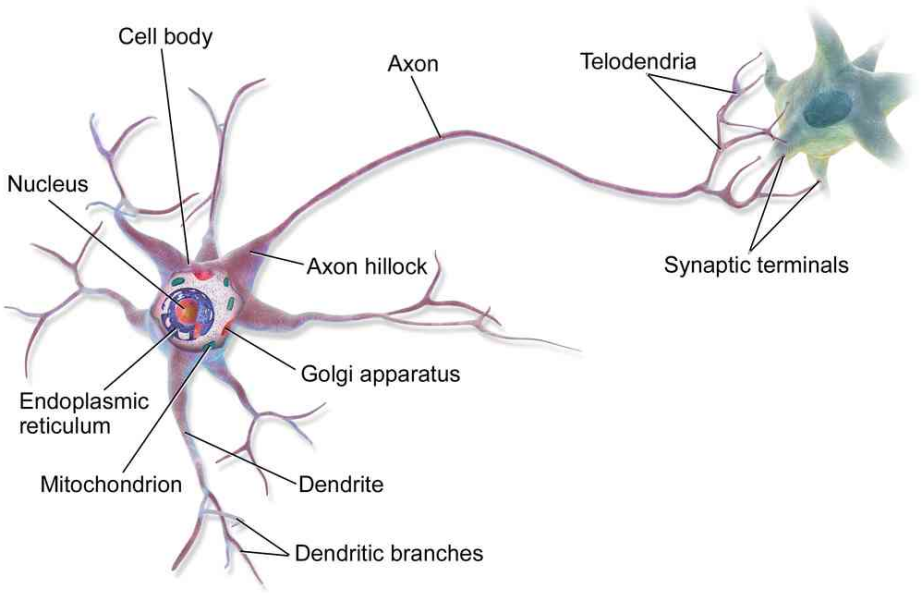

Figure 3: Biological Neuron (https://en.wikipedia.org/wiki/Neuron)[2]*

Artificial neurons model receives several inputs $\mathbf{x}$ and sum them after assigning weights $\mathbf{w}$ for each input (synaptic operation). The weighted sum passes by an activation or transfer function $\phi(.)$ which usually has a sigmoid shape. Equations (1), (2) and (3) summarizes the artificial neuron model as follows

$$\mathbf{x} = [x_0 = 1, x_1, x_2, ..., x_n]^T, \tag{1}$$

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]^T, \tag{2}$$

$$y = \phi(\mathbf{w} \cdot \mathbf{x}). \tag{3}$$

2* By BruceBlaus [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)], from Wikimedia Commons

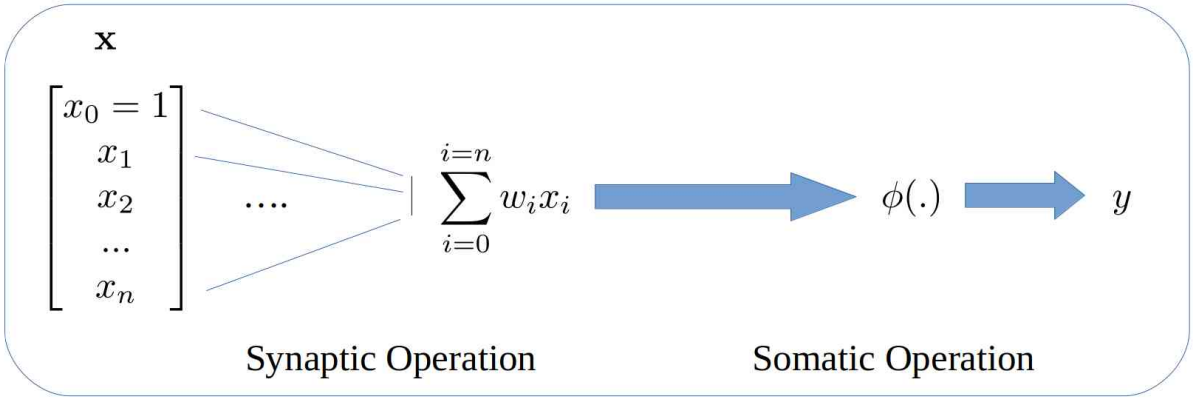

Figure 4: Artificial Neuron

Higher order neural units (HONUs) is a class of neural networks where neurons involve polynomials. HONU aggregates the inputs in higher order polynomial terms during the synaptic operation. These polynomial terms help HONU to learn and respond to the relations between the inputs as explained in [6]. This thesis uses HONU representation in [18] which is explained as follows

$$y = \mathbf{w} \cdot col^r(\mathbf{x}), \tag{4}$$

where $col^r(\mathbf{x})$ is flattening operation that aggregates the inputs in terms of order r. For example, a linear neuron unit (LNU) where $r = 1$ is as follows

$$y = \sum_{i=0}^{i=n} x_i w_i = \mathbf{w} \cdot col^1(\mathbf{x}), \tag{5}$$

, and cubic neuron unit (CNU) where $r = 3$ is as follows

$$y = \sum_{\substack{i=0 \dots n_x \\ j=i \dots n_x \\ k=j \dots n_x}} x_i \cdot x_j \cdot x_k \cdot w_{i,j,k} = \mathbf{w} \cdot col^3(\mathbf{x}), \tag{6}$$

the general HONU representation [18] is as follows

$$y = \sum_{\substack{i=0 \dots n_x \\ j=i \dots n_x \\ k=j \dots n_x \\ \dots}} x_i \cdot x_j \cdot x_k \dots \cdot w_{i,j,k \dots} = \mathbf{w} \cdot col^r(\mathbf{x}). \tag{7}$$

If n is the length of input vector $\mathbf{x}$ and r is HONU order, then the length l of the flattened vector $col^r(\mathbf{x})$ is defined by equation (8) as combination with repetitions as follows

$$l = \frac{(n+r-1)!}{(r)!(n-1)!}\,. \tag{8}$$

Some of HONUs applications are identification and control of dynamic systems [4] and [20]. Equation (9) and (10) represent a general dynamic system or (11) and (12) for a discrete system as follows

$$\frac{d\mathbf{s}}{dt} = f(\mathbf{s}, \mathbf{u}), \tag{9}$$

$$yr = g(\mathbf{s}, \mathbf{u}), \tag{10}$$

where $\mathbf{s}, \mathbf{u}, yr$ are the system states, inputs and output. Using forward Euler method with sampling interval Δt to simulate it as a discrete system

$$\mathbf{s}_{k+1} = \mathbf{s}_k + \Delta t f(\mathbf{s}_k, \mathbf{u}_k), \tag{11}$$

$$yr_{k+1} = g(\mathbf{s}_{k+1}, \mathbf{u}_{k+1}). \tag{12}$$

Recurrent or dynamic HONU is used to identify dynamic system where some of the HONU inputs have feedback from HONU output. On the other hand, Static HONU is a feedforward unit without feedback and is used to predict only one sample ahead of the dynamic system. The architectures of both recurrent and static HONUs are explained in the following subsections.

2.1 Static HONU

Static HONU is used to predict the output yr_{k+1} of a dynamic system as in equations (11) and (12). For prediction, HONU input vector $\mathbf{x}$ contains previous measurements of the dynamic system output yr_k and input u_k as in equation (13) or equation (14) if the system has one input $\mathbf{u} = [u]$ as follows

$$\mathbf{x} = [1, yr_k, yr_{k-1}, ..., yr_{k-n_y}, \mathbf{u}_k^T, \mathbf{u}_{k-1}^T, ..., \mathbf{u}_{k-n_u}^T]^T, \tag{13}$$

$$\mathbf{x} = [1, yr_k, yr_{k-1}, ..., yr_{k-n_y}, u_k, u_{k-1}, ..., u_{k-n_u}]^T. \tag{14}$$

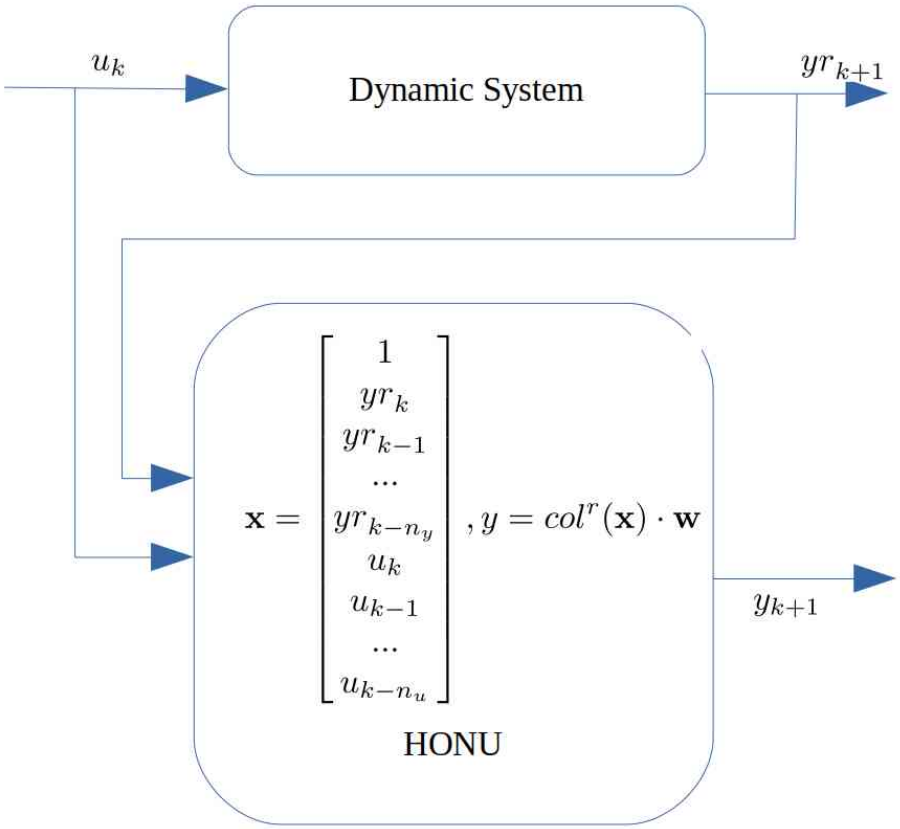

Figure 5: Static HONU for dynamic system prediction

2.2 Dynamic HONU

Dynamic HONU is a standalone neuron that can be fed only with the same input of the dynamic system. Therefore, it is supposed to produce the same output as the dynamic system during all the simulation or testing time without watching the dynamic system output. For HONU to mimic the dynamic system behavior, it needs internal feedback from its own output as shown in Figure 6 and the following equation (15)

$$\mathbf{x} = [1, y_k, y_{k-1}, ..., y_{k-n_y}, u_k, u_{k-1}, ..., u_{k-n_u}]^T. \tag{15}$$

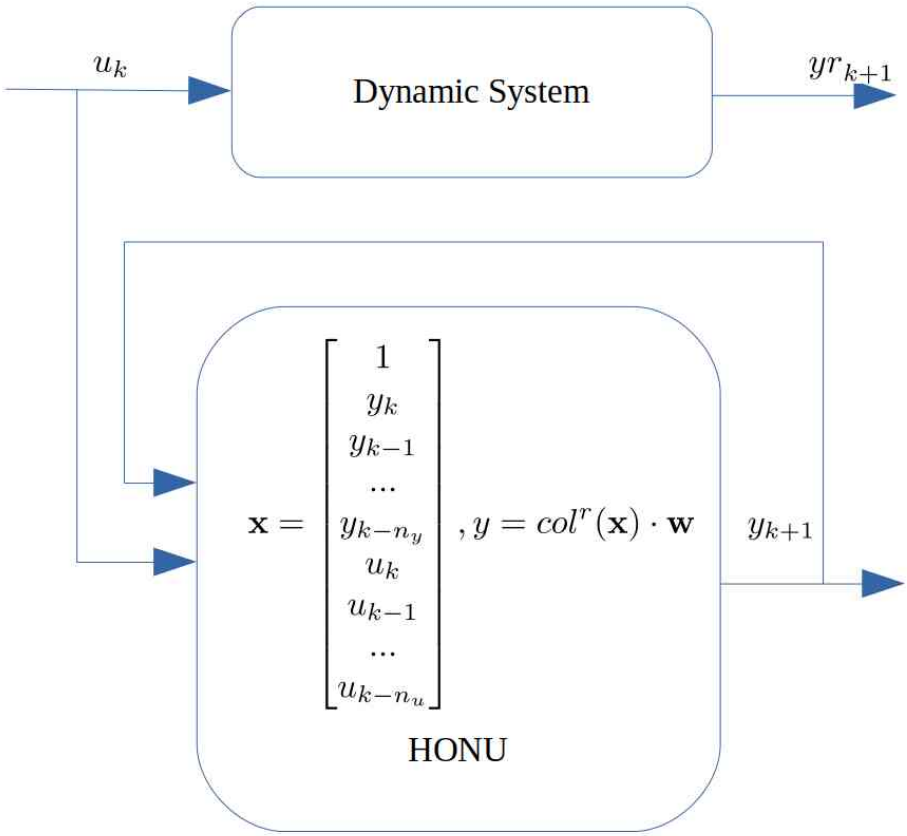

Figure 6: Recurrent HONU for dynamic system identification

3 Mathematical Models

The derived mathematical models in this chapter produce training data as well as test data to train and test HONUs. The mathematical models are a torsional pendulum, 2-DOF manipulator and complex-valued electric impedance. The first two models represent systems with sinusoidal nonlinearity. The last model is a simple complex-valued impedance that responds to temperature differences.

3.1 1-DOF Torsional Pendulum Model

Torsional pendulum, Figure 7, is a classical dynamic system with sinusoidal nonlinearity. The studied pendulum is a concentrated mass at the end of one link which connects the mass with a joint. In the joint, there is a motor which is assumed to provide variable torque without any delay. Also, there is a damping factor in the joint.

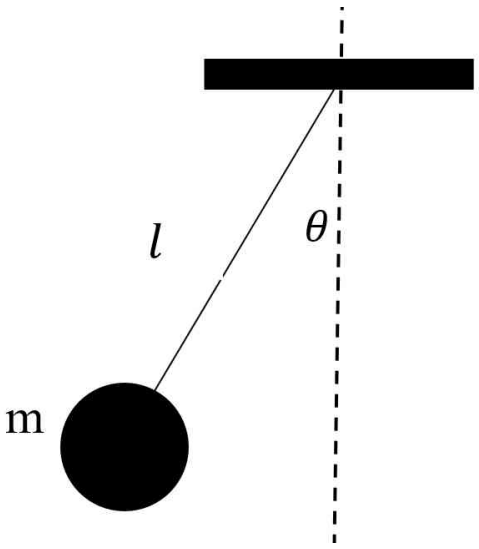

Figure 7: Torsional pendulum sketch

The state vector $\mathbf{s}$ is the angular position θ and speed ω of the joint while the output y is the angular position as in the following equations (17), (16) and (18)

$$\mathbf{s} = [\theta, \omega]^T, \tag{16}$$

$$y = [1, 0]\ \mathbf{s} = \theta. \tag{17}$$

The pendulum equation of motion is

$$s' = f(s, u) = \begin{bmatrix} \omega \\ \frac{1}{J}(u - m\ g\ l\ sin(\theta) - fd\ \omega) \end{bmatrix}, \tag{18}$$

where the mass $m = 1/3\ kgM = 1/3\ kg$, the length of the pendulum bar $l = 2/3\ m$ and the gravity acceleration $g = 9.81\ m/s^2$. Then, the pendulum moment of inertia $J = 4/3\ M\ l^2$. The frictional factor $fd = 0.6$. Another extension to the system is a P controller over the angular position,

$$u = kp(\theta_d - y). \tag{19}$$

3.2 Planar 2-DOF Manipulator Model

One of the objectives of this work is to examine the performance of both static and dynamic HONU in prediction and identification of dynamic systems with sinusoidal non-linearities. The second extension to a torsional pendulum (Section 3.1) is a 2-DOF manipulator, Figure 8. The manipulator moves in one plane by rotation of two angular joints.

