Process control by Learning entropy

Stability conditions

Gejza Dohnal, Czech Technical University in Prague

Process control by Learning entropy

Stability conditions

Gejza Dohnal, Czech Technical University in Prague

Energy Days 2025

I. The basic starting point

A system is a whole composed of parts that interact with each other. Information, matter, and energy can flow between parts of a system. (Wikipedia)

- The system can be in different states over time.
- The state of the system is a measurable (observable) quantity Y, which can have several components - it is an element of the space R^n.
- The states of the system over time constitute a random process.
- We assume that the state of the system depends on the internal variables (factors) of the system and usually includes a random component caused by external, unpredictable influences.

A system is a whole composed of parts that interact with each other. Information, matter, and energy can flow between parts of a system. (Wikipedia)

- The system can be in different states over time.
- The state of the system is a measurable (observable) quantity Y, which can have several components - it is an element of the space R^n.
- The states of the system over time constitute a random process.
- We assume that the state of the system depends on the internal variables (factors) of the system and usually includes a random component caused by external, unpredictable influences.
- **The behavior** of the system is characterized by a random process of its states $\{Y_t\}_{t\geq 0}$
- System behaviour is usually monitored at discrete moments $Y_{t_0}, Y_{t_1}, \ldots, Y_{t_i}, \cdots \subset R^n$
- A system "behaves well" when the corresponding process of its states "behaves well".
    - A process behaves well if it is predictable.
        - If the process is predictable, we can find its model.

Everything that can go wrong will go wrong.

[Edward A. Murphy, 1978]

Schnatterly's summary of all the implications:

If something can't go wrong, it will.

(The real life application of the second „law“ of thermodynamics, determining the natural direction in which natural processes proceed.)

Everything that can go wrong will go wrong.

[Edward A. Murphy, 1978]

Schnatterly's summary of all the implications:

If something can't go wrong, it will.

(The real life application of the second „law“ of thermodynamics, determining the natural direction in which natural processes proceed.)

British mathematician Augustus De Morgan (pictured circa 1860) wrote in 1866 that "whatever can happen will happen".

Changes to the system will usually result in a change to the model

➡ its behavior will change and its predictability will be broken.

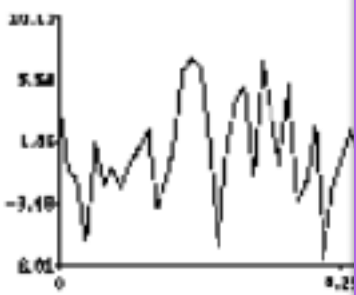
British stage magician Nevil Maskelyne wrote in 1908 that, during special occasions, "everything that can go wrong will go wrong".

II. What about it?

If we want to keep the running system under control, we basically have two options:

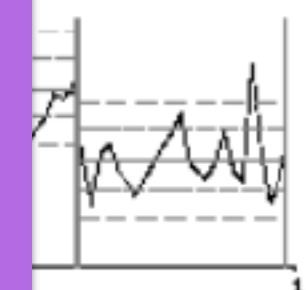
off-line detection: consists in finding points of change in "historical" data. This means that we have available observations from some period (0, T) and we find out whether one or more changes in the behavior of the process (violation of homogeneity) occurred in this period.

The most frequently used tools are: maximum likelihood method, Bayesian approach, simulation, permutation methods and others



> To consult the statistician after an experiment is finished is often merely to ask him to conduct a post mortem examination. He can perhaps say what the experiment died of.

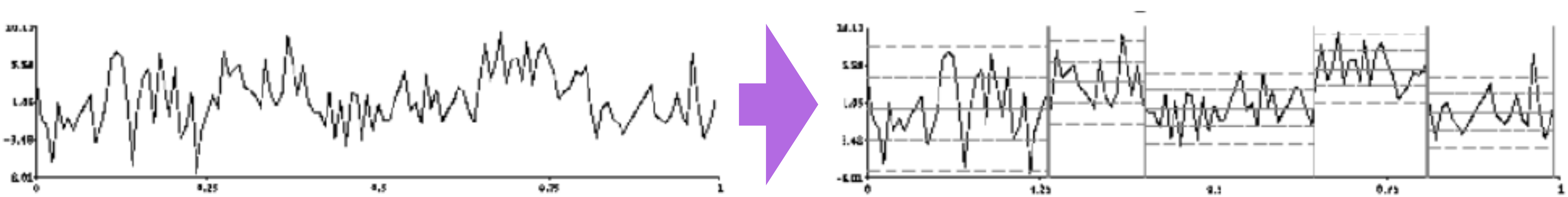
Ronald Fischer (1890 - 1962)



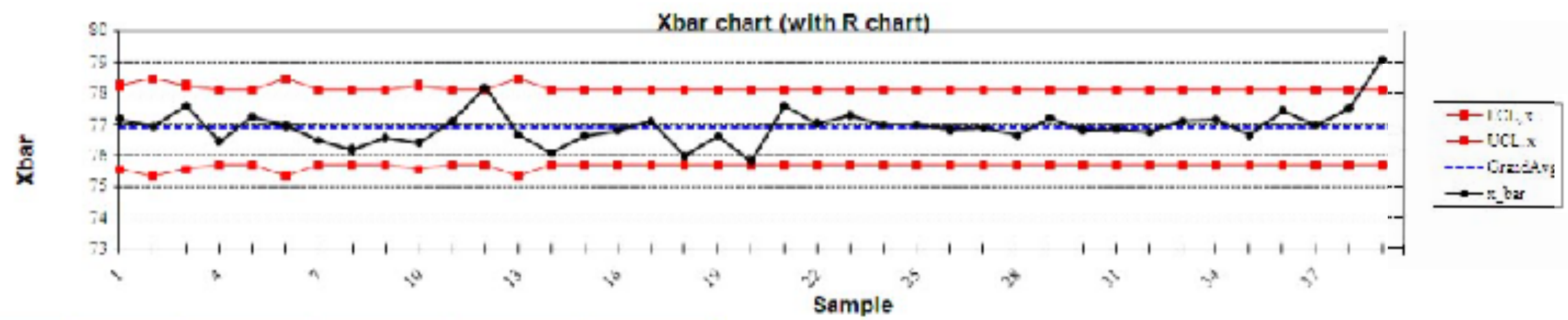
If we want to keep the running system under control, we basically have two options:

off-line detection: consists in finding points of change in "historical" data. This means that we have available observations from some period (0, T) and we find out whether one or more changes in the behavior of the process (violation of homogeneity) occurred in this period.

The most frequently used tools are: maximum likelihood method, Bayesian approach, simulation, permutation methods and others



on-line detection:

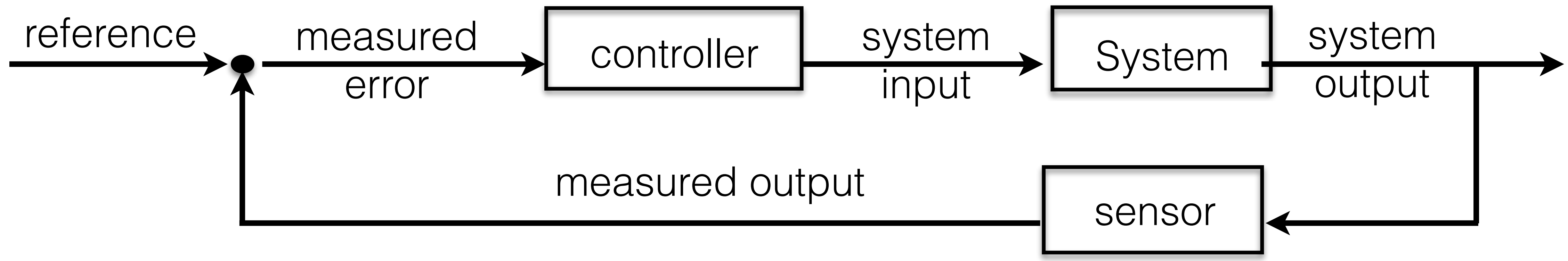


it is characterized by the fact that the values of the state variables are currently arriving in time based on the monitoring ring, and our task is to detect a change from the previous behavior as soon as possible, if it occurs.