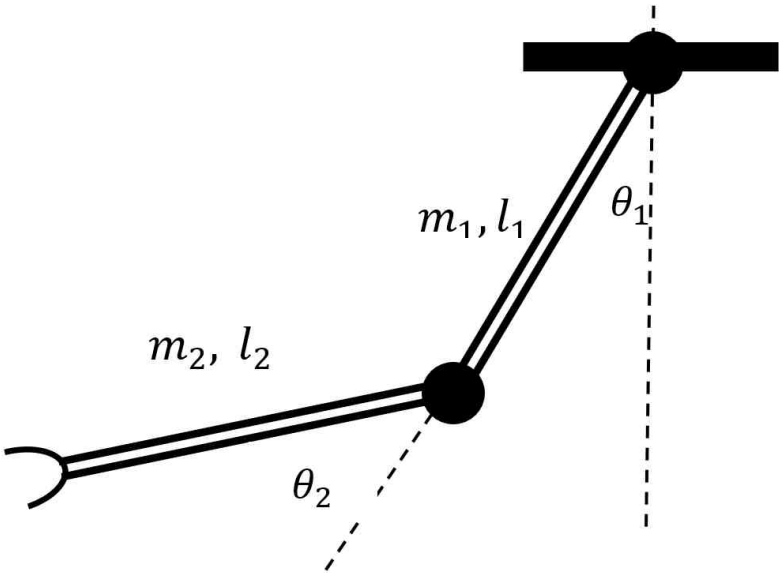

Figure 8: Sketch of a planar 2-DOF manipulator

3.2.1 Mechanics

The following bullet points summarize the steps to derive the mathematical model of such system using Lagrangian mechanics, following [21].

- Dynamic System definitions
  states: $s = [\theta_1, \theta_2, \theta_1', \theta_2']^T$
  Inputs: $\tau = [\tau_1, \tau_2]^T$
  Outputs: $y = [\theta_1, \theta_2]^T$
- Definition of parameters and symbols
  m_i is the mass of link i

l_i is the length of link i
I_i is the moment of inertia of link i
g is the gravity acceleration
$g_v = [0 \ -1]^T \ g$
$c_1 = cos(\theta_1 + \pi), \ c_2 = cos(\theta_2)$
$s_1 = sin(\theta_1 + \pi), \ s_2 = sin(\theta_2)$
$c_{12} = sin(\theta_1 + \pi + \theta_2), \ s_{12} = sin(\theta_1 + \pi + \theta_2)$

- Kinematics
  1. First Link:
     Angular Velocity
     $$\omega_1 = \theta_1', \tag{20}$$
     linear Velocity of center of mass
     $$\boldsymbol{v_1} = l_1/2 \ \omega_1[-c_1, -s_1]^T, \tag{21}$$
     position of center of mass
     $$\boldsymbol{p_1} = l/2[-s_1, c_1]^T. \tag{22}$$
  2. Second Link:
     Angular Velocity
     $$\omega_2 = \omega_1 + \theta_2' \, , \tag{23}$$
     linear Velocity of center of mass
     $$\boldsymbol{v_2} = 2\boldsymbol{v_1} + l_2/2 \ \omega_2[-c_{12}, -s_{12}]^T, \tag{24}$$
     position of center of mass
     $$\boldsymbol{p_2} = \boldsymbol{p_1} + l/2[-s_{12}, c_{12}]^T \tag{25}$$

- Total Kinetic Energy
$$T = 1/2 \ m_1 \boldsymbol{v_1}^T \boldsymbol{v_1} + 1/2 \ \omega_1 I_1 \omega_1 + 1/2 \ m_2 \boldsymbol{v_2}^T \boldsymbol{v_2} + 1/2 \ \omega_2 I_2 \omega_2 \, . \tag{26}$$

- Total Potential Energy
$$U = -m_1 \boldsymbol{g_v}^T \boldsymbol{p_1} - m_2 \boldsymbol{g_v}^T \boldsymbol{p_2} \, . \tag{27}$$
- Lagrange Dynamics
$$L = T - U, \tag{28}$$

$$\tau_1 = \frac{\partial L}{\partial \theta_1} - \frac{d}{dt}[\frac{\partial L}{\partial \theta_1'}], \tag{29}$$

$$\tau_2 = \frac{\partial L}{\partial \theta_2} - \frac{d}{dt}[\frac{\partial L}{\partial \theta_2'}]. \tag{30}$$

- Finally the torques $\boldsymbol{\tau}$ yield

$$\boldsymbol{\tau} = M(\boldsymbol{\theta}) \begin{bmatrix} \theta_1'' \\ \theta_2'' \end{bmatrix} + V(\boldsymbol{\theta}, \boldsymbol{\theta'}) + G(\boldsymbol{\theta}), \tag{31}$$

where M, the Mass Matrix and V, centrifugal and Coriolis effects, and G, Gravity Effect are defined in equations (32), (33) and (34) as follows

$$M(\theta) = \begin{bmatrix} 0.25m_1 l_1^2 + m_2 l_1^2 + c_2 l_2 m_2 l_1 + I_1 + I_2 + 0.25 l_2^2 m_2 & I_2 + 0.25 l_2 (2c_2 l_1 + l_2) m_2 \\ I_2 + 0.25 l_2 (2c_2 l_1 + l_2) m_2 & 0.25 m_2 l_2^2 + I_2 \end{bmatrix}, \tag{32}$$

$$V(\theta, \theta') = \begin{bmatrix} -s_2 l_1 l_2 m_2 \theta_2' & -0.5 s_2 l_1 l_2 m_2 \theta_2' \\ 0.5 s_2 l_1 l_2 m_2 \theta_1' & 0 \end{bmatrix}, \tag{33}$$

$$G(\theta) = \begin{bmatrix} -0.5g(s_{12} l_2 m_2 + s_1 l_1 (m_1 + 2m_2)) \\ -0.5 g s_{12} l_2 m_2 \end{bmatrix}. \tag{34}$$

3.2.2 Modeling Motors Inertia and Friction

The motor gearbox connects the torque τ_{mi} of motor i to the dynamics torque τ_i as follows

$$\tau_{mi} = \eta_i^{-1} \tau_i + I_{mi} \theta_{m_i}'' + f_{mi} \theta_{m_i}' , \tag{35}$$

as

$$\theta_{mi} = \eta_i \theta_i , \tag{36}$$

then

$$\tau_{mi} \eta_i = \tau_i + I_{mi} \eta_i^2 \theta_i'' + f d_m \eta_i^2 \theta_i' , \tag{37}$$

finally

$$\begin{bmatrix} \tau_{m1}\eta_1 \\ \tau_{m2}\eta_2 \end{bmatrix} = \left(\mathrm{M}(\boldsymbol{\theta}) + \begin{bmatrix} I_{m1}\eta_1^2 & 0 \\ 0 & I_{m2}\eta_2^2 \end{bmatrix} \right) \begin{bmatrix} \theta_1'' \\ \theta_2'' \end{bmatrix} + (\mathrm{V}(\boldsymbol{\theta}, \boldsymbol{\theta'}) + \begin{bmatrix} f_{m1}\eta_1^2 & 0 \\ 0 & f_{m2}\eta_2^2 \end{bmatrix} \begin{bmatrix} \theta_1' \\ \theta_2' \end{bmatrix}) + \mathrm{G}(\boldsymbol{\theta}) \tag{38}$$

Either equation (31) (without gearboxes) or (38) (with gearbox) can be used to simulate the planar 2-DOF manipulator.

3.3 Electric Impedance

It is useful to represent electric impedance as a complex number in studying AC circuit. The Impedance is defined as

$$z = r + jx, \tag{39}$$

where $j^2 = -1$.

The complex-valued impedance responds to temperature differences as shown in the following equation

$$z = z_{nom} + \alpha(T - 20)(\frac{k_1}{\tau_1 s + 1} + j\frac{k_2}{\tau_2 s + 1}), \tag{40}$$

where α is heat coefficient, z_{nom} is nominal impedance at temperature $T = 20\ ^\circ\mathrm{C}$.

4 Supervised Learning of HONU

This chapter trains real-valued HONUs, described in chapter 2, to learn different dynamic systems whose mathematical models are derived in chapter 3. Further, the trained HONUs are simulated to test their approximation performance of dynamic systems with sinusoidal nonlinearity.

Supervised learning trains HONU using a set of training data of inputs and outputs. After training, HONU is tested on another test data set which examines if the model scales to the new data and does not overfit the training data. The next two subsections show two main approaches: sample-by-sample and batch training. Sample-by-sample learning is preferable to perform online learning during the dynamic system operation. On the other hand, batch learning works for offline training when set of training data is already available.

4.1 Sample by Sample Learning

4.1.1 Gradient Descent

Gradient descent update HONU weights for every new sample such that certain cost function is minimized. In other words, the new weights every sample time moves downhill the cost function with certain learning rate. The learning process stability depends on the magnitude of inputs and outputs as well as the learning rate. The following equations are derived for cost function e_k as follows

$$\mathbf{w} \rightarrow \mathbf{w} + \Delta\mathbf{w}, \tag{41}$$

$$e_k = \frac{1}{2}(yr_k - y_k)^2, \tag{42}$$

$$\Delta\mathbf{w} = -\mu \frac{\partial e_k}{\partial y} \frac{\partial y}{\partial \mathbf{w}} = \mu \cdot (yr_k - y_k) \cdot col^r(\mathbf{x}_k). \tag{43}$$

From equation (43) The incremental learning of weights $\mathbf{w}$ depends on the real system inputs and outputs which makes setting the learning rate more difficult. To cope with this issue, a normalized learning rate is defined as follows

$$\Delta\mathbf{w} = \frac{\mu}{1 + ||col^r(\mathbf{x}_k)||_2^2} (\mathrm{yr}_k - y_k) col^r(\mathbf{x}_k). \tag{44}$$

4.1.2 Recursive Least Squares RLS

RLS is a recursive algorithm that minimize the sum of the squared error between HONU and the real system. The fundamental difference between RLS and gradient-descent is the cost function as RLS is sum of the error till the current sample while the gradient-descent is only the current error, (42) vs (45)

$$e_k = \sum_{k=0}^{k=m} \lambda^{m_i} \cdot (yr_k - y_k)^2, \tag{45}$$

where $\lambda \in]0, 1]$ is a forgetting factor. If $\lambda = 1$, then all the previous samples are treated equally to the new sample but if $\lambda < 1$, then the older samples of data has exponentially less weight in the the cost function e_k. The summary of RLS equations is as follow

$$R(0) = \frac{1}{\delta} I, \tag{46}$$

where $\delta, I$are a small positive number and an identity matrix.

$$R(k) = \frac{1}{\mu}(R(k-1) - \frac{R(k-1)col^r(\mathbf{x}_k) \cdot col^r(\mathbf{x}_k)^T R(k-1)}{\mu + col^r(\mathbf{x}_k)^T R(k-1) col^r(\mathbf{x}_k)}), \tag{47}$$

then, $R(k)$ is used to update $\mathbf{w}$ as follows

$$\Delta \mathbf{w} = R(k) col^r(\mathbf{x}_k)(yr_k - y_k). \tag{48}$$

4.1.3 Adam Optimizer

Adam optimizer [22] is an iterative algorithm to update neural networks weights. The authors in [22] introduced Adam optimizer for non-convex problems instead of stochastic gradient descent. Also, Adam optimizer weights update is invariant to scaling the gradient. Next section compares Adam with Normalized gradient-descent. The highlights of Adam optimizer are as follows:

- The cost function can change with time and Adam still converges.
- Scaling the cost function or the gradient does not affect the weights update.
- Varies the step size which is very helpful to use the same parameters to wide range of problems.

Adam has the following parameters

- β_1, β_2 exponential decay rates for the first and second gradient moments respectively.
- α The learning rate
- $\eta = 10^{-8}$ a small number to prevent division by zero.

The Algorithm steps are as follows

- Initialize $m_0 = 0, v_0 = 0, t = 0$
- While $t < simulationtime$
  - gradient g
  - $t = t + 1$
  - $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g$
  - $v_t = \beta_1 v_{t-1} + (1 - \beta_1)g^2$
  - $\widehat{m_t} = m_t/(1 - \beta_1^t)$
  - $\widehat{v_t} = v_t/(1 - \beta_2^t)$
  - $w = w - \alpha\widehat{m_t}/(\sqrt{\widehat{v_t}} + \eta)$

4.2 Sample-by-sample Results

The real dynamic system is an oscillating second order system with sampling interval $\Delta t = 0.5$ as follows

$$yr_{k+1} = [0, 1.305, -0.6049, 0.1628, 0.1374] \cdot [1, yr_k, yr_{k-1}, u_k, u_{k-1}] \quad . \tag{49}$$

The first step is to adapt the HONU online with sample-by-sample algorithm. ,……. shows the online sample-by-sample learning. The dynamic HONU architecture is as follows

$$y_{k+1} = \mathbf{w} \cdot col^r(\mathbf{x}_k), r = 1, \tag{50}$$

$$\mathbf{x}_k = [1, y_k, y_{k-1}, ..., y_{k-n_y+1}, u_k, u_{k-1}, ..., u_{k-n_u+1}]^T, n_y = n_u = 6. \tag{51}$$

Figure 9, Figure 10 and Figure 11 show the online sample-by-sample of the system using normalized gradient descent and Adam optimizer. But Figure 12 shows the test data after turning off the learning and using different inputs. The results shows that the adapted HONU responds to different input steps in the test data.

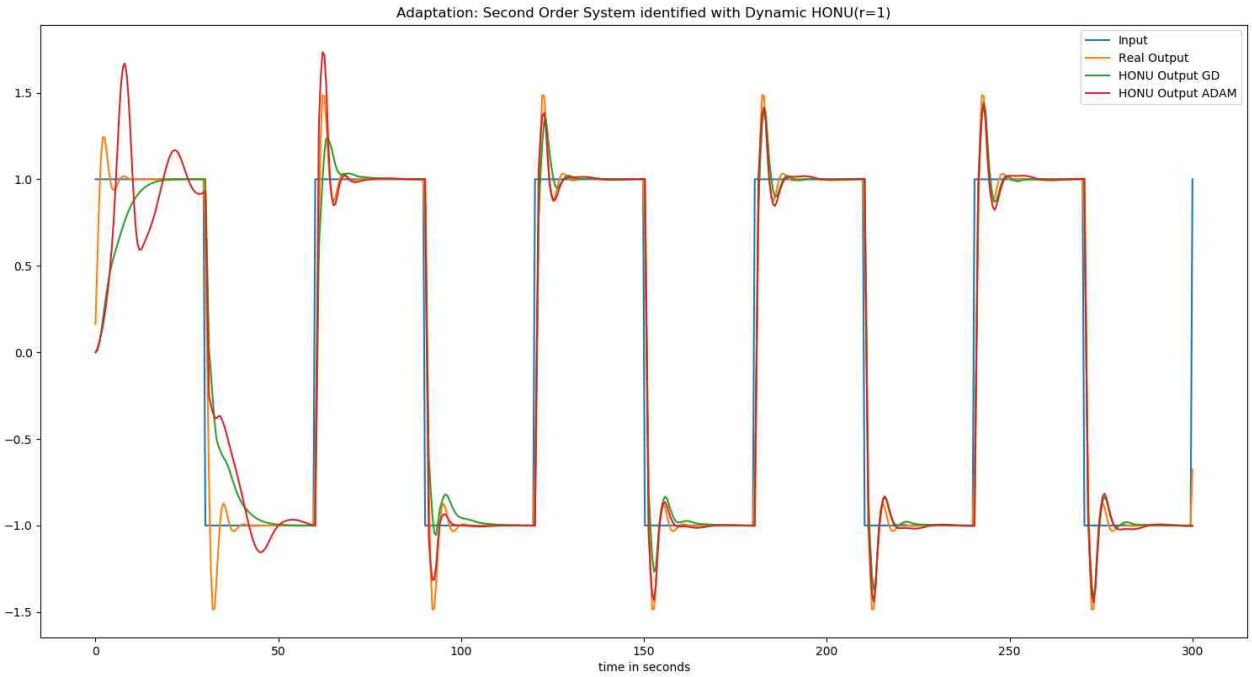

Figure 9: Online learning sample-by-sample of oscillating system.
The comparison between normalized GD and Adam optimizer shows that Adam optimizer learns faster and its LNU responds earlier to the oscillation.

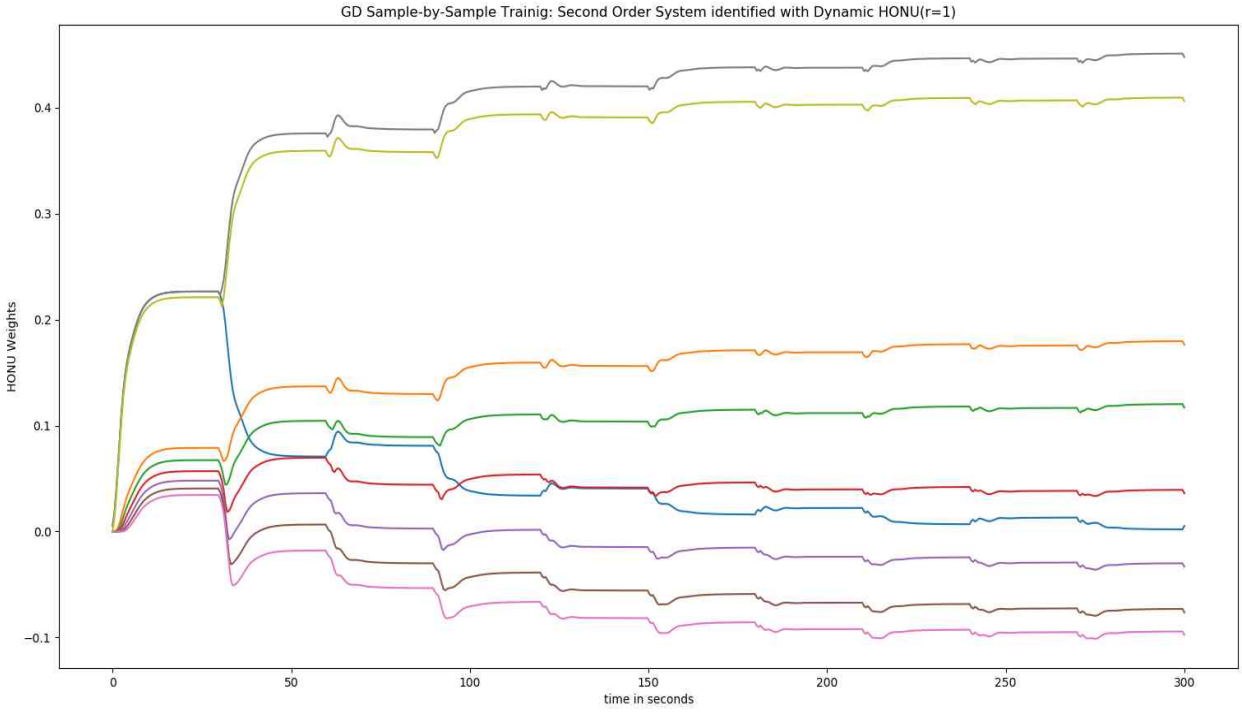

Figure 10: Online learning sample-by-sample using normalized gradient descent.

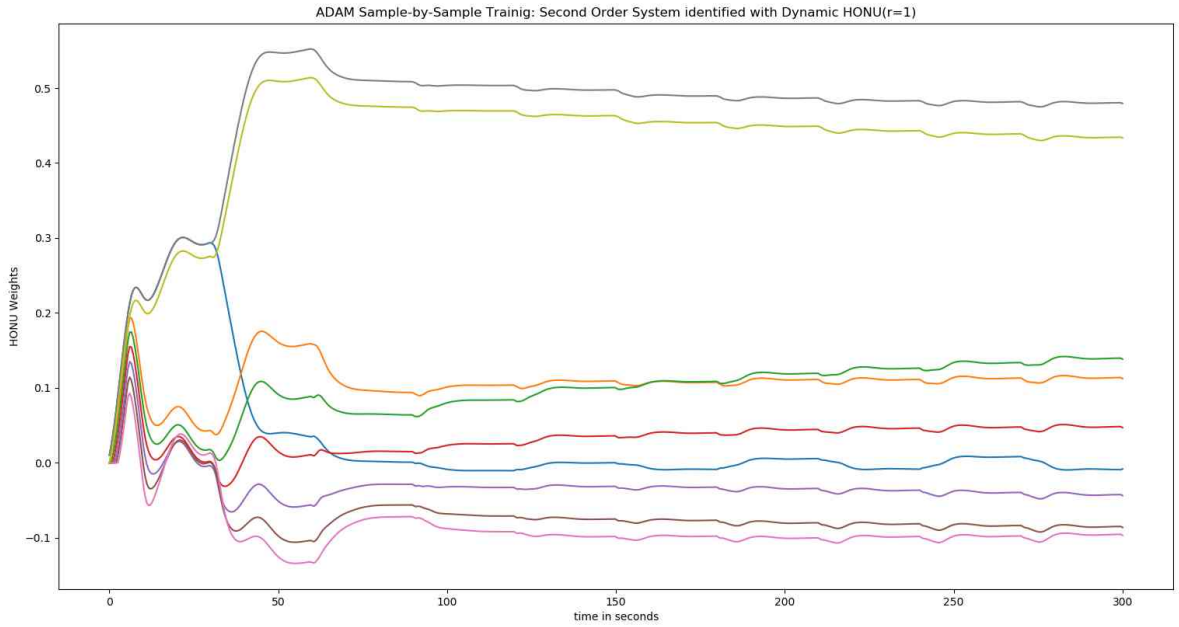

Figure 11: Online learning sample-by-sample using Adam optimizer

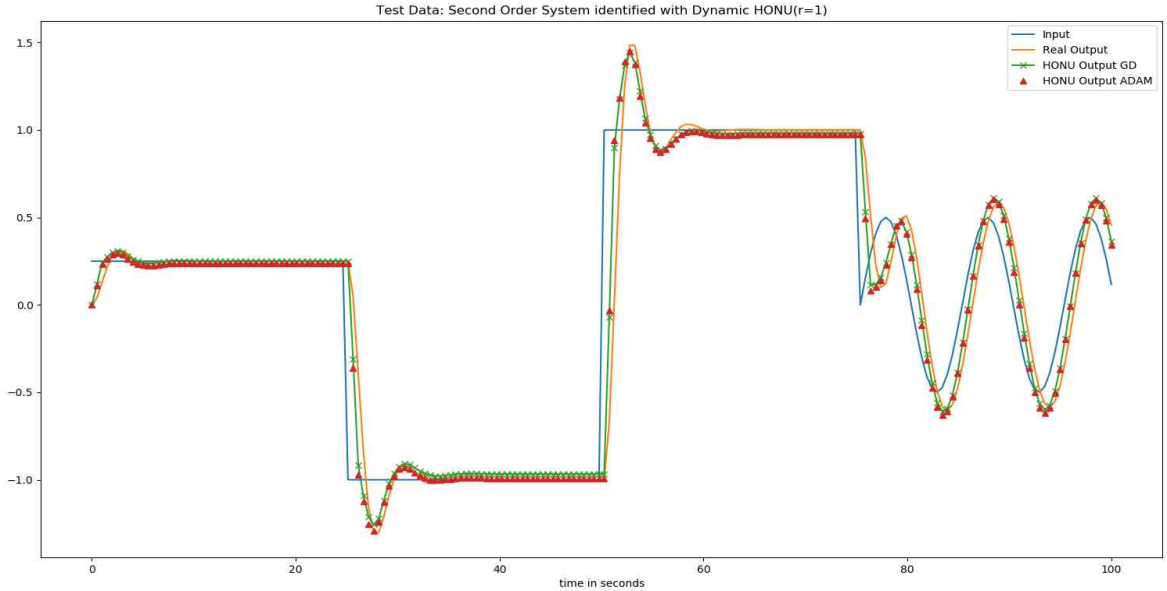

Figure 12: Test data after turning off the sample-by-sample online learning

4.3 Batch Learning

There are several batch learning algorithms for HONU. Levenberg-Marqaurdt L-M and conjugate gradient CG are described with all implementation details for HONU in [13] and [23]. It is noticeable that CG usually accelerates the learning of HONU but after starting using L-M for the first training epochs.

4.3.1 Levenberg-Marquardt - L-M

Levenberg-Marquardt algorithm for HONU [13] batch learning depends of Jacobian Matrix J which is the gradient of HONU output to the weights at every time step in training data as follows

$$J = \begin{bmatrix} \partial y_0/\partial \mathbf{w}^T \\ \partial y_1/\partial \mathbf{w}^T \\ \cdots \\ \partial y_n/\partial \mathbf{w}^T \end{bmatrix}, \tag{52}$$

where $\partial y/\partial \mathbf{w} = col\mathbf{x}^r$.

The Jacobian matrix is constant for static HONUs which reduces the required number of computations. Finally, the change in the HONUs wieghts is calculated as follows

$$\mathbf{\Delta w} = \left(\mathbf{J}^T \cdot \mathbf{J} + \frac{1}{\mu} \cdot \mathbf{I}\right)^{-1} \cdot \mathbf{J}^T \cdot \mathbf{e} \tag{53}$$

$$\mathbf{e} = \mathbf{yr} - \mathbf{y} \tag{54}$$

4.3.2 Conjugate Gradient – CG

As HONU is linear in weights, then CG for HONUs [13] can directly be applied to solve HONU's weights as follows

$$\mathbf{y} - \mathbf{colX} \cdot \mathbf{w} = \mathbf{0}, \tag{55}$$

where

$$\mathbf{colX} = \begin{bmatrix} \mathbf{colx}(k=1)^T \\ \mathbf{colx}(k=2)^T \\ \vdots \\ \mathbf{colx}(k=N)^T \end{bmatrix} = \mathbf{J}. \tag{56}$$

To put equation (55) in CG form $\mathbf{b} - \mathbf{A} \cdot \mathbf{w} = \mathbf{0}$ as $\mathbf{A}$ is a positive semi-definite matrix, multiply (55) with $\mathbf{colX}^T$. Then,

$$\mathbf{A} = \mathbf{colX}^T \cdot \mathbf{colX}, \tag{57}$$

$$\mathbf{b} = \mathbf{colX}^T. \tag{58}$$

Summary of CG Algorithm [13]

- Initialize (training epoch $\epsilon = 0$)
  - $\mathbf{r_e}(\epsilon = 0) = \mathbf{c} - \mathbf{A} \cdot \mathbf{w}(\epsilon = 0)$
  - $\mathbf{p}(\epsilon = 0) = \mathbf{r_e}(\epsilon = 0)$
- For further training epochs $\epsilon > 0$
  - $\alpha(\epsilon) = \dfrac{\mathbf{r_e}^T(\epsilon) \cdot \mathbf{r_e}(\epsilon)}{\mathbf{p}^T(\epsilon) \cdot \mathbf{A} \cdot \mathbf{p}(\epsilon)}$
    - Update the weights
    - $\mathbf{w}(\epsilon + 1) = \mathbf{w}(\epsilon) + \alpha(\epsilon) \cdot \mathbf{p}(\epsilon)$
  - Update CG parameters for next training epoch
    - $\mathbf{r_e}(\epsilon + 1) = \mathbf{r_e}(\epsilon) - \alpha(\epsilon) \cdot \mathbf{A} \cdot \mathbf{p}(\epsilon)$
    - $\beta(\epsilon) = \dfrac{\mathbf{r_e}^T(\epsilon + 1) \cdot \mathbf{r_e}(\epsilon + 1)}{\mathbf{r_e}^T(\epsilon) \cdot \mathbf{r_e}(\epsilon)}$
    - $\mathbf{p}(\epsilon + 1) = \mathbf{r_e}(\epsilon + 1) + \beta(\epsilon) \cdot \mathbf{p}(\epsilon)$

4.4 Batch Learning Results

4.4.1 P-Controlled Pendulum Batch Learning

The mathematical model of P-Controlled pendulum is described in details in chapter 3.1. The model is used to produce both the training and test data. A Dynamic HONU of order three or Cubic Neural Unit (CNU) is used to learn the system dynamics as described in the following equations

$$y_{k+1} = \mathbf{w} \cdot col^r(\mathbf{x}_k), r = 3, \tag{59}$$

$$\mathbf{x}_k = [1, y_k, y_{k-1}, ..., y_{k-n_y+1}, u_k, u_{k-1}, ..., u_{k-n_u+1}]^T, n_y = 6, n_u = 4. \tag{60}$$

The results of training and testing HONU is summarized in Figure 13, Figure 14 and Error: Reference source not found. Figure 13 shows the training data generated from the model, while Figure 14 shows the weights update at every training epoch using L-M. Finally the test data in Figure 15 uses different step inputs and compares the response of the real system and HONU.