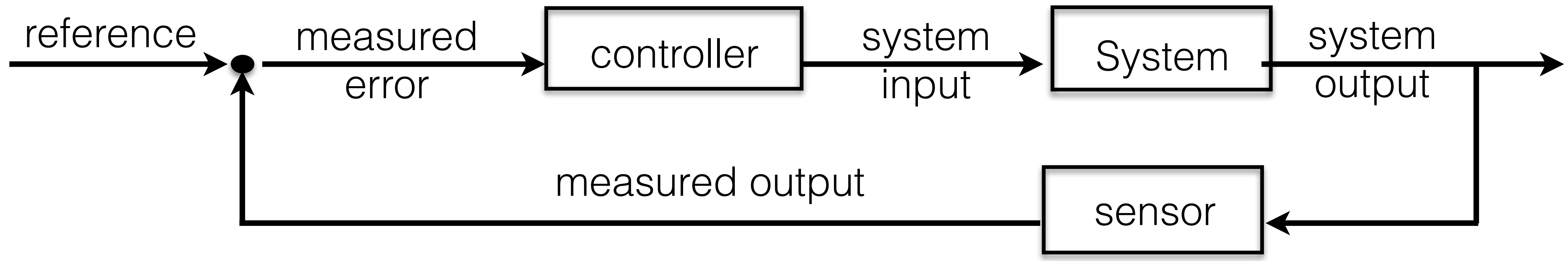
The most commonly used tool is statistical process control (SPC) and control charts

III. On-line problem

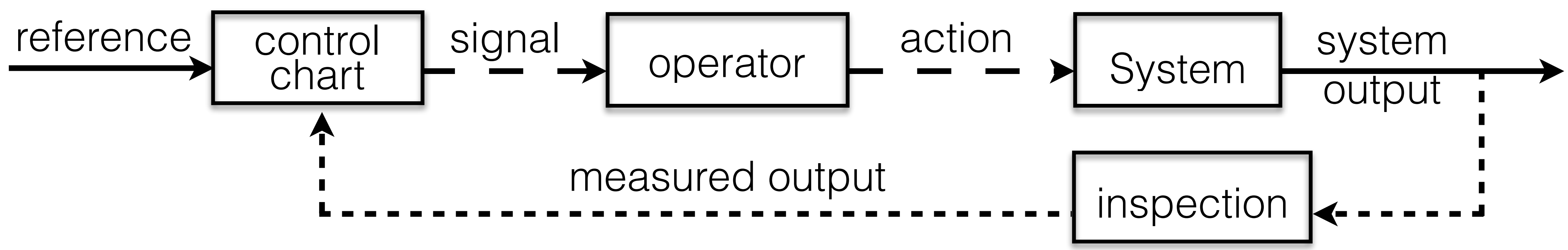
1. **Control the system automatically** (automatically respond to changes in behavior and maintain the system in a steady state)



1. **Control the system automatically** (automatically respond to changes in behavior and maintain the system in a steady state)



2. **Detect a change in behavior and take action** - return the system to an in-control state.



We assume d-dimensional random process $\boldsymbol{X}(t)$, $t \geqq 0$, which follows some model $\boldsymbol{X}(t) = \boldsymbol{h}(t) + \xi(t)$, $t \geqq 0$, where $\boldsymbol{h}(t)$ is d-dimensional real function, generally unknown, and $\xi(t)$ is some d-dimensional centered weak stationary random process.

Then we suppose there exists an unknown time point $T > 0$ such that for $T \leqq t$ the process $\boldsymbol{X}(t)$ follows a model $\boldsymbol{X}(t) = \boldsymbol{g}(t) + \psi(t)$ with some (usually unknown) function $\boldsymbol{g}(t)$ and some random process $\psi(t)$.

$\boldsymbol{h}(t)$ is known => $\boldsymbol{Y}(t) = \boldsymbol{X}(t) - \boldsymbol{h}(t)$ we can use classical methods (e.g. SPC)

We assume d-dimensional random process $\boldsymbol{X}(t)$, $t \geqq 0$, which follows some model $\boldsymbol{X}(t) = \boldsymbol{h}(t) + \xi(t)$, $t \geqq 0$, where $\boldsymbol{h}(t)$ is d-dimensional real function, generally unknown, and $\xi(t)$ is some d-dimensional centered weak stationary random process.

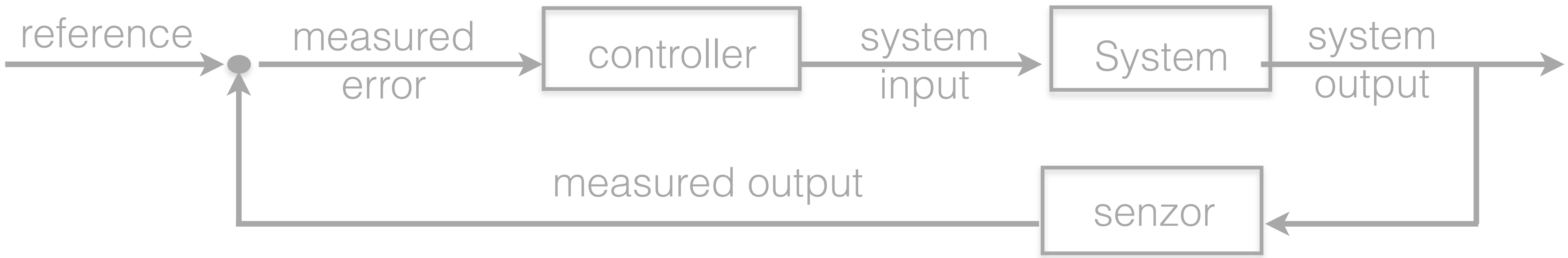
Then we suppose there exists an unknown time point $T > 0$ such that for $T \leqq t$ the process $\boldsymbol{X}(t)$ follows a model $\boldsymbol{X}(t) = \boldsymbol{g}(t) + \psi(t)$ with some (usually unknown) function $\boldsymbol{g}(t)$ and some random process $\psi(t)$.

$\boldsymbol{h}(t)$ is known => $\boldsymbol{Y}(t) = \boldsymbol{X}(t) - \boldsymbol{h}(t)$ we can use classical methods (e.g. SPC)

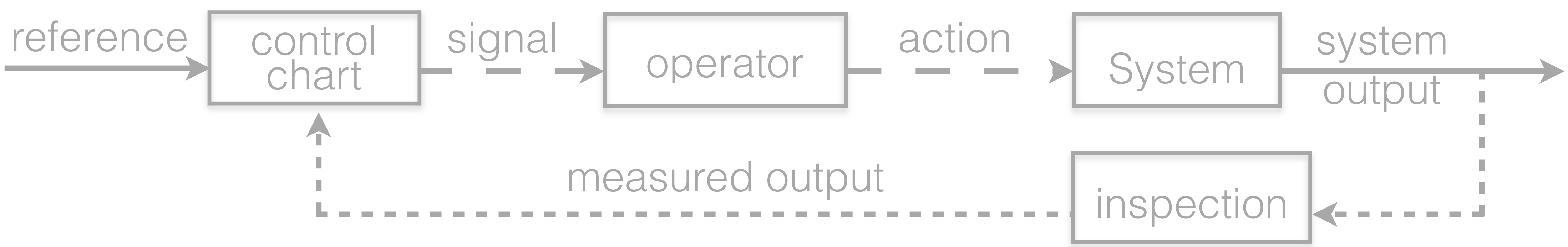
$\boldsymbol{h}(t)$ is unknown => $\boldsymbol{X}(t)$ is nonstationary and we cannot use classical methods

For detection of a change, we need some method for $\boldsymbol{h}(t)$ identification.

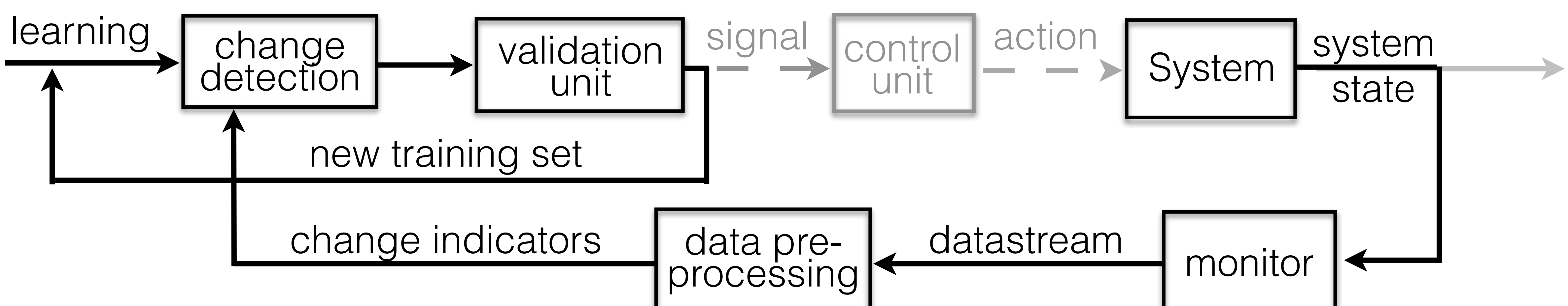
1. **Control the system automatically** (automatically respond to changes in behavior and maintain the system in a steady state) Automatic control reacts to a specific time series and usually uses a deterministic approach.



2. **Detect a change in behavior and take action** - return the system to a in-control state. This always requires a statistical approach - SPC, SDS, ….



3. **Monitoring Process and Anomaly (Change) Detection based on machine learning**



IV. Entropy-based detection

Entropy based detection

Shanon-based Entropies are data-window and probabilistic based computations that are widely used for time series analyses.