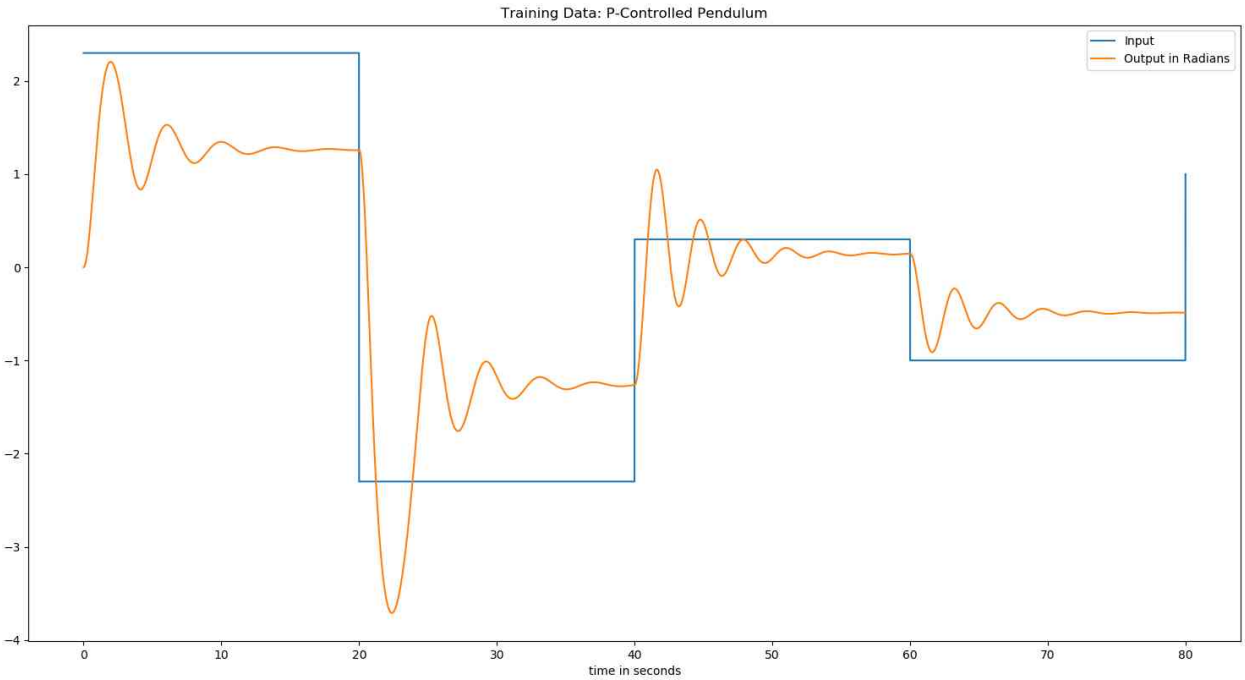

Figure 13: Training data of a p-controlled pendulum

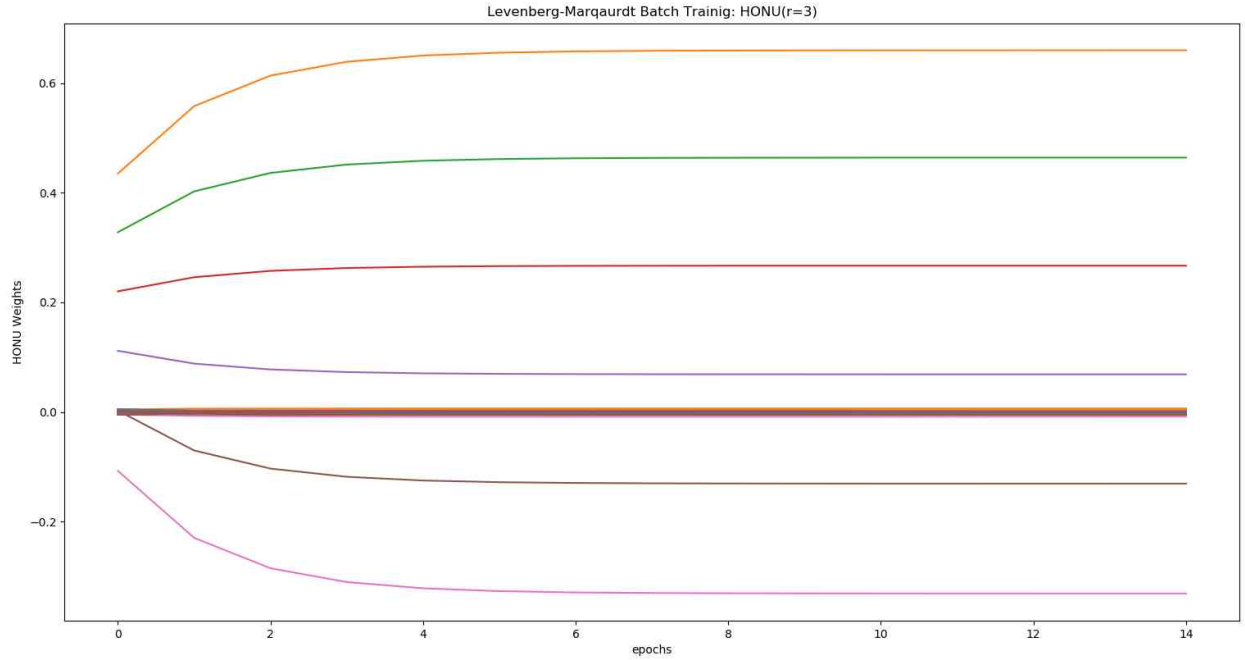

Figure 14: HONU weights learning P-Controlled Pendulum using L-M

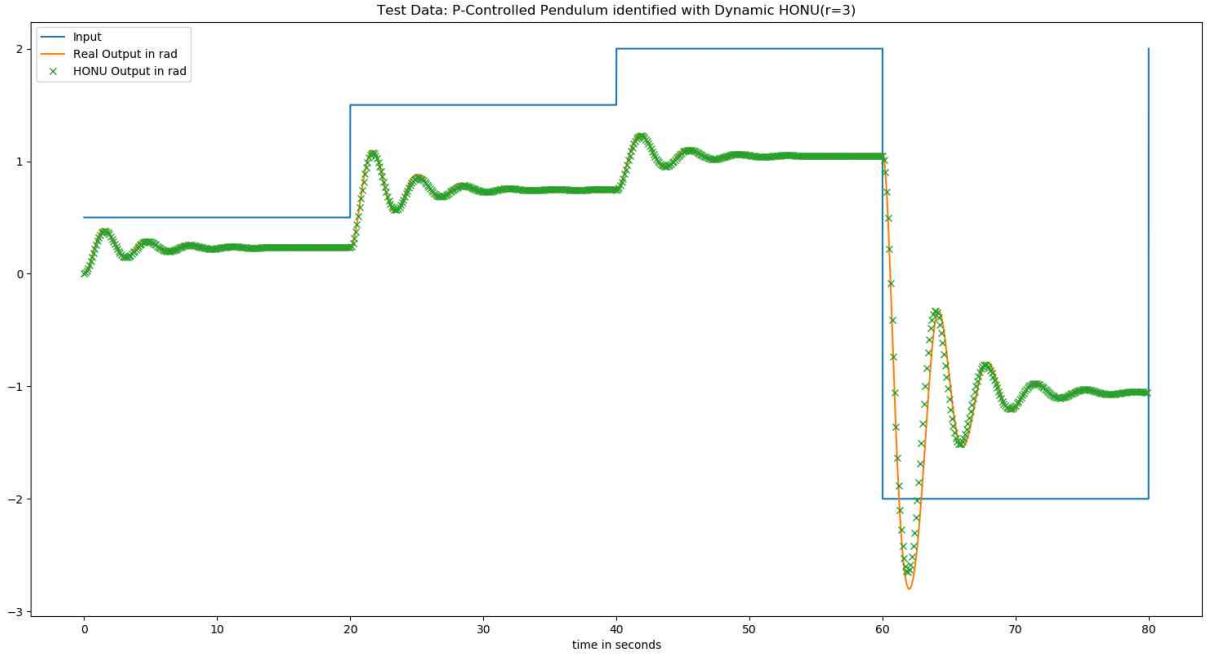

Figure 15: Test data to HONU after learning a p-controlled pendulum.
The pre-trained CNU approximates the real system for different inputs which indicates that CNU does not overfit the training data.

4.4.2 Pendulum Batch Learning

The mathematical model of a torsional pendulum is described in details in chapter 3.1. The model is used to produce both the training and test data. A Dynamic HONU of order three or Cubic Neural Unit (CNU) is used to learn the system dynamics as described in the following equations

$$y_{k+1} = \mathbf{w} \cdot col^r(\mathbf{x}_k), r = 3, \tag{61}$$

$$\mathbf{x}_k = [1, y_k, y_{k-1}, ..., y_{k-n_y+1}, u_k, u_{k-1}, ..., u_{k-n_u+1}]^T, n_y = 6, n_u = 4. \tag{62}$$

The results of training and testing HONU is summarized in Figure 16, Figure 17 and Figure 18. The test data in Figure 18 uses different step inputs and compares the response of the real system and HONU. However it is obvious the dynamic HONU fails to approximate the system nonlinear dynamics for large inputs. However, static HONU works fine to predict one sample ahead as show in Figure 19.

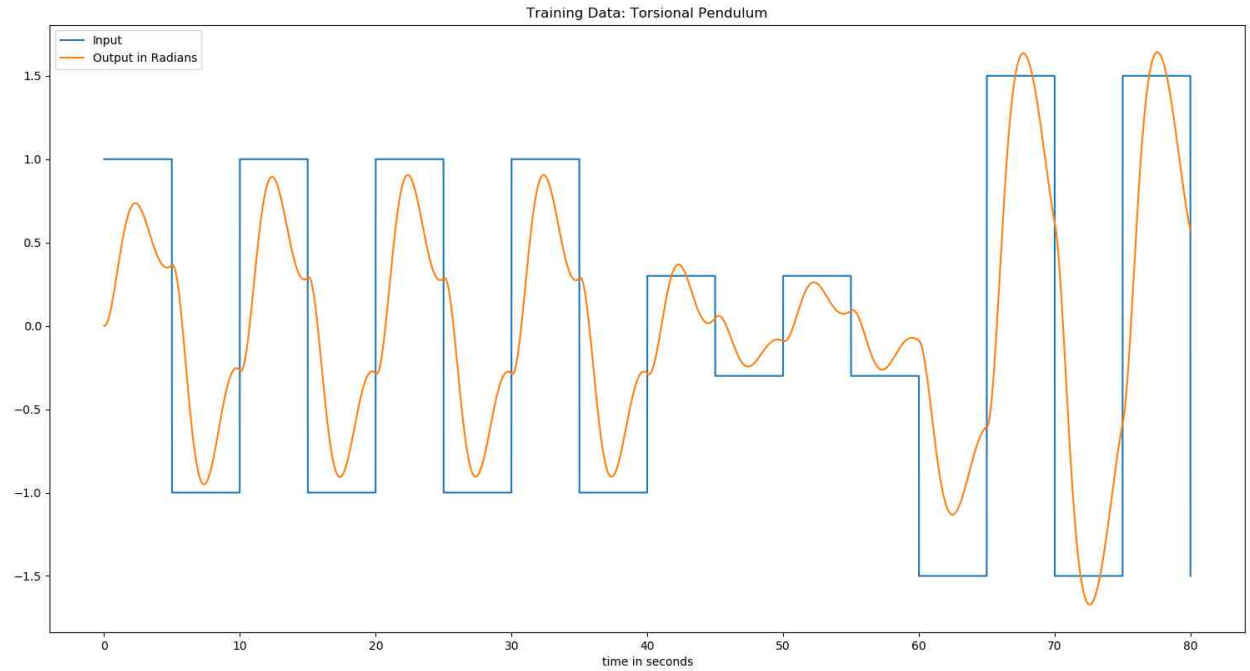

Figure 16: Pendulum training data

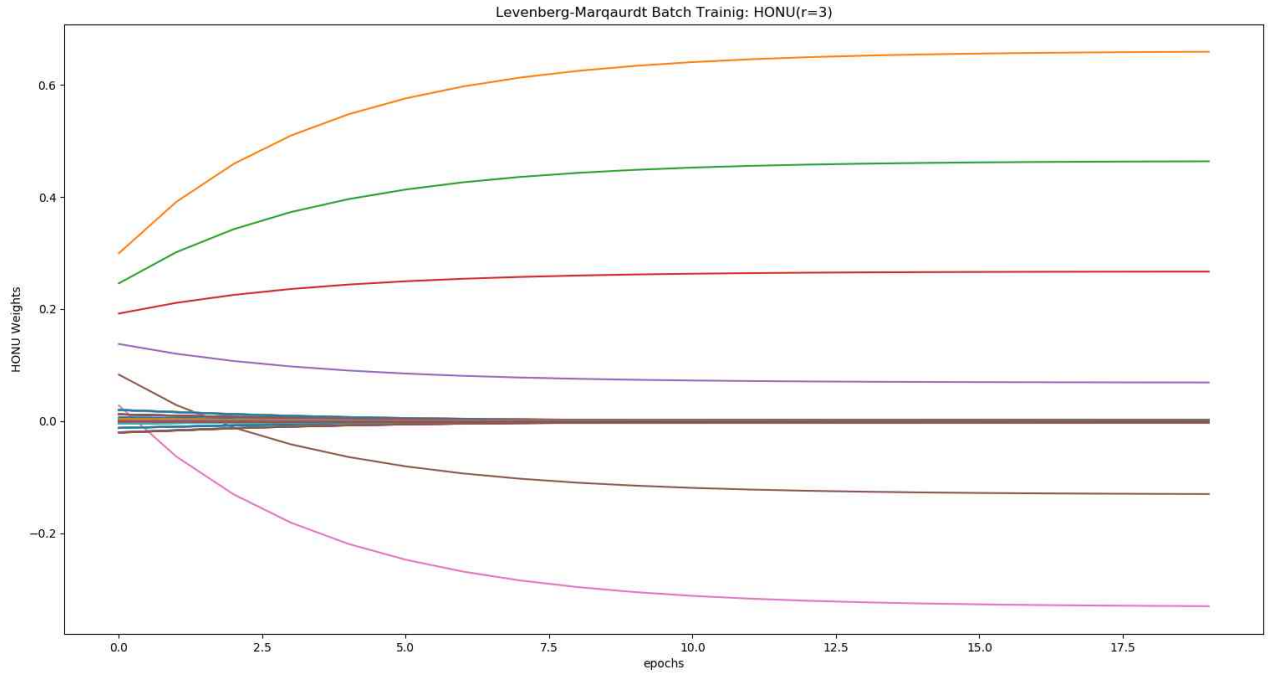

Figure 17: HONU Weights learning torsional pendulum training data

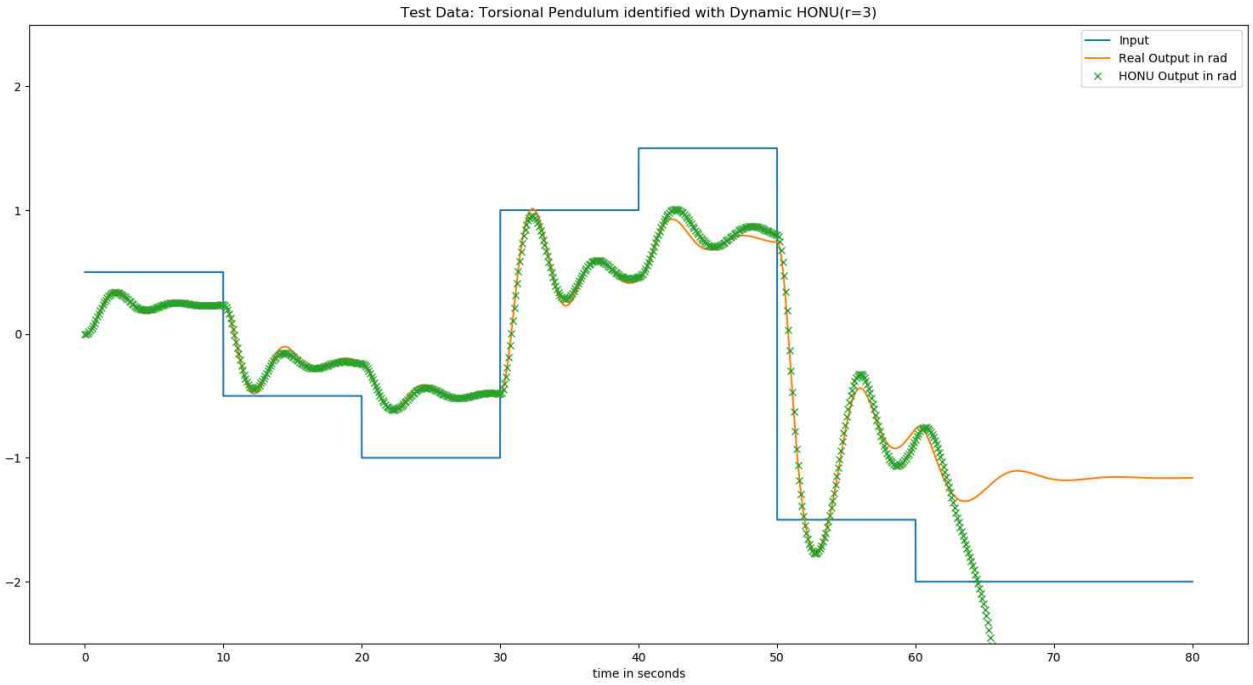

Figure 18: Test data of a torsional pendulum and dynamic HONU CNU fails to approximate the system nonlinear dynamics for large inputs.

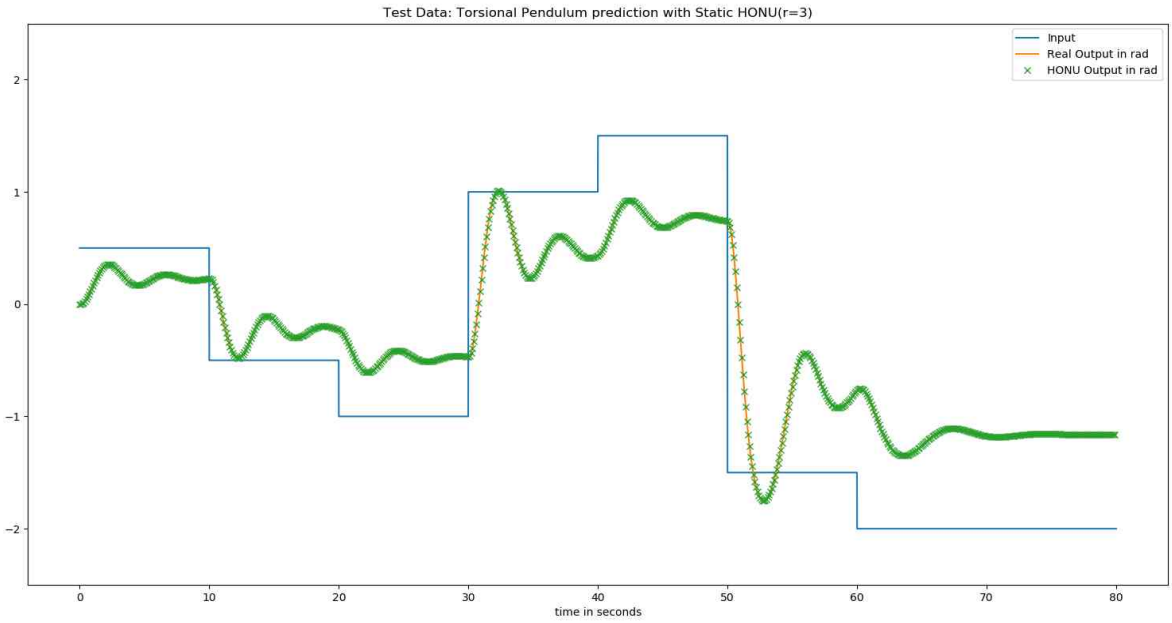

Figure 19: Test data of a torsional pendulum and static HONU
Dynamic CNU fails to approximate the system nonlinear dynamics for large inputs. However, the static CNU works fine to predict one sample ahead

4.4.3 Planar 2-DOF Manipulator Batch Learning

The mathematical model of a planar 2-DOF manipulator is described in details in chapter 3.2. The model is used to produce both the training and test data. A Dynamic HONU of order three or Cubic Neural Unit (CNU) is used to learn the system dynamics. The system has two outputs (first and second joint angles), thus two different HONUs are used for each output. Also, only the motor on the first joint provides torque in this simulation and the second motor torque is zero all the time.

- The first joint angle CNU

$$y_{k+1} = \mathbf{w} \cdot col^r(\mathbf{x}_k), r = 3, \tag{63}$$

$$\mathbf{x}_k = [1, y_k, y_{k-1}, ..., y_{k-n_y+1}, u_k, u_{k-1}, ..., u_{k-n_u+1}]^T, n_y = 6, n_u = 4 \quad , \tag{64}$$

where y, u are the first joint angle and the first motor torque respectively.

- The second joint angle CNU

$$y_{k+1} = \mathbf{w} \cdot col^r(\mathbf{x}_k), r = 3, \tag{65}$$

$$\mathbf{x}_k = [1, y_k, y_{k-1}, ..., y_{k-n_y+1}, u_{1k}, u_{2k}]^T, n_y = 6, \tag{66}$$

where y, u_1, u_2 are the second joint angle, the first joint angle and the the first motor torque. HONU composes its own output selection mechanism so that feed the first angle as an input is not necessary. However, it helps to reduce HONU size.

Figure 20, Figure 21 and Figure 22 shows the training process of both joints HONUs. First L-M starts the learning for several epochs then CG accelerates the learning. In Figure 23, the dynamic HONUs are tested against new test data. For small inputs, both dynamic HONUs work fine but for relatively large inputs they fail to identify the manipulator dynamics. However, static HONU succeed to predict one sample ahead for all levels of inputs as shown Figure 24.

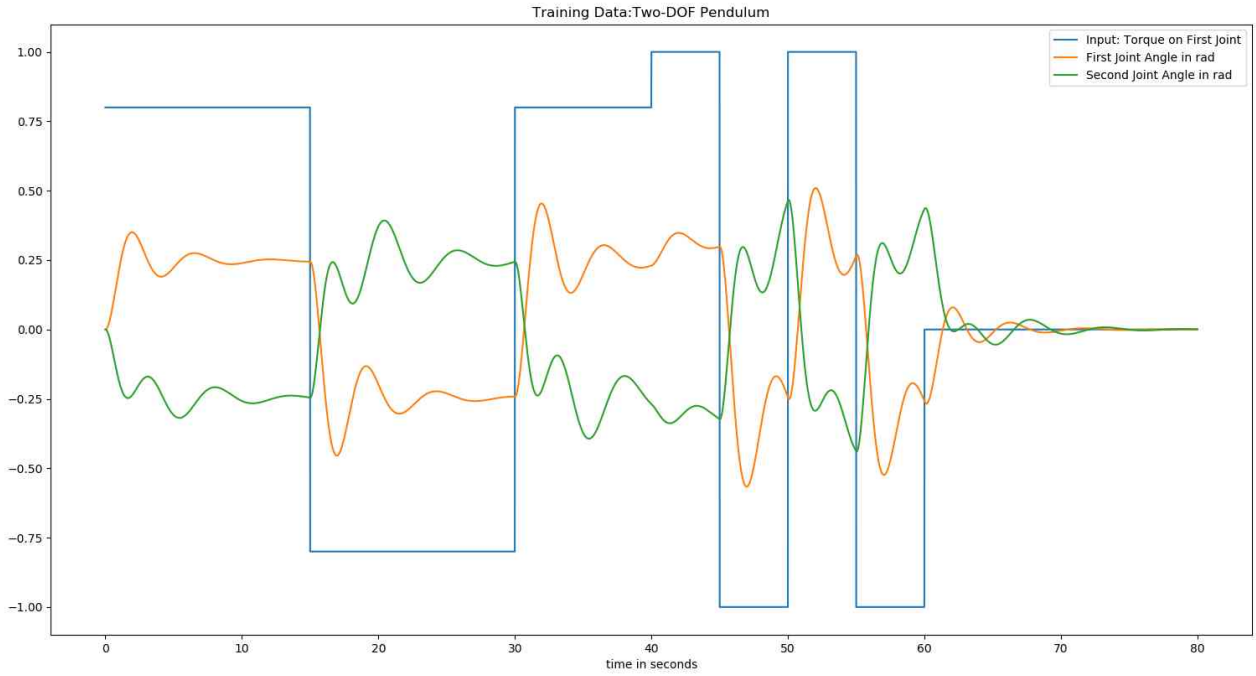

Figure 20: Training data to a planar 2-DOF manipulator

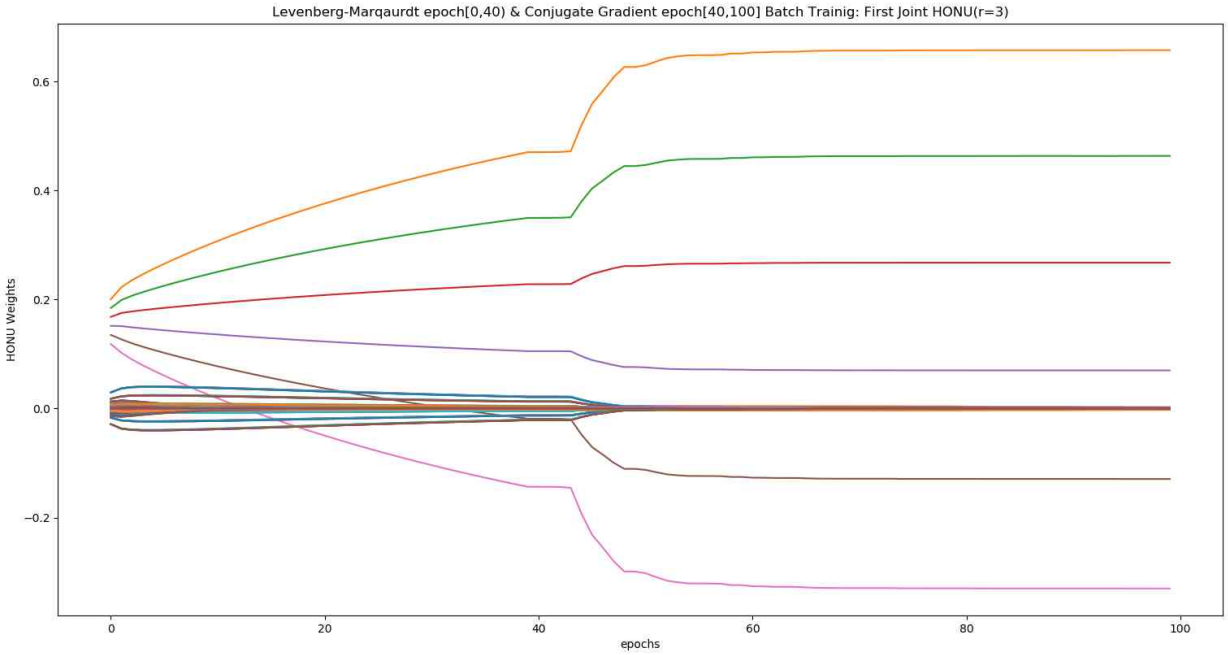

Figure 21: CNU Weights learning the manipulator first joint dynamics using L_M then CG

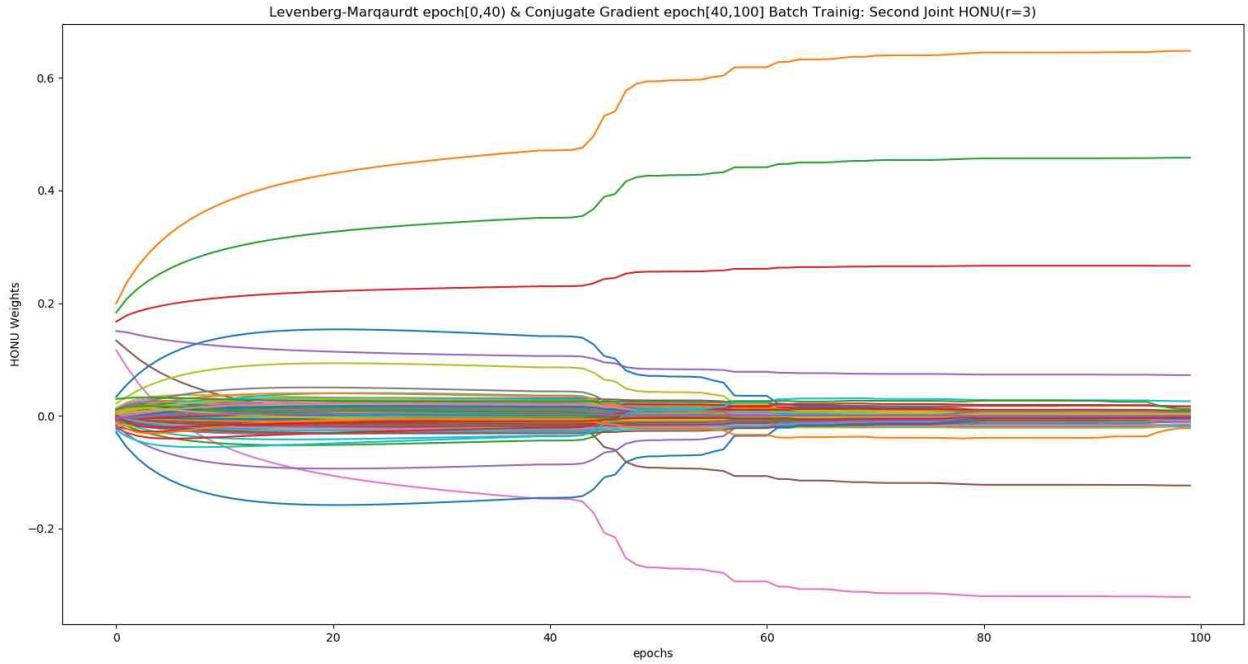

Figure 22: CNU weights learning the manipulator second joint dynamics using L_M then CG

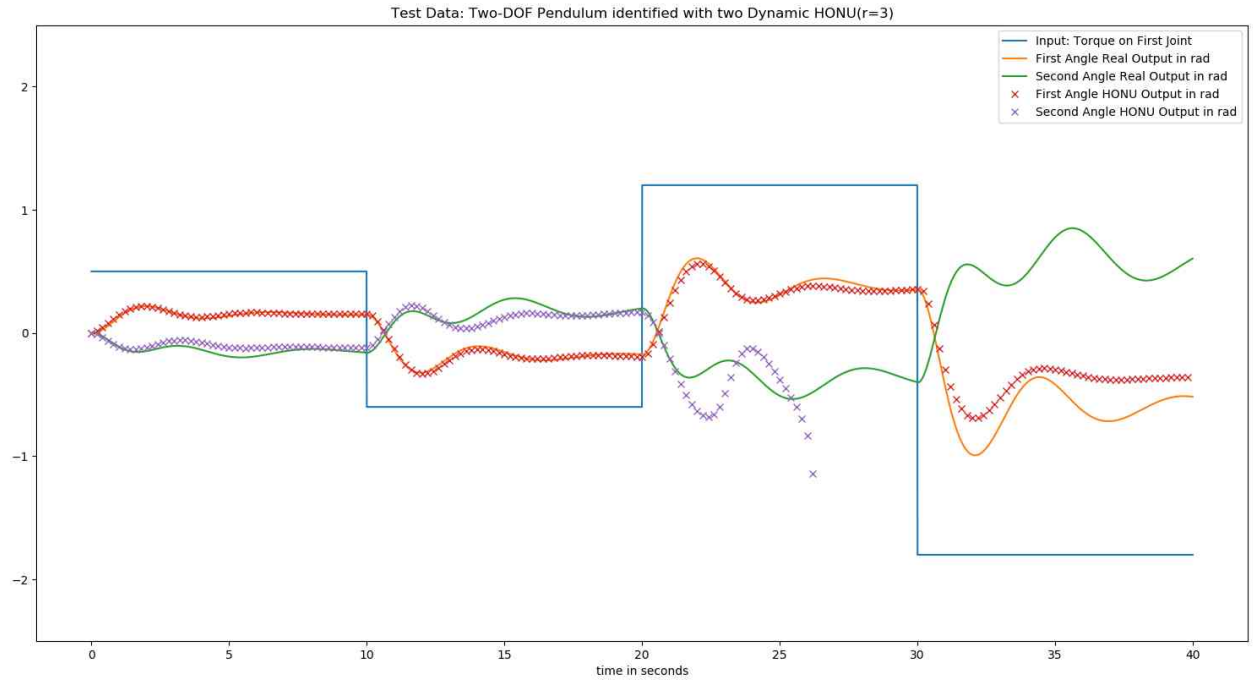

Figure 23: Test of 2-DOF manipulator with two dynamic real-valued HONUs.
Dynamic CNU fails to approximate the system nonlinearity specially for larger inputs. Thus there is a need of another type of HONU for such nonlinear system.