- **Sample Entropy** is a signal complexity evaluation algorithm (floating window based quantification of signal complexity, probability-based approach).
  [S. M. Pincus (1991): *Approximate entropy as a measure of system complexity.* Proc. Nat. Acad. Sci. U.S.A., **88**]

- **Entropy Learning** is a Shannon-inspired neural network learning algorithm based on minimizing complexity (entropy) of neural weights in a network.
  [Gajowniczek, K.; Orłowski, A.; Ząbkowski, T. (2018): *Simulation Study on the Application of the Generalized Entropy Concept in Artificial Neural Networks*. Entropy, **20**]

Entropy based detection

Shanon-based Entropies are data-window and probabilistic based computations that are widely used for time series analyses.

- **Sample Entropy** is a signal complexity evaluation algorithm (floating window based quantification of signal complexity, probability-based approach).
  [S. M. Pincus (1991): *Approximate entropy as a measure of system complexity.* Proc. Nat. Acad. Sci. U.S.A., **88**]

- **Entropy Learning** is a Shannon-inspired neural network learning algorithm based on minimizing complexity (entropy) of neural weights in a network.
  [Gajowniczek, K.; Orłowski, A.; Ząbkowski, T. (2018): *Simulation Study on the Application of the Generalized Entropy Concept in Artificial Neural Networks*. Entropy, **20**]

Learning Entropy is a non-Shannon based novelty detection algorithm based on observation of unusual learning effort of incrementally learning systems. LE is a relative measure of novelty (information) recognized as unusual learning effort of pre-trained learning system on individual data samples.

[I. Bukovsky (2013):*Learning Entropy: Multiscale Measure for Incremental Learning*. Entropy, **15**
G. Dohnal, I. Bukovsky (2020): Novelty detection based on learning entropy. ASMBI, 36]

Learning entropy based detection

Apply some adaptive learning system in a form of neural network:

The function $\tilde{y}(t+\delta) = f(\mathbf{x}(t), \mathbf{w}(t))$ such that for any time $t \geqq 0$ and a given horizon $\delta > 0$ minimize the error $|X(t+\delta) - f(\mathbf{x}(t), \mathbf{w}(t))|$ we call a predictor.

For a change detection, we can use as a predictor a neural network based on so called high order neural unit (HONU):

LNU: $\tilde{y}(t) = \mathbf{x}(t)\mathbf{w}(t)$ (linear)

QNU: $\tilde{y}(t) = \mathbf{x}^{\mathrm{T}}(t)\mathbf{W}(t)\mathbf{x}(t)$ (quadratic)

$\mathbf{w}(t)$ is n-dimensional vector of weights (adaptive parameters):

$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$, where $\triangle\mathbf{w}(t) = -\frac{\mu}{2}\frac{\partial e(t)^2}{\partial \mathbf{w}}$

static gradient descent (GD) algorithm

LNU: $\triangle\mathbf{w}(t) = \mu e(t)\mathbf{x}(t)$

QNU: $\triangle\mathbf{W}(t) = \mu e(t)\mathbf{x}(t)\mathbf{x}(t)^T$

Learning entropy based detection

$\{\mathbf{x}(l\text{-}M), \ldots, \mathbf{x}(l\text{-}1), \mathbf{x}(l)\}$ => $\{\mathbf{w}(l\text{-}M), \ldots, \mathbf{w}(l\text{-}1), \mathbf{w}(l)\}$

$\Delta\mathbf{w}(k) = \mathbf{w}(k+1) - \mathbf{w}(k)$ => $\{\Delta\mathbf{w}(l\text{-}M), \ldots, \Delta\mathbf{w}(l\text{-}1)\}$

$$|\triangle\bar{\mathbf{w}}(l)| = \frac{1}{M}\sum_{i=1}^{M}|\triangle\mathbf{w}(l-i)|$$

Learning Entropy: $$E_A(k) = \frac{1}{n_A n_w}\sum_{j=1}^{n_A}\sum_{i=1}^{n_w} I_{(|\triangle w_i(k)|>\alpha_j|\triangle\bar{w}_i(k)|)}$$

where $A = (\alpha_1, \ldots, \alpha_{n_A})$ is n_A-dimensional vector of detection sensitivities.

Note: We use $\Delta\mathbf{w}(t)$ for detection rather than prediction error due to its higher sensitivity.

Until $\boldsymbol{X}(t) = \boldsymbol{h}(t) + \xi(t)$, the predictor learns and $\{\Delta\mathbf{w}(t)\}$ stabilises („settles down“)

After some change, when $\boldsymbol{X}(t) = \boldsymbol{g}(t) + \psi(t)$, the process $\{\Delta\mathbf{w}(t)\}$ starts to oscillate (the predictor tries to adapt to a new model)

=> the change can be detected using classical methods applied to the process $\{\Delta\mathbf{w}(t)\}$.

=> **Stability is crucial**

Learning entropy based detection - example

For illustration, we use data generated by the process

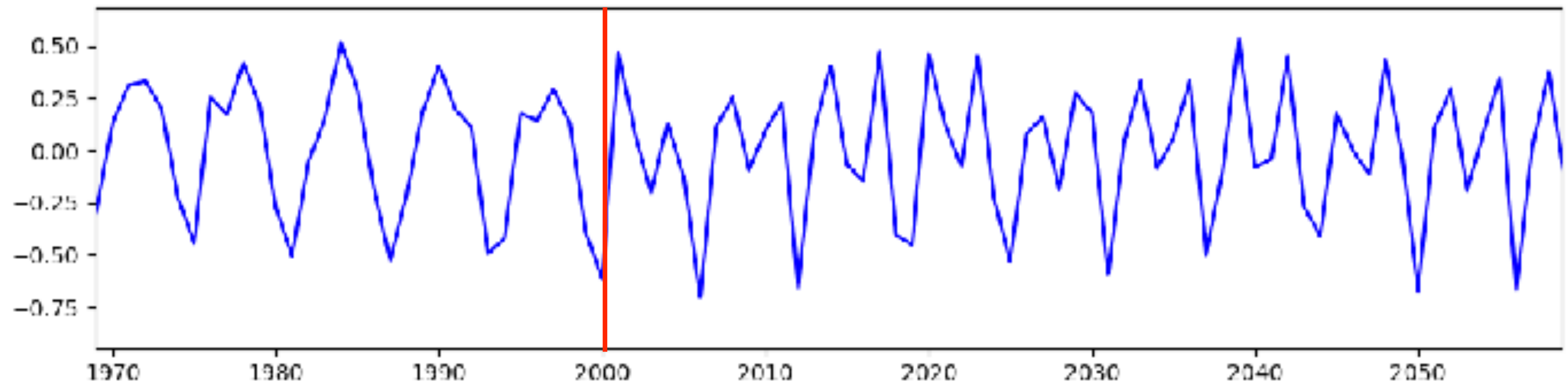
$$X(t) = a\sin^4 t + b\cos^3 t + c\cos^2 t + d\sin t + e + \xi$$

with parameters $a = 2$, $b = -1$, $c = 3$, $d = -1$, $e = 0$ for $t \geqq 0$ and random fluctuations $\xi \backsim$ N(0, 1/3).

$$X(t) = 2\sin^4 t - \cos^3 t + 3\cos^2 t - \sin t + \xi, \quad 0 \le t < 2000, \quad \xi \sim N(0, 1/3)$$

The change is simulated in the time t = 2000, from which onwards is a = 0, d = 0 and e = -0.7.

$$X(t) = -\cos^3 t + 3\cos^2 t - 0.7 + \xi, \quad 2000 \le t, \quad \xi \sim N(0, 1/3)$$

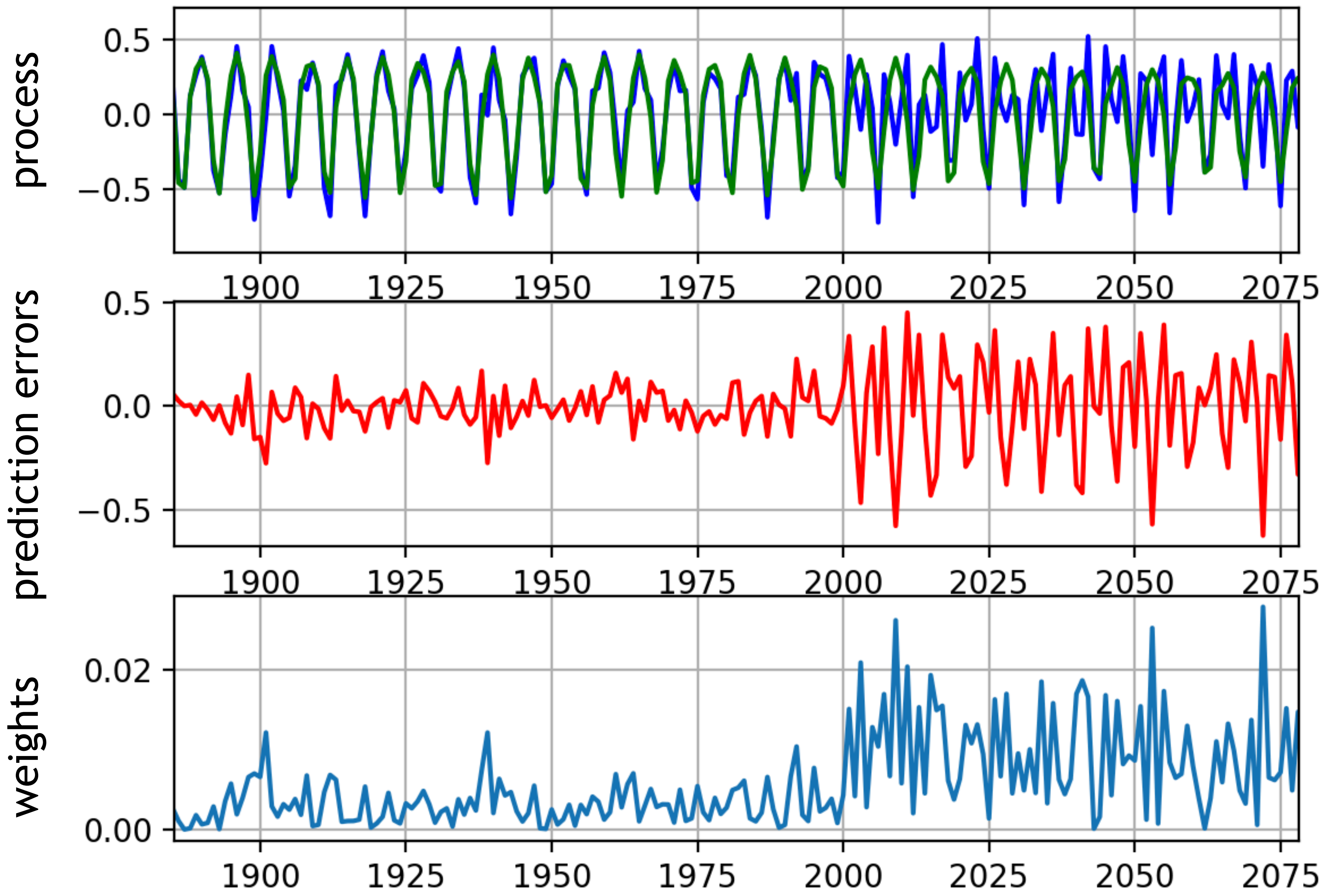


$\{X(t),\ t \geqq \mathrm{T}\}$ is no more $\boldsymbol{h}$-stationary => $\{\Delta \mathbf{w}(t)\}$ loses its stationarity

Learning entropy based detection - example

$$X(t) = 2\sin^4 t - \cos^3 t + 3\cos^2 t - \sin t + \xi, \quad 0 \le t < 2000, \quad \xi \sim N(0, 1/3)$$

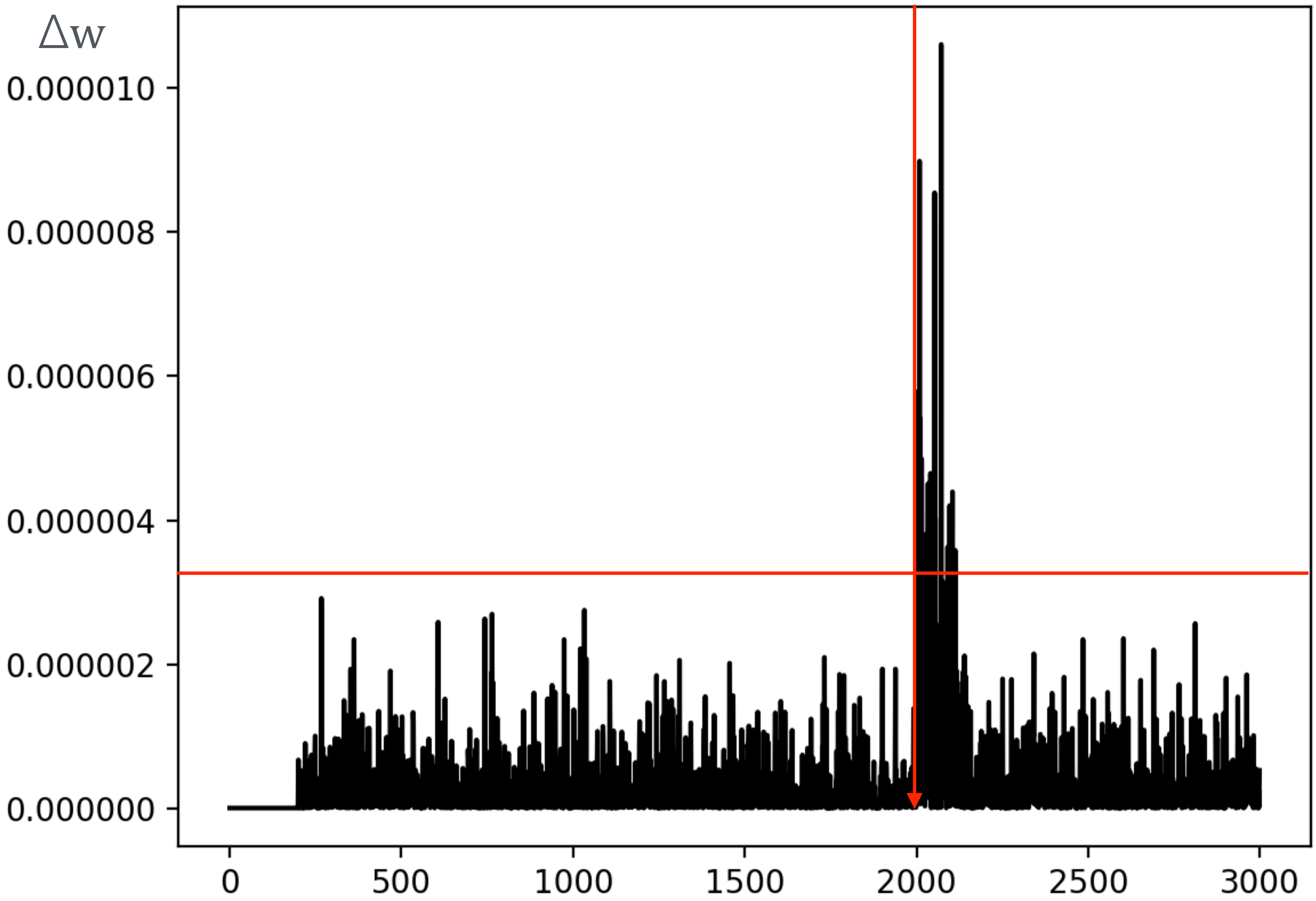
$$X(t) = -\cos^3 t + 3\cos^2 t - 0.7 + \xi, \quad 2000 \le t, \quad \xi \sim N(0, 1/3)$$



Learning entropy based detection - example

$$X(t) = 2\sin^4 t - \cos^3 t + 3\cos^2 t - \sin t + \xi, \quad 0 \leq t < 2000, \quad \xi \sim N(0, 1/3)$$

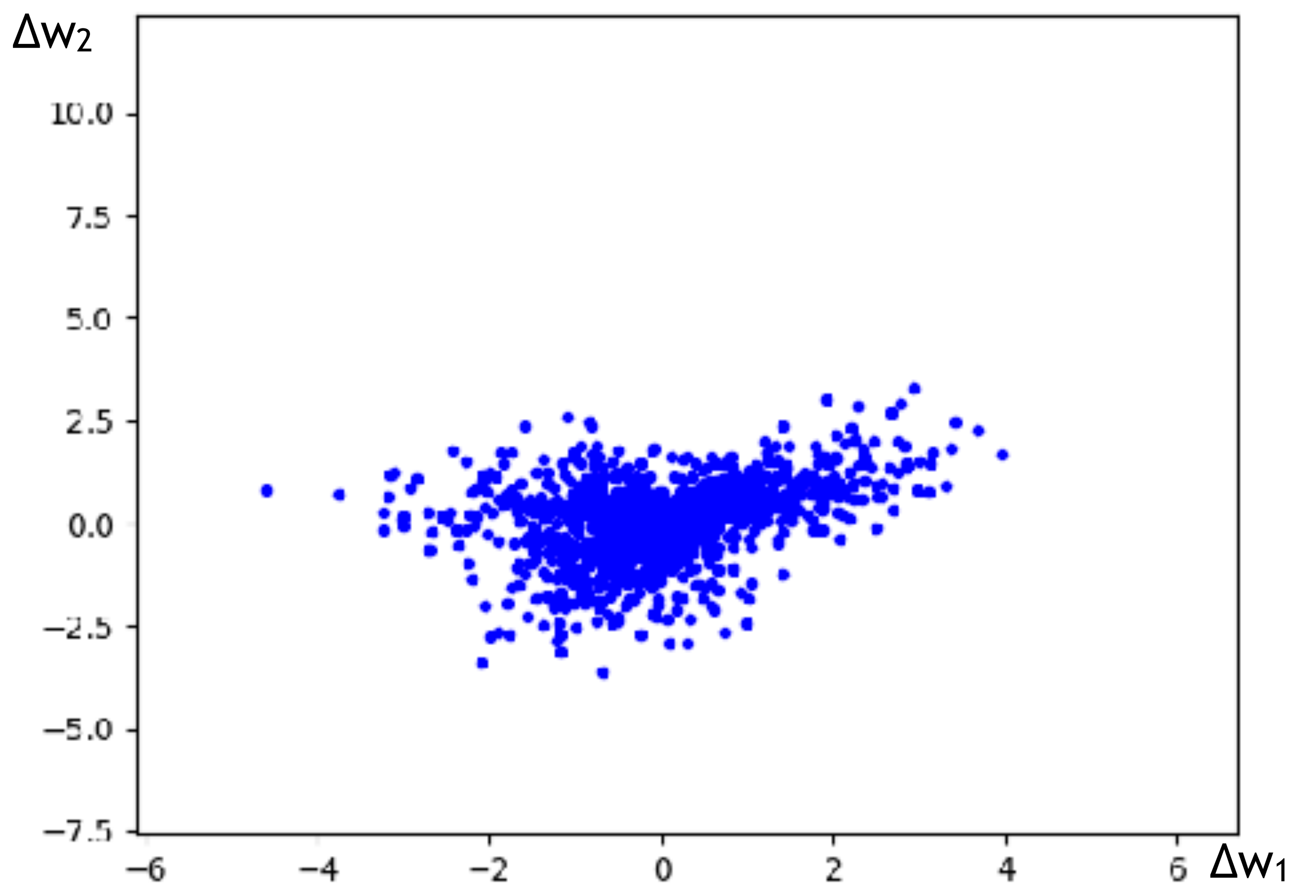
$$X(t) = -\cos^3 t + 3\cos^2 t - 0.7 + \xi, \quad 2000 \leq t, \quad \xi \sim N(0, 1/3)$$