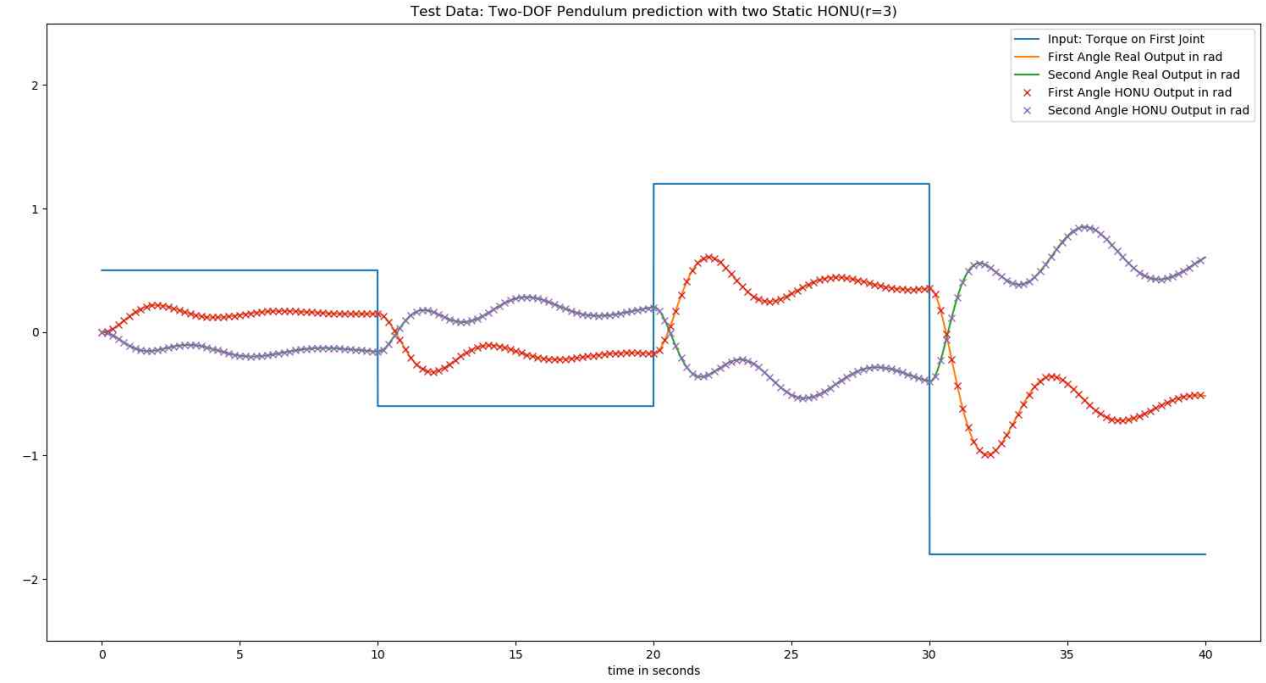

Figure 24: Test data of 2-DOF manupulator and two static HONUs.
Dynamic CNU fails to approximate the system nonlinearity specially for larger inputs. However static CNU is still able to predict one sample ahead for all inputs in the test data.

5 Complex-Valued Higher Order Neural Units – CV-HONU

Chapter 4 investigates the real-valued HONUs for system identification of models with sinusoidal nonlinearities, introduced in Chapter 3. The real-valued static HONUs are able to predict one sample ahead for P-controlled pendulum, pendulum and a planar 2-DOF manipulator. The dynamic HONU is able to approximate the p-controlled pendulum for different input steps. However, dynamic HONU fails to approximate the pendulum and 2-DOF manipulator for large inputs. Therefore, this chapter investigates how complex-valued HONU can cope with this problem.

This chapter introduces the main theoretical contribution of this thesis. The proposal is to develop complex-valued modifications for HONUs and their fundamental learning algorithms. The proposed complex-valued HONUs are tested and compared to the approximation of the dynamical systems (from Chapter 3) and compared to results achieved with real valued HONUs in Chapter 4.

This chapter is divided into two main sections to introduce two different proposals for complex-valued HONU architecture:

- complex valued HONU, denoted as CV-HONU and
- exponent complex valued HONU, denoted as ECV-HONU.

5.1 Introduction to Complex-Valued System Identification

The cost function for complex-valued system is defined as follows

$$e_k = \frac{1}{2}(yr_k - y_k)\overline{(yr_k - y_k)} \tag{67}$$

Theorem 1. By treating z and z as independent variables, the quantity pointing in the direction of the maximum rate of change of $f(z, \overline{z})$ is $\nabla_{\overline{z}}(f(z))$.

Theorem 1 in [24, 25] defines the direction to minimize the cost function. Following theorem 1, one can define the derivatives in the conjugate direction of complex-valued HONU weights. The sample-by-sample methods defined in chapter 4 such as gradient descent, Adam optimizer or any methods that uses a similar cost function as in (67), follows theorem 1 in defining the gradients. The gradients are explained in detail for CV-HONU and ECV-HONU in the following subsections. However other learning methods, that does not use cost function as in (67), does not follow theorem 1. For example there is no alternation to L-M methods. Also, CG can apply if CV-HONU architecture stays linear in weights.

5.2 Complex Valued HONUs (CV-HONUs)

The first proposal is similar to real-valued HONU but with complex weights. This proposal is not expected to work better than real-valued HONU for sinusoidal nonlinearity. However, it is for better understanding the fundamentals of complex valued weights learning as follows

$$\mathbf{x} = [x_0 = 1, x_1, x_2, ..., x_n]^T, \tag{68}$$

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]^T, \forall i \in [0, n], w_i = w_i^{re} + jw_i^{im}, \tag{69}$$

where $j^2 = -1$, Also, both x, y are complex-valued. Finally, CV-HONU architecture is as follows

$$y = \mathbf{w} \cdot col^r(\mathbf{x}). \tag{70}$$

5.3 CV-HONU Learning

Similar concepts of a real-valued HONU applies [17, 26]. The use of gradient descent to minimize certain cost function applies however both the cost function and the gradient are different as described in the following equations

$$e_k = \frac{1}{2}(yr_k - y_k)\overline{(yr_k - y_k)}. \tag{71}$$

where $\overline{(x_r + jx_i)} = (x_r - jx_i)$. The way downhill the cost function e_k follows the gradient of $\overline{\mathbf{w}}$ according to theorem 1 in subsection 5.1

$$\frac{\partial e_k}{\partial \overline{\mathbf{w}}} = -(yr_k - y_k)\frac{\partial \overline{y}}{\partial \overline{\mathbf{w}}}, \tag{72}$$

$$\frac{\partial \overline{y}}{\partial \overline{\mathbf{w}}} = \overline{col^r(\mathbf{x})}. \tag{73}$$

Then, sample-by-sample methods , described in subsection 4.1, applies easily. However, L-M and CG batch learning method, described in subsection 4.3, has no changes, only the Jacobian is calculated according to the complex weights as follow

$$J = \begin{bmatrix} \partial y_0/\partial \mathbf{w}^T \\ \partial y_1/\partial \mathbf{w}^T \\ ... \\ \partial y_n/\partial \mathbf{w}^T \end{bmatrix}, \tag{74}$$

where $\partial y/\partial \mathbf{w} = col\mathbf{x}^r$.

As an example of this first simple proposal to CV-HONU. A complex electric impedance is identified using a dynamic CV-HONU. The impedance changes with temperature as described in

detail in chapter 3.3. Figure 19, Figure 20 and Figure 21 show the learning and testing of the recurrent CV-HONU after adaptation.

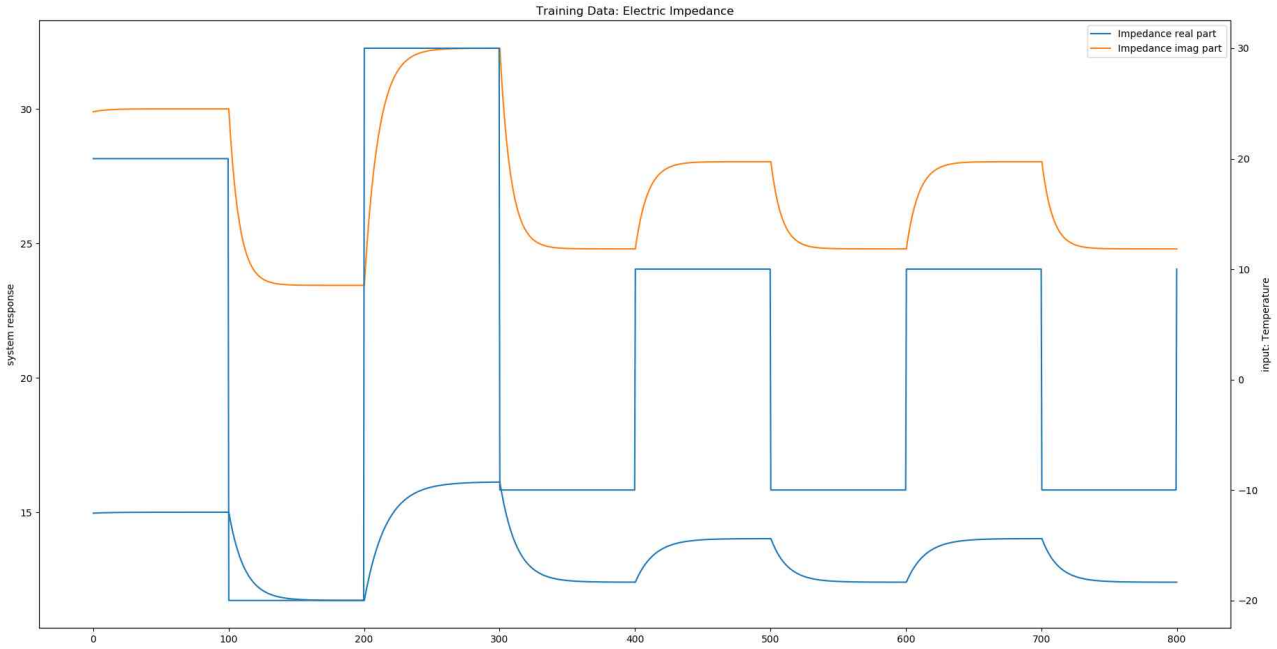

Figure 25: Training Data of an Electric Impedance that responds to temperature change

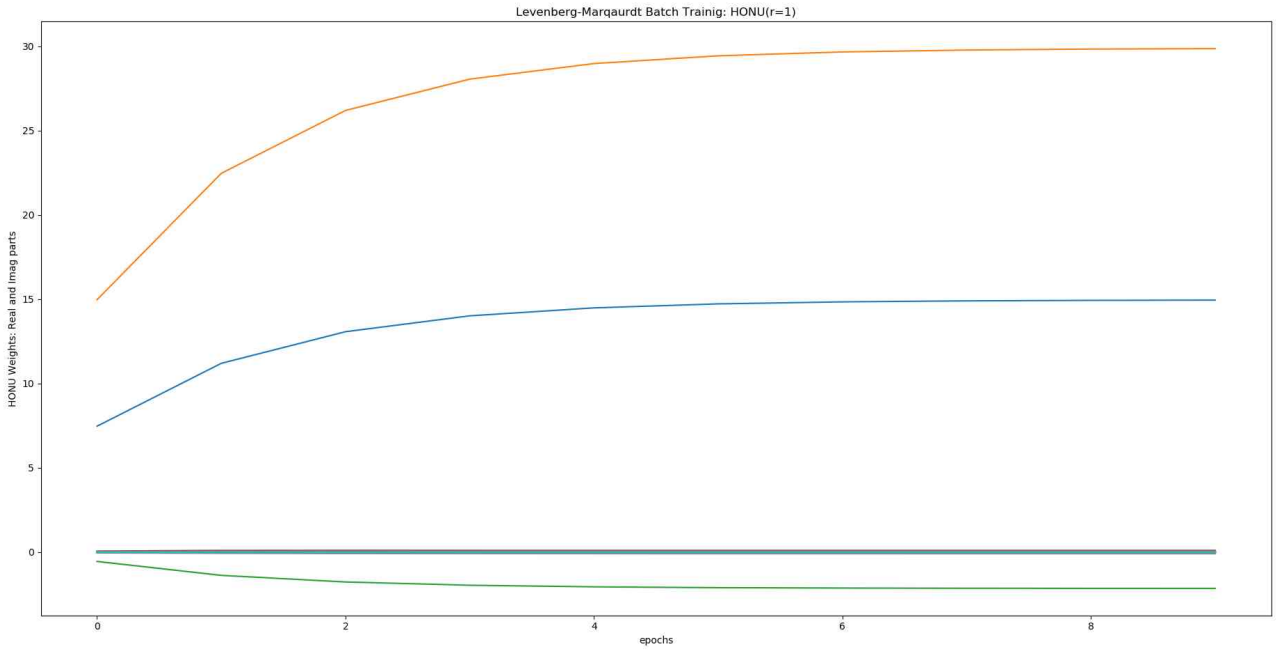

Figure 26: Dynamic CV-HONU learning

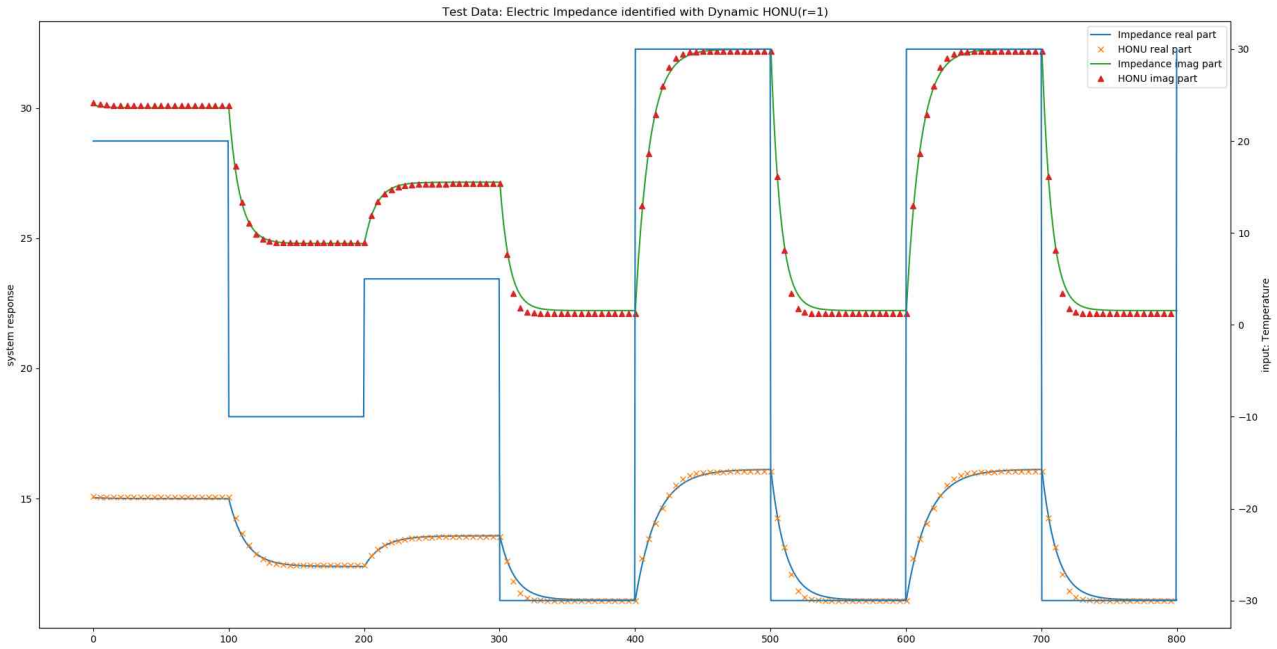

Figure 27: Test of dynamic CV-HONU
Both real and complex outputs of CV-HONU are pretrained successfully which assert learn methods described in this chapter.