Learning entropy based detection - example

Hotelling's t-square test:

1) In the Phase I: Observe $\{\mathbf{x}(l\text{-}M), \ldots, \mathbf{x}(l\text{-}1), \mathbf{x}(l)\}$ and compute $\{\mathbf{w}(l\text{-}M), \ldots, \mathbf{w}(l\text{-}1), \mathbf{w}(l)\}$, evaluate the centroid $|\Delta\mathbf{w}(l)|$ and sample correlation matrix $\boldsymbol{\Sigma}$
2) For $k = l+1, \ldots$ evaluate the predictor $y(k) = f(\mathbf{x}(k\text{-}1), \mathbf{w}(k\text{-}1))$ and compute the Hotelling's t^2-statistics $t^2 = (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))^T \Sigma^{-1} (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))$
3) $t^2 \sim T^2_{n,M-1} = \frac{n(M-1)}{M-n} F_{n,M-n}$



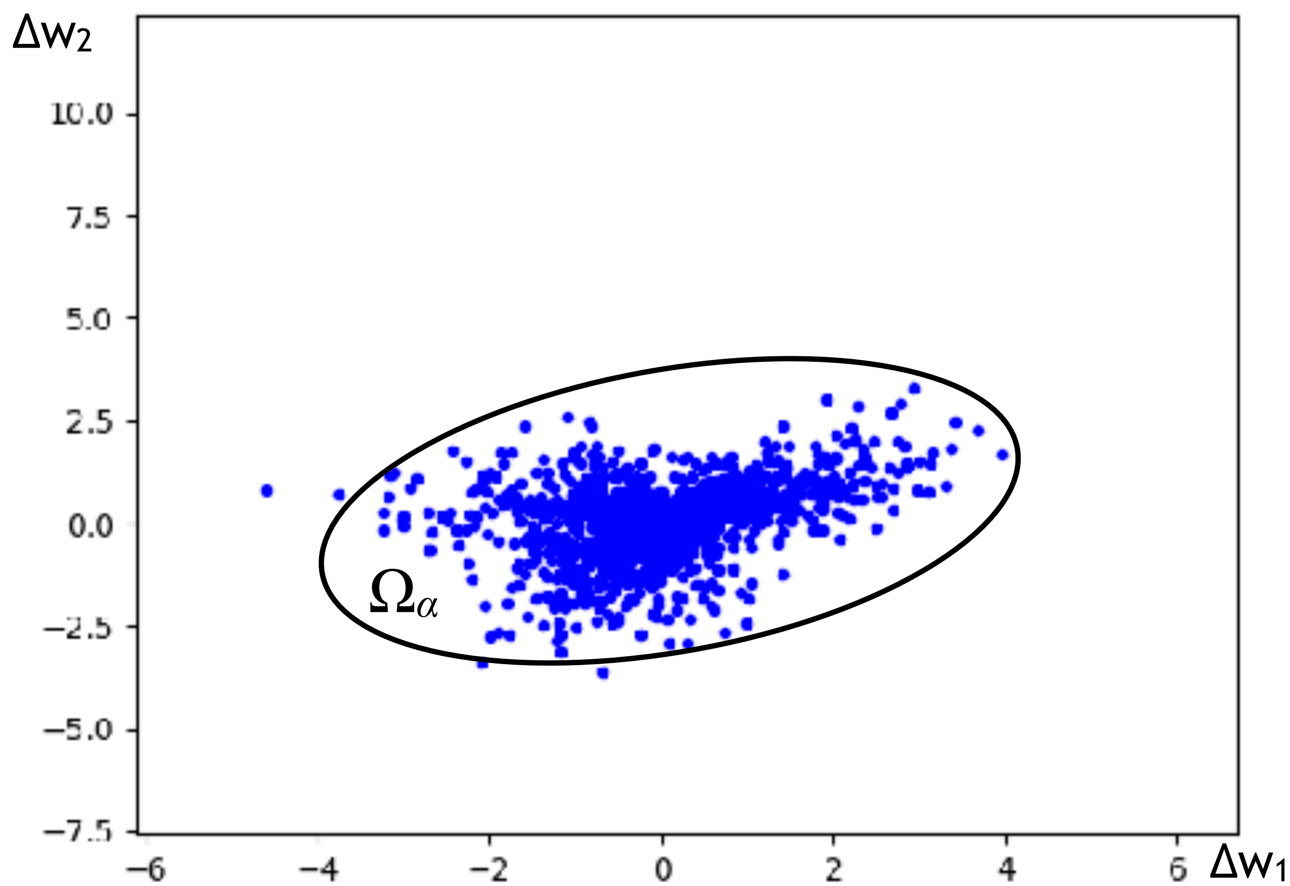
Learning entropy based detection - example

Hotelling's t-square test:

1) In the Phase I: Observe $\{\mathbf{x}(l\text{-}M), \ldots, \mathbf{x}(l\text{-}1), \mathbf{x}(l)\}$ and compute $\{\mathbf{w}(l\text{-}M), \ldots, \mathbf{w}(l\text{-}1), \mathbf{w}(l)\}$, evaluate the centroid $|\Delta\mathbf{w}(l)|$ and sample correlation matrix $\boldsymbol{\Sigma}$

2) For $k = l+1, \ldots$ evaluate the predictor $y(k) = f(\mathbf{x}(k\text{-}1), \mathbf{w}(k\text{-}1))$ and compute the Hotelling's t^2-statistics $t^2 = (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))^T\Sigma^{-1}(|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))$

3) $t^2 \sim T^2_{n,M-1} = \frac{n(M-1)}{M-n}F_{n,M-n}$

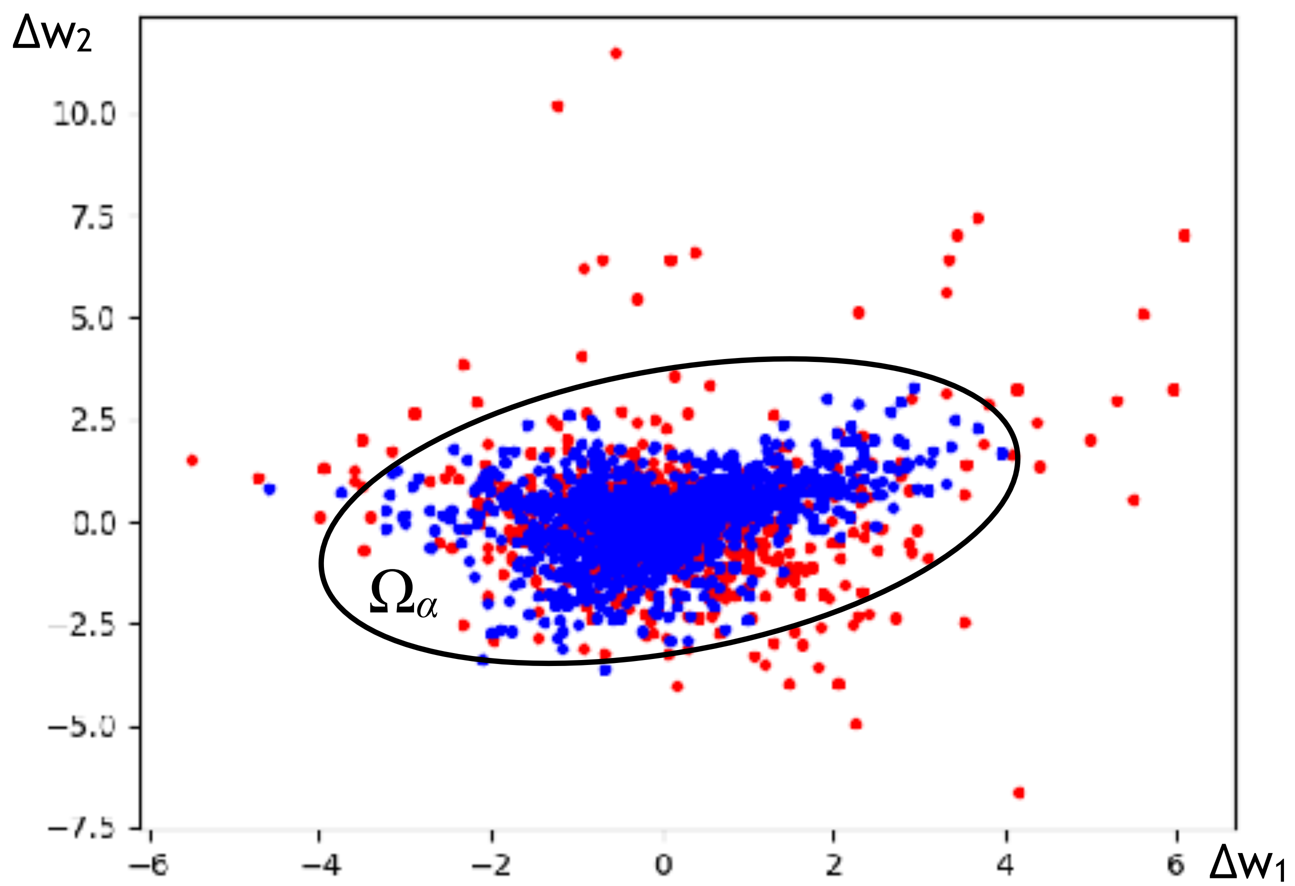


$$\Omega_\alpha(k) = \{\, \mathbf{v} \in \mathbb{R}^n : \ (\mathbf{v} - |\triangle\bar{\mathbf{w}}(k)|)^T\Sigma^{-1}(\mathbf{v} - |\triangle\bar{\mathbf{w}}(k)|) \le \omega_\alpha^2 \,\}$$

Learning entropy based detection - example

Hotelling's t-square test:

1) In the Phase I: Observe $\{\mathbf{x}(l\text{-}M), \dots, \mathbf{x}(l\text{-}1), \mathbf{x}(l)\}$ and compute $\{\mathbf{w}(l\text{-}M), \dots, \mathbf{w}(l\text{-}1), \mathbf{w}(l)\}$, evaluate the centroid $|\Delta\mathbf{w}(l)|$ and sample correlation matrix $\mathbf{\Sigma}$
2) For $k = l{+}1, \dots$ evaluate the predictor $y(k) = f(\mathbf{x}(k\text{-}1), \mathbf{w}(k\text{-}1))$ and compute the Hotelling's t^2-statistics $t^2 = (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))^T \Sigma^{-1} (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))$
3) $t^2 \sim T^2_{n,M-1} = \frac{n(M-1)}{M-n} F_{n,M-n}$



$$\Omega_\alpha(k) = \{\, \mathbf{v} \in \mathbb{R}^n : \ (\mathbf{v} - |\triangle\bar{\mathbf{w}}(k)|)^T \Sigma^{-1} (\mathbf{v} - |\triangle\bar{\mathbf{w}}(k)|) \le \omega^2_\alpha \,\}$$

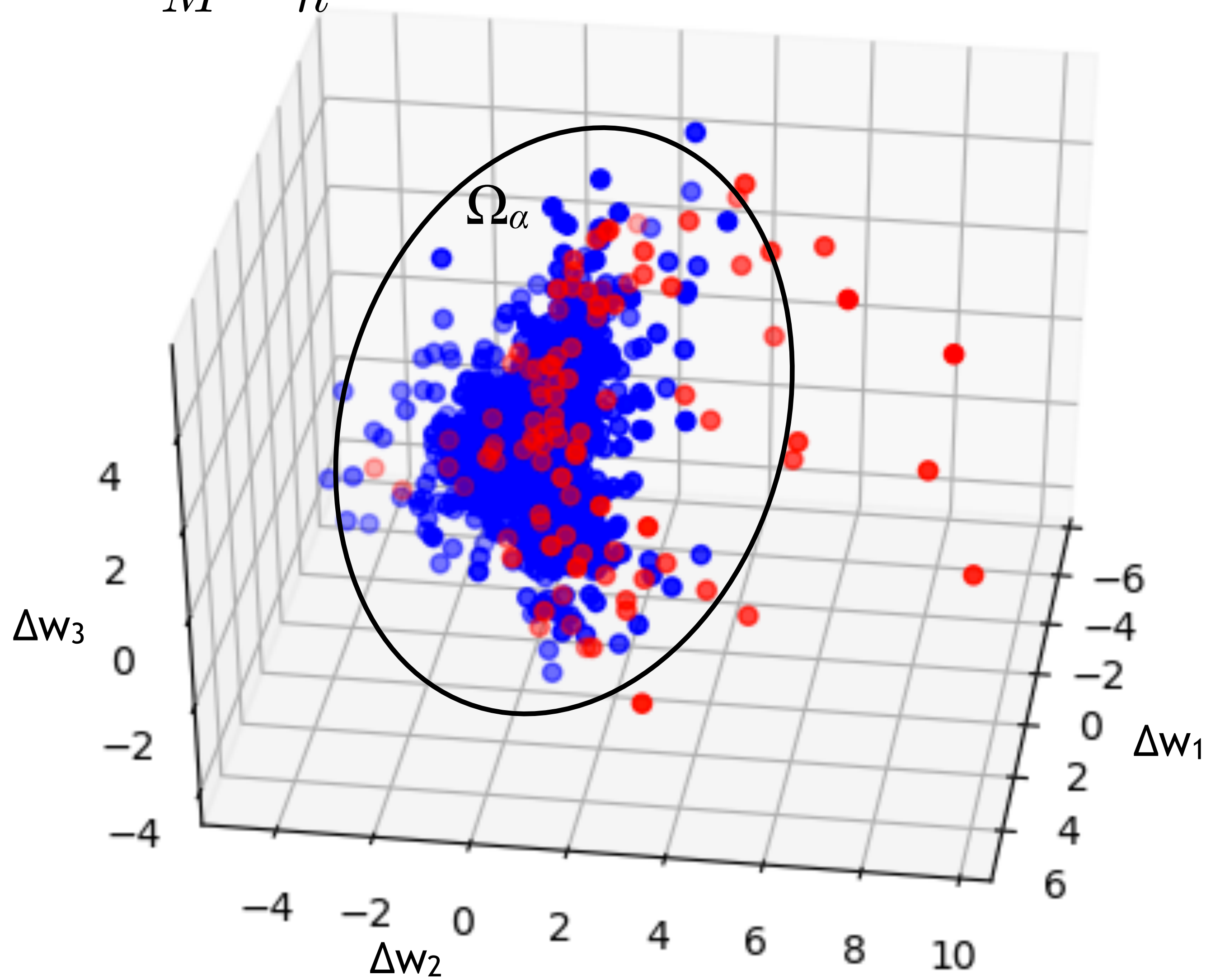
Learning entropy based detection - example

Hotelling's t-square test:

1) In the Phase I: Observe $\{\mathbf{x}(l\text{-}M), \ldots, \mathbf{x}(l\text{-}1), \mathbf{x}(l)\}$ and compute $\{\mathbf{w}(l\text{-}M), \ldots, \mathbf{w}(l\text{-}1), \mathbf{w}(l)\}$, evaluate the centroid $|\Delta\mathbf{w}(l)|$ and sample correlation matrix $\mathbf{\Sigma}$

2) For $k = l+1, \ldots$ evaluate the predictor $y(k) = f(\mathbf{x}(k\text{-}1), \mathbf{w}(k\text{-}1))$ and compute the Hotelling's t^2-statistics $t^2 = (|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))^T\Sigma^{-1}(|\triangle\bar{\mathbf{w}}(l)| - \triangle\mathbf{w}(k))$

3) $t^2 \sim T^2_{n,M-1} = \dfrac{n(M-1)}{M-n} F_{n,M-n}$



V. Stability

Stability of learning entropy based detection

We assume d-dimensional random process $\boldsymbol{X}(t)$, $t \geqq 0$, which follows some model $\boldsymbol{X}(t) = \boldsymbol{h}(t) + \xi(t)$, $t \geqq 0$, where $\boldsymbol{h}(t)$ is d-dimensional real function, generally unknown, and $\xi(t)$ is some d-dimensional centered weak stationary random process.