5.4 Exponent Complex Valued HONUs (ECV-HONU)

The batch learning results of real-valued HONU in subsection 4.4 shows that HONU fails to approximate systems with sinusoidal nonlinearity as in Figure 23 and Figure 18. To cope with this real-valued HONU limitation, several new architectures for HONU were proposed along this work. HONU approximation capability is that it aggregates higher-order polynomial terms. This aggregation forms new nonlinear relations between the inputs. The idea that we, my supervisor and me, tried to develop in this subsection is to introduce new sinusoidal terms in the HONU architecture. This subsection starts with the new architecture, the learning rules, followed by the simulation results and finally states the finding. The summary of the new HONU proposal is explained in the following equations

$$y = \mathbf{w} \cdot col^r(\mathbf{x}) + \sum_{i=0}^{n_\omega} (e^{\omega_i \cdot colx_i} - 1), \tag{75}$$

$$\mathbf{x} = [x_0 = 1, x_1, x_2, ..., x_n]^T, \tag{76}$$

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]^T, \forall i \in [0, n], w_i = w_i^{re} + j w_i^{im}, \tag{77}$$

$$\boldsymbol{\omega} = [\omega_0, \omega_1, \omega_2, ..., \omega_n]^T, \forall i \in [0, n], \omega_i = \omega_i^{re} + j\omega_i^{im}. \tag{78}$$

where $j^2 = -1$ and both x, y can also be complex values.

As seen in equation (75), a simple summation of natural exponent is added to HONU which can enhance the CV-HONU approximation to sinusoidal functions without using a very high order HONU $(r >> 1)$. The subtraction of -1, from the natural exponent in the summation in equation (75), is to remove the bias from ECV-HONU so that when $\mathbf{x} = 0$ then output $y = 0$. An alternatively considered version of equation (75) that caused bias was as follows

$$y = \mathbf{w} \cdot col^r(\mathbf{x}) + \sum_{i=0}^{n_\omega} (e^{\omega_i \cdot colx_i} - 0). \tag{79}$$

5.5 ECV-HONU Learning

Following similar steps to CV-HONU learning, the cost function is the same as in first proposal 5.3

$$e_k = \frac{1}{2}(yr_k - y_k)\overline{(yr_k - y_k)}, \tag{80}$$

where $\overline{(x_r + jx_i)} = (x_r - jx_i)$.

The way downhill the cost function e_k follows the gradient of $\overline{\mathbf{w}}$ and $\overline{\boldsymbol{\omega}}$ according to theorem 1 in subsection 5.1 that is

for non-exponential part of ECV-HONUs, i.e. $\overline{\mathbf{w}}$, as follows

$$\frac{\partial e_k}{\partial \overline{\mathbf{w}}} = -(yr_k - y_k)\frac{\partial \overline{y}}{\partial \overline{\mathbf{w}}}, \tag{81}$$

$$\frac{\partial \overline{y}}{\partial \overline{\mathbf{w}}} = \overline{col^r(\mathbf{x})}, \tag{82}$$

and for exponential part of ECV-HONUs, i.e. $\overline{\boldsymbol{\omega}}$, it is as follows

$$\frac{\partial e_k}{\partial \overline{\boldsymbol{\omega}}} = -(yr_k - y_k)\frac{\partial \overline{y}}{\partial \overline{\boldsymbol{\omega}}}, \tag{83}$$

$$\frac{\partial \overline{y}}{\partial \overline{\boldsymbol{\omega}}} = \sum_{i=0}^{n_\omega} \overline{(e^{\omega_i \cdot colx_i} \times colx_i)}. \tag{84}$$

Then, sample-by-sample methods , described in subsection 4.1, applies easily. According to the batch learning methods , described in subsection 4.3, CG does not apply to ECV-HONU as it is not linear in weights. However, L-M is be used, but the Jacobian is calculated as follows

$$J = \begin{bmatrix} \partial y_0/\partial \mathbf{w}^T & \partial y_0/\partial \boldsymbol{\omega}^T \\ \partial y_1/\partial \mathbf{w}^T & \partial y_1/\partial \boldsymbol{\omega}^T \\ \cdots & \cdots \\ \partial y_n/\partial \mathbf{w}^T & \partial y_n/\partial \boldsymbol{\omega}^T \end{bmatrix}, \tag{85}$$

where

$$\frac{\partial y}{\partial \boldsymbol{\omega}} = \sum_{i=0}^{n_\omega} (e^{\omega_i \cdot colx_i} \times colx_i), \tag{86}$$

$$\frac{\partial y}{\partial \mathbf{w}} = col\mathbf{x}^r, \tag{87}$$

then

$$\begin{bmatrix} \boldsymbol{\Delta}\mathbf{w} \\ \boldsymbol{\Delta}\boldsymbol{\omega} \end{bmatrix} = \left(\mathbf{J}^T \cdot \mathbf{J} + \frac{1}{\mu} \cdot \mathbf{I}\right)^{-1} \cdot \mathbf{J}^T \cdot \mathbf{e}. \tag{88}$$

The first test uses a linear ECV-HONU(r=1) using the same training data of a torsional pendulum in chapter 4.4.2. At this work level, only static ECV-HONU is tested and in future work dynamic ECV-HONU should be tested against real-valued HONU. The results of the static ECV-HONU(r=1), Figure 28 and Figure 29 show that the new proposed ECV-HONU can be trained easily.

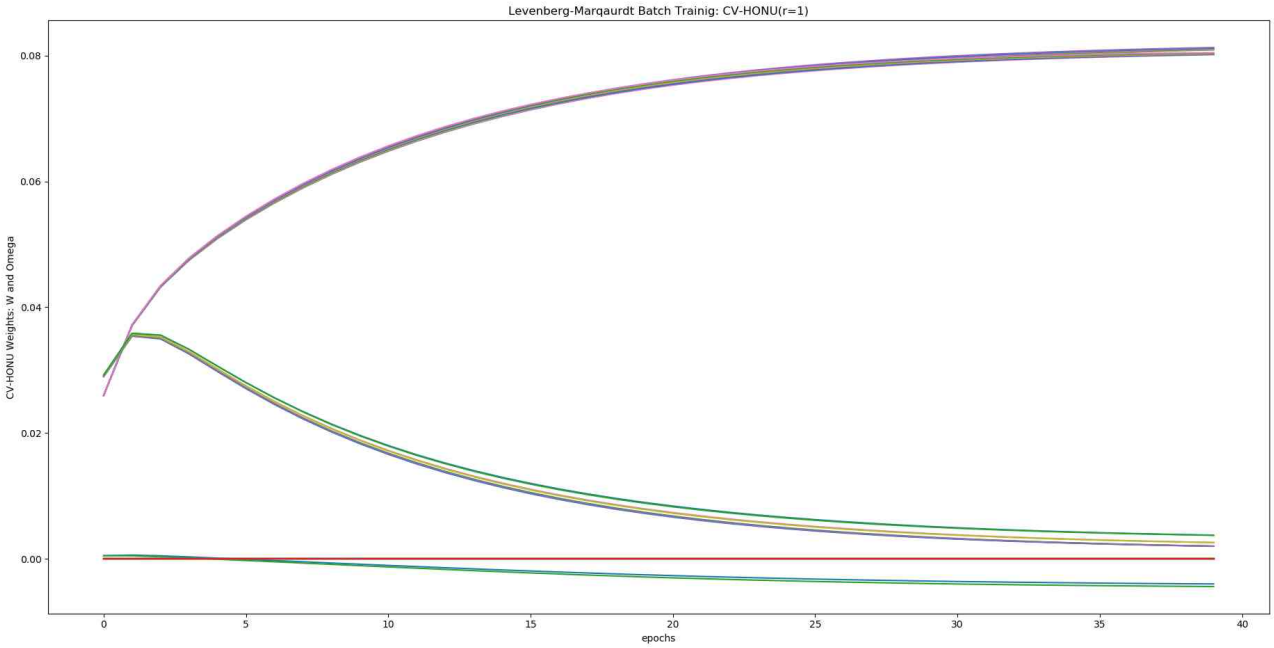

Figure 28: Learning of ECV-HONU weights

5.6 Chapter Summary

Two different complex-valued HONU architectures are proposed. The first, CV-HONU, serves for understanding the fundamentals of learning complex weights in HONU and it is tested as a dynamic CV-HONU. The second, ECV-HONU, is work in progress. Several iterations was needed to finally choose ECV-HONU architecture such as removing the bias from the exponent summation. Then, the learning gradients are derived. Finally, the ECV-HONU is tested as a static neuron for simulation data of a torsional pendulum. In future work, dynamic ECV-HONU will be tested against dynamic HONU on different nonlinear system.

	Static HONU (r=1)	Static ECV-HONU (r=1)
Mean of $\|(yr_k - y_k)\|, \forall epochs$	9×10^{-3}	2×10^{-1}
STD of $\|(yr_k - y_k)\|, \forall epochs$	2×10^{-2}	5×10^{-1}
Learning Time	0.01 seconds	19.65 seconds

Table 1: Comparison between static CV-HONU and HONU for the simulation in Figure 29

However, ECV-HONU lacks one of the distinctive characteristics of HONU which is the linearity in parameters (weights). HONU's linearity in parameters reduces the computation for L-M as the Jacobian is constant for all the training epochs. Table 1 compares static HONU to static ECV-HONU. From the comparison, static HONU predicts the torsional pendulum in Figure 29 more accurately. Also, static HONU consumes significantly less computation power because it has a

constant Jacobian per training data. Also, Figure 30 compares the convergence between static HONU and ECV-HONU.

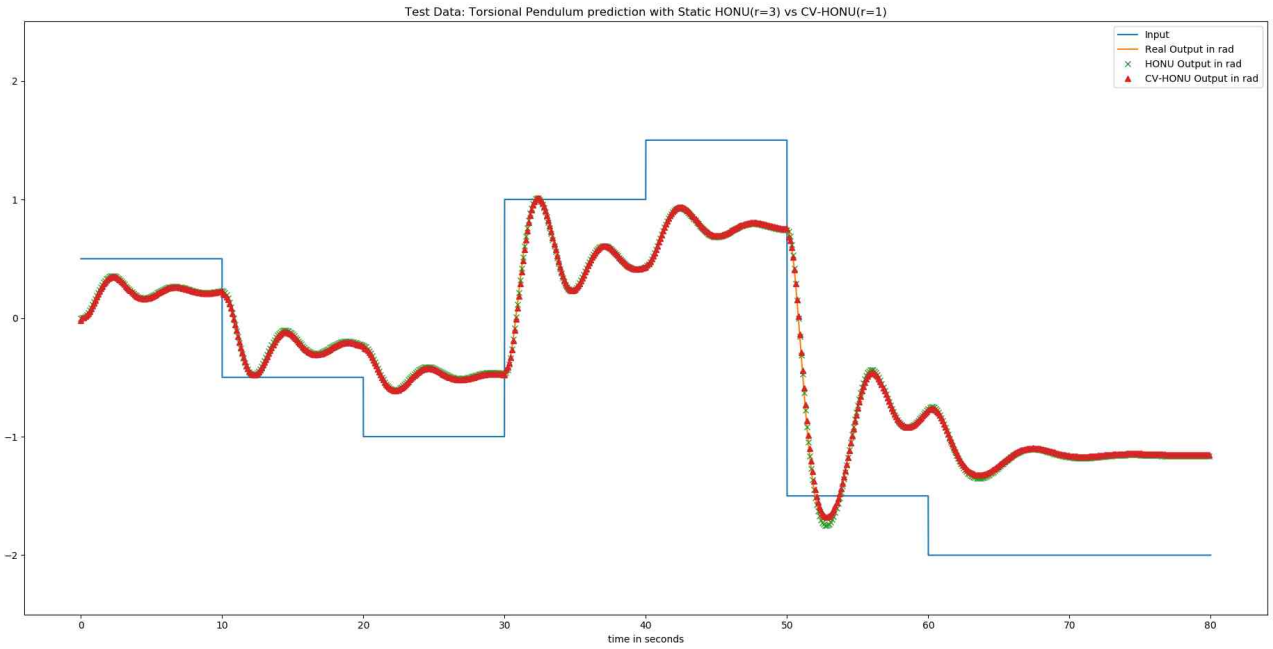

Figure 29: Static ECV-HONU test data for a torsional pendulum

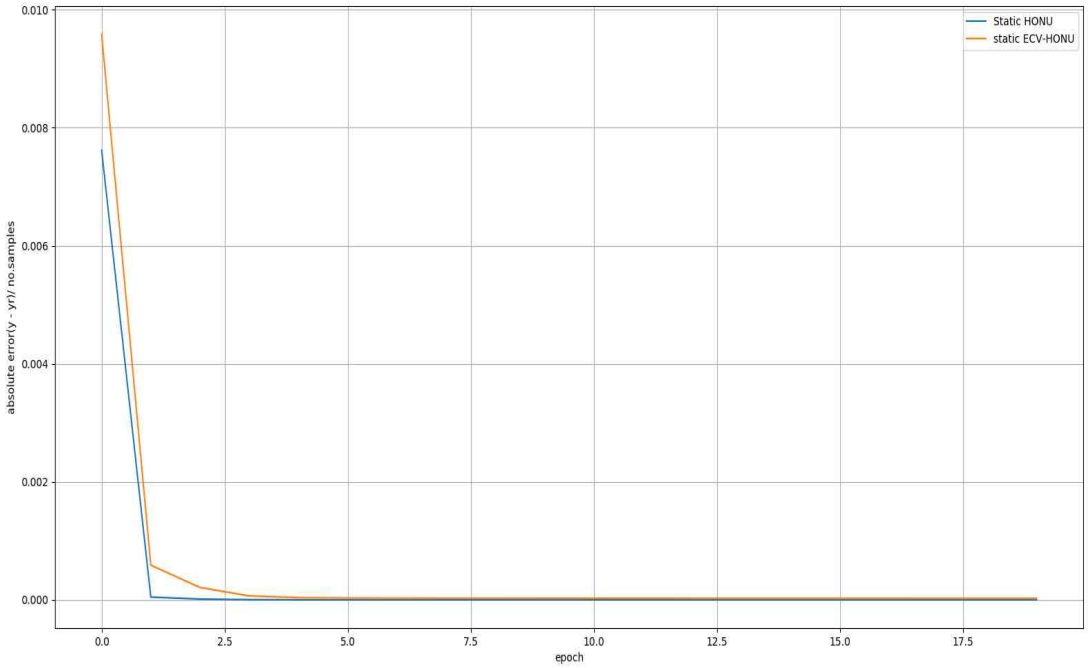

Figure 30: Convergence of Static HONU(r=1) and Static ECV-HONU(r=1). Using L-M to train the system in Figure 29

6 Object-Oriented Python Toolbox for HONU

The study also includes the development of a software toolbox Figure 31 for HONUs, complex-valued HONUs and their learning approaches. The developed software toolbox has two main purposes. First, the toolbox helps for better understanding of HONUs by visualizing their learning process and simulating the output. Secondly, a well-designed toolbox is the mean to integrate this study results with a production level software in different platforms. Also, the development of such toolbox is essential to make the results of this work easily used by other colleagues. As Python is a powerful programming language for scientific research and machine learning, an Object-Oriented python toolbox has been developed during this work Figure 31. Previous students works at CTU on software for real-valued HONU with fundamental learning rules are as follows:

- work [27] in Java
- and [28] in Python 2.7, WX Python including control algorithms.