Until $\boldsymbol{X}(t) = \boldsymbol{h}(t) + \xi(t)$, the predictor learns and $\{\Delta \mathbf{w}(t)\}$ stabilises („settles down“)

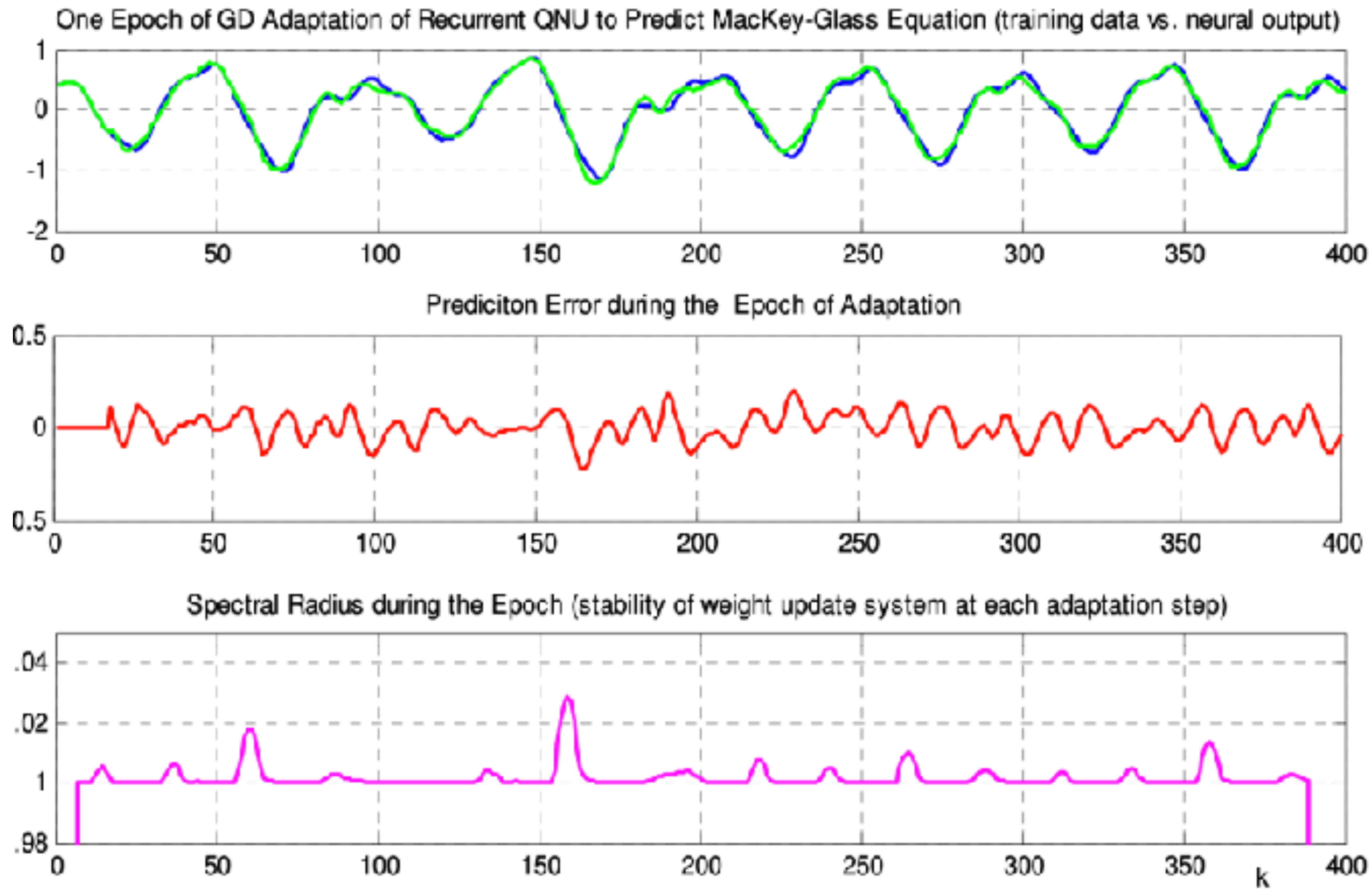
Then there comes a moment $T > 0$ and for $t \geqq T$ the process $\boldsymbol{X}(t)$ follows a new model $\boldsymbol{X}(t) = \boldsymbol{g}(t) + \psi(t)$ with some (usually unknown) function $\boldsymbol{g}(t)$ and some random process $\psi(t)$.

After the change occurs, the process $\{\Delta \mathbf{w}(t)\}$ starts to oscillate (the predictor tries to adapt to a new model)

In online adaptive learning it is crucial that the weights of a neural network do not grow uncontrollably. The stability of the weights ensures that the system is usable in real time and does not diverge. => **Stability of the predictor is crucial**

Role of the stability of a weight-update system - example

One epoch of gradient descent (GD) adaptation of recurrent QNU to predict MacKey-Glass equation (training data vs. neural output) [M. C. Mackey, L. Glass (1977): Oscillation and chaos in physiological control systems. Science, vol. 197, no. 4300, pp. 287–289]

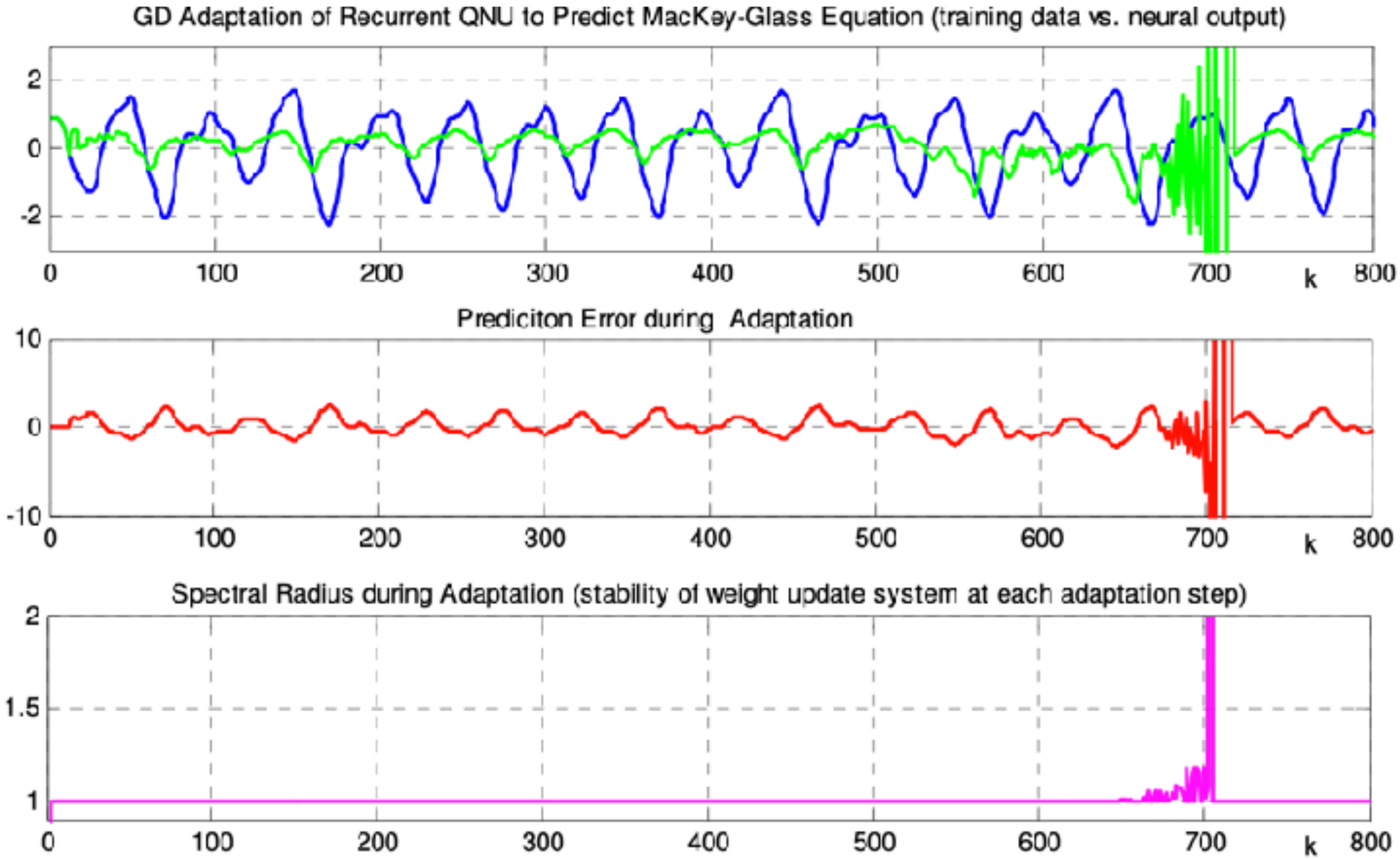


Stable adaptation of recurrent HONU (r=2); the bottom plot monitors the stability (i.e. the spectral radius) of the weight-update system using GD.