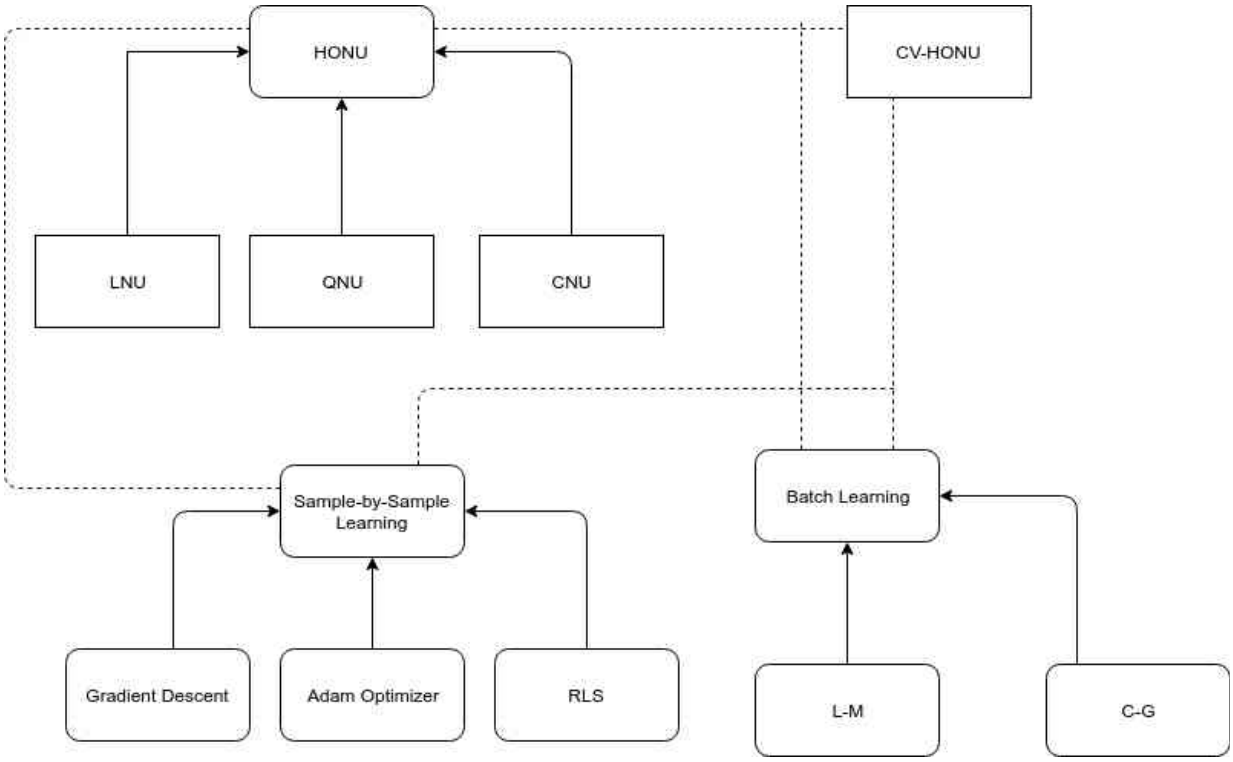

Figure 31: HONU SW toolbox simplified class diagram.

The classes in the toolbox are divides into four main base classes: HONU, sample-by-sample learning, batch learning and experimental architecture of HONUs. The following bullets points describe these base classes and some of their sub-classes:

- **HONU** Simulate HONU, update the state vector, calculate Jacobian and gradient for any order of HONU. Subsections 2.1, 2.2
  - ***LNU** cubic HONU(r=1)* a wrapper class that helps customizing the base class **HONU**

- ***QNU** cubic HONU(r=2)* a wrapper class that helps customizing the base class **HONU**
- ***CNU*** cubic HONU(r=3) a wrapper class that helps customizing the base class **HONU**

- **sample-by-sample** an abstract class for sample-by-sample learning approaches
  - ***Adam Optimizer*** implementation of the base class **sample-by-sample** to Adam Optimizer. Subsection 4.1.3
  - ***Gradient Descent*** implementation of the base class **sample-by-sample** to gradient-descent. Subsection 4.1.1
  - ***RLS*** recursive least square implementation of the base class **sample-by-sample**. Subsection 4.1.2

- **Batch Learning** an abstract class to for batch learning methods from subsection 4.3
  - ***CG*** conjugate gradient implementation from subsection 4.3.2
  - ***L-M*** Levenburg-Marquardt implementation from subsection 4.3.1

- **experimental architecture of HONUs**
  - ***CV-HONU*** has the same implementation as described in subsection 5.2
  - ***ECV-HONU*** has the same implementation as described in subsection 5.4
- **Dynamic Systems** abstract class to dynamic systems used to produce HONUs training and testing data.
  - ***Torsional pendulum***
  - ***P-Controlled Pendulum***
  - ***2-DOF manipulator***
  - ***Temperature-dependent Impedance***
  - ***Second order system***
  - ***First order system***

The toolbox produced all the plots and most of the figures in this thesis. All the scripts, used to produce the figures, are saved and attached with the toolbox which is attached as a CD with the printed version of this thesis.

7 Conclusion

The thesis introduces a review of dynamic and static high-order neural units (HONU) in approximating dynamic systems. Chapter 3 derives the mathematical models of two main dynamic systems, torsional pendulum and 2-DOF manipulator, using Lagrangian mechanics. These two dynamic systems include sinusoidal nonlinearities. Then, chapter 4 discusses two fundamental approaches to HONU supervised learning: sample-by-sample and batch learning. The sample-by-sample learning is tested using an oscillating second-order system. Both gradient-descent and Adam optimizer are able to adapt HONU online. However, Adam optimizer convergence time is half gradient-descent convergence time, Figure 10 and Figure 11.

Furthermore, chapter 4 discusses HONU capability to approximate the nonlinear dynamics of p-controlled pendulum, torsional pendulum and 2-DOF manipulator. Both static and dynamic HONU are able to predict the response of a p-controlled pendulum. While static HONU is able to predict one sample ahead in the test simulation of a torsional pendulum, the dynamic HONU is able to approximate the dynamics only for small inputs. Increasing the sinusoidal nonlinearity in 2-DOF manipulator, static HONU is still able to predict one sample-ahead during the test simulation. However, dynamic HONU is not able to approximate the manipulator dynamics for large inputs. These findings confirm the outcomes of most recent research at CTU in Prague.

Being inspired from complex-valued neural networks and their supervised learning, according to my current knowledge, this thesis is the first to propose complex-valued high-order neural units (CV-HONU) (subsection 5.2). Moreover, Chapter 5 derives in detail different architectures to CV-HONU and their learning rules: sample-by-sample and batch learning. The first architecture expands real-valued HONU to cover the complex values using the same approximation capability. A model of complex-valued impedance, that changes with temperature, is approximated using dynamic CV-HONU. In addition, the thesis investigates ho CV-HONU can approximate the sinusoidal nonlinearity. **Therefore, a new architecture of HONU is proposed, Exponent complex-valued high order neural units (ECV-HONU), subsection 5.4.** The added exponent terms aggregate sinusoidal relations between the inputs. In subsection 5.5, the thesis continues to derive the learning rules of ECV-HONU: sample-by-sample and batch learning. However, since ECV-HONU is not linear in parameters (weights), the Jacobian matrix in L-M batch learning has to be computed at every training epoch. The lack of parameters linearity significantly increases the learning time of ECV-HONU, Table 1.

During the thesis work, there was a main focus on developing a mature and clean toolbox for all the investigated HONUs and dynamic system. The result is a HONU toolbox and several learning algorithms. Chapter 6 presents the toolbox main features and design. The toolbox enables other colleagues to use all the results in this thesis in their work as well as help the authors to continue research on the same topic. The toolbox is attached on a CD with the printed version of this thesis.

8 References

[1] KARABEGOVIC, Edina, Isak KARABEGOVIC a Edit HADZALIC. Industrial Robot Application Trend in World's Metal Industry. *Engineering Economics* [online]. 2012, **23**(4) [vid. 2018-06-12]. ISSN 2029-5839, 1392-2785. Dostupné z: doi:10.5755/j01.ee.23.4.2567

[2] MAURTUA, Iñaki, Aitor IBARGUREN, Johan KILDAL, Loreto SUSPERREGI a Basilio SIERRA. Human–robot collaboration in industrial applications: Safety, interaction and trust. *International Journal of Advanced Robotic Systems* [online]. 2017, **14**(4), 172988141771601. ISSN 1729-8814, 1729-8814. Dostupné z: doi:10.1177/1729881417716010

[3] IVAKHNENKO, A.G. Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man and Cybernetics* [online]. 1971, **SMC-1**(4), 364–378. ISSN 0018-9472. Dostupné z: doi:10.1109/TSMC.1971.4308320

[4] KOSMATOPOULOS, E.B., M.M. POLYCARPOU, M.A. CHRISTODOULOU a P.A. IOANNOU. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks* [online]. 1995, **6**(2), 422–431. ISSN 1045-9227. Dostupné z: doi:10.1109/72.363477

[5] GUPTA, Madan, Liang JIN a Noriyasu HOMMA. *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. B.m.: John Wiley & Sons, 2004. ISBN 978-0-471-46092-3.

[6] TRIPATHI, Bipin Kumar. Higher-Order Computational Model for Novel Neurons. In: Bipin Kumar TRIPATHI*High Dimensional Neurocomputing* [online]. New Delhi: Springer India, 2015 [vid. 2018-06-10], s. 79–103. ISBN 978-81-322-2073-2. Dostupné z: doi:10.1007/978-81-322-2074-9_4

[7] GUPTA, Madan M, Noriyasu HOMMA, Zeng-Guang HOU, MG SOLO a Ivo BUKOVSKY. Higher order neural networks: fundamental theory and applications. *Artificial Higher Order Neural Networks for Computer Science and Engineering: Trends for Emerging Applications*. 2010, 397–422.

[8] BUKOVSKY, I., S. REDLAPALLI a M.M. GUPTA. Quadratic and cubic neural units for identification and fast state feedback control of unknown nonlinear dynamic systems. In: *Fourth International Symposium on Uncertainty Modeling and Analysis, 2003. ISUMA 2003*: *Fourth International Symposium on Uncertainty Modeling and Analysis, 2003. ISUMA 2003* [online]. 2003, s. 330–334. Dostupné z: doi:10.1109/ISUMA.2003.1236182

[9] IVO BUKOVSKY. *Modeling of Complex Dynamic Systems by Nonconventional Artificial Neural Architectures and Adaptive Approach to Evaluation of Chaotic Time Series*. B.m.: Czech Technical University in Prague, Faculty of Mechanical Engineering, (PhD thesis). 2007. 00000

[10] SMETANA, L. *Nonlinear Neuro-Controller for Automatic Control Laboratory System*. Prague, Czech Republic, 2008. Master's Thesis. Czech Technical University in Prague.

[11] BENES, Peter Mark, Miroslav ERBEN, Martin VESELY, Ondrej LISKA a Ivo BUKOVSKY. HONU and Supervised Learning Algorithms in Adaptive Feedback Control.

http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-0063-6.ch002 [online]. 2016, 35–60. Dostupné z: doi:10.4018/978-1-5225-0063-6.ch002

[12] CYRIL OSWALD, MATOUS CEJNEK, JAN VRBA a IVO BUKOVSKY. Novelty Detection in System Monitoring and Control with HONU. In: *Applied Artificial Higher Order Neural Networks for Control and Recognition*. B.m.: IGI Global, 2016, s. pp.61-78. ISBN 978-1-5225-0063-6.

[13] BUKOVSKY, Ivo, Jan VORACEK, Kei ICHIJI a Homma NORIYASU. Higher Order Neural Units for Efficient Adaptive Control of Weakly Nonlinear Systems: In: [online]. B.m.: SCITEPRESS - Science and Technology Publications, 2017, s. 149–157 [vid. 2018-06-10]. ISBN 978-989-758-274-5. Dostupné z: doi:10.5220/0006557301490157

[14] DERONG LIU a QINGLAI WEI. Policy Iteration Adaptive Dynamic Programming Algorithm for Discrete-Time Nonlinear Systems. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2014, **25**(3), 621–634. ISSN 2162-237X, 2162-2388. Dostupné z: doi:10.1109/TNNLS.2013.2281663

[15] MANDIC, Danilo P. a Vanessa Su Lee GOH. *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*. B.m.: John Wiley & Sons, 2009. ISBN 978-0-470-74263-1.

[16] HIROSE, Akira. *Complex-valued neural networks*. 2nd ed. Heidelberg ; New York: Springer, 2012. Studies in computational intelligence, v. 400. ISBN 978-3-642-27631-6.

[17] CHISTYAKOV, Yury S., Elena V. KHOLODOVA, Alexey S. MININ, Hans-Georg ZIMMERMANN a Alois KNOLL. Modeling of Electric Power Transformer Using Complex-Valued Neural Networks. *Energy Procedia* [online]. 2011, **12**, 638–647. ISSN 18766102. Dostupné z: doi:10.1016/j.egypro.2011.10.087

[18] BUKOVSKY, Ivo a Noriyasu HOMMA. An Approach to Stable Gradient-Descent Adaptation of Higher Order Neural Units. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2016, 1–13. ISSN 2162-237X, 2162-2388. Dostupné z: doi:10.1109/TNNLS.2016.2572310

[19] BENES, Peter a Ivo BUKOVSKY. Neural network approach to hoist deceleration control. In: [online]. B.m.: IEEE, 2014, s. 1864–1869 [vid. 2018-06-10]. ISBN 978-1-4799-1484-5. Dostupné z: doi:10.1109/IJCNN.2014.6889831

[20] BUKOVSKY, Ivo, Peter BENES a Matous SLAMA. Laboratory Systems Control with Adaptively Tuned Higher Order Neural Units. In: Radek SILHAVY, Roman SENKERIK, Zuzana Kominkova OPLATKOVA, Zdenka PROKOPOVA a Petr SILHAVY, ed. *Intelligent Systems in Cybernetics and Automation Theory* [online]. Cham: Springer International Publishing, 2015 [vid. 2018-06-10], s. 275–284. ISBN 978-3-319-18502-6. Dostupné z: doi:10.1007/978-3-319-18503-3_27

[21] WIDNALL, S. Lecture L20 - Energy Methods: Lagrange's Equations. N.p.: 16.07 Dyanmics, 2009. In: . nedatováno.

[22] KINGMA, Diederik P. a Jimmy BA. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* [online]. 2014 [vid. 2018-06-10]. Dostupné z: http://arxiv.org/abs/1412.6980

[23] EL-NABARAWY, Islam, Ashraf M. ABDELBAR a Donald C. WUNSCH. Levenberg-Marquardt and Conjugate Gradient methods applied to a high-order neural network. In: [online]. B.m.: IEEE, 2013, s. 1–7 [vid. 2018-06-10]. ISBN 978-1-4673-6129-3. Dostupné z: doi:10.1109/IJCNN.2013.6707004[24] JOHNSON, Don. *Optimization Theory* [online]. 2008. Dostupné z: http://cnx.org/contents/fc8682d5-96c3-4648-b845-f55ef3b243f1@4

[25] MININ, Alexey, Alois KNOLL a Hans-Georg ZIMMERMANN. Complex Valued Recurrent Neural Network: From Architecture to Training. *Journal of Signal and Information Processing* [online]. 2012, **03**(02), 192–197. ISSN 2159-4465, 2159-4481. Dostupné z: doi:10.4236/jsip.2012.32026

[26] AMIN, Md. Faijul, Muhammad Ilias AMIN, A. Y. H. AL-NUAIMI a Kazuyuki MURASE. Wirtinger Calculus Based Gradient Descent and Levenberg-Marquardt Learning Algorithms in Complex-Valued Neural Networks. In: Bao-Liang LU, Liqing ZHANG a James KWOK, ed. *Neural Information Processing* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011 [vid. 2018-06-12], s. 550–559. ISBN 978-3-642-24954-9. Dostupné z: doi:10.1007/978-3-642-24955-6_66

[27] Kopecký, M.: Simulační prostředí pro webové prohlížeče, Master's degree thesis (Supervisor Ivo Bukovsky), Faculty of Mechanical Engineering, CTU in Prague 2007

[28] Peter M. Beneš: Software Application for Adaptive Identification and Controller Tuning (Supervisor Ivo Bukovsky) Student's Conference STC, Faculty of Mechanical Engineering, CTU in Prague 2013