[Ivo Bukovsky, Noriasu Homma (2017): *An Approach to Stable Gradient Descent Adaptation of Higher-Order Neural Units. IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2022-2034]

Role of the stability of a weight-update system - example

One epoch of gradient descent (GD) adaptation of recurrent QNU to predict MacKey-Glass equation (training data vs. neural output) [M. C. Mackey, L. Glass (1977): Oscillation and chaos in physiological control systems. Science, vol. 197, no. 4300, pp. 287–289]

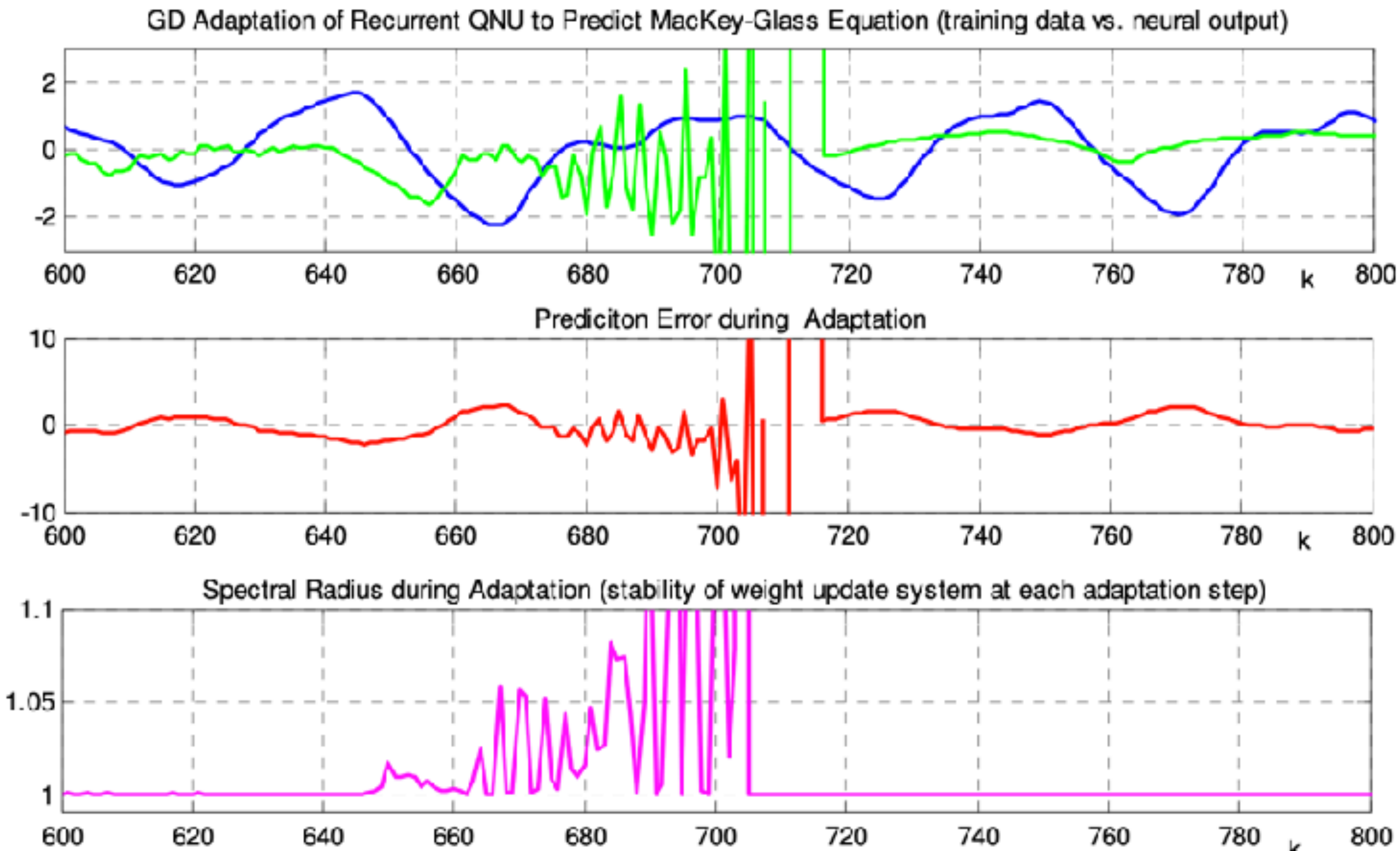


… instability of weights originates at around of k=650

[Ivo Bukovsky, Noriasu Homma (2017): *An Approach to Stable Gradient Descent Adaptation of Higher-Order Neural Units. IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2022-2034]

Role of the stability of a weight-update system - example

One epoch of gradient descent (GD) adaptation of recurrent QNU to predict MacKey-Glass equation (training data vs. neural output) [M. C. Mackey, L. Glass (1977): Oscillation and chaos in physiological control systems. Science, vol. 197, no. 4300, pp. 287–289]



Unstable adaptation (detail of previous figure): the stability became significantly violated well before unusually large oscillations and divergence of neural output.

[Ivo Bukovsky, Noriasu Homma (2017): *An Approach to Stable Gradient Descent Adaptation of Higher-Order Neural Units. IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2022-2034]

Stability of learning entropy based detection

HONU is generally in-parameter linear nonlinear architecture (IPLNA) and the predictor is in the form:

$$\tilde{y}(k+1) = \mathbf{w}^T(k) \cdot \mathbf{g}\left(\mathbf{x}(\mathbf{v}, k)\right)$$

In this case, the time-variant state-space representation of such system is in the form:

$$\mathbf{w}(k+1) = \mathbf{A}(k) \cdot \mathbf{w}(k) + \mathbf{B}(k) \cdot \mathbf{u}(k), \quad k > 0, \quad \mathbf{w}(k_0) = \mathbf{0}.$$

It can be shown, that stability of the IPLNA system depends on a properties of the local matrix of dynamics **A**(*k*), which is in the form:

$$\mathbf{A}(k) = \left(\mathbf{I} - \eta(k) \cdot \mathbf{g}(\mathbf{x}(k)) \cdot \mathbf{g}(\mathbf{x}(k))^T\right)$$

In the case of classical stability (e.g., Ljapunov), we ask $\rho(\mathbf{A}(k)) < 1$.

[G. Dohnal, I. Bukovsky, P. M. Benes, K. Ichiji and N. Homma (2020): *Letter on Convergence of In-Parameter-Linear Nonlinear Neural Architectures With Gradient Learnings.* IEEE Transactions on Neural Networks and Learning Systems]

Stability of learning entropy based detection

HONU is generally in-parameter linear nonlinear architecture (IPLNA) and the predictor is in the form:

$$\tilde{y}(k+1) = \mathbf{w}^T(k) \cdot \mathbf{g}\left(\mathbf{x}(\mathbf{v}, k)\right)$$

In this case, the time-variant state-space representation of such system is in the form:

$$\mathbf{w}(k+1) = \mathbf{A}(k) \cdot \mathbf{w}(k) + \mathbf{B}(k) \cdot \mathbf{u}(k), \quad k > 0, \quad \mathbf{w}(k_0) = \mathbf{0}.$$

It can be shown, that stability of the IPLNA system depends on a properties of the local matrix of dynamics **A**(*k*), which is in the form:

$$\mathbf{A}(k) = \left(\mathbf{I} - \eta(k) \cdot \mathbf{g}(\mathbf{x}(k)) \cdot \mathbf{g}(\mathbf{x}(k))^T\right)$$

In the case of classical stability (e.g., Ljapunov), we ask $\rho(\mathbf{A}(k)) < 1$.

For the IPLNA, the bounded-input-bounded-state stability (BIBS) is defined as follows:

If there exist two positive constants $0 < L_u, L_w < \infty$ such that for all $k \geq k_0$

$$||\mathbf{u}(k)|| \leq L_u \;\Rightarrow ||\mathbf{w}(k)|| \leq L_w$$

then the system is BIBS.

[G. Dohnal, I. Bukovsky, P. M. Benes, K. Ichiji and N. Homma (2020): *Letter on Convergence of In-Parameter-Linear Nonlinear Neural Architectures With Gradient Learnings.* IEEE Transactions on Neural Networks and Learning Systems]

Convergence of Neural Architectures

The bounded-input bounded-state (BIBS) stability concept is recently popular in neural networks. We introduce the Bounded-input bounded-state stability (BIBS) concept for weight convergence of a broad family of incremental gradient learning IPLNAs.

We can show that the input-to-state stability (ISS) concept and BIBS stability generally apply to the gradient learning algorithms and their many modifications for IPLNAs.

Let us consider a IPLNAs system with time-variant state-space representation in the form

$$\mathbf{w}(k+1) = \mathbf{A}(k) \cdot \mathbf{w}(k) + \mathbf{B}(k) \cdot \mathbf{u}(k)$$

Definition 2: The system is *BIBS stable* if there exist positive constants $0 < L_{\mathrm{u}}, L_{\mathrm{w}} < \infty$, such that the conditions $\mathbf{w}(k_0) = \mathbf{0}, \quad ||\mathbf{u}(k)|| \leq L_u \quad \forall k > k_0$ imply that

$$||\mathbf{w}(k)|| \leq L_w \quad \forall k > k_0$$

.

The system is *BIBO stable* if there exist constants $0 < L_{\mathrm{u}}, L_{\mathrm{y}} < \infty$, such that the above condition implies that $||y(k)|| \leq L_y \quad \forall k > k_0$.

[Z. Wang and D. Liu, "Stability analysis for a class of systems: From model-based methods to data-driven methods," IEEE Trans. Ind. Electron., vol. 61, no. 11, pp. 6463–6471, Nov. 2014, doi: 10.1109/TIE.2014.2308146.]

Convergence of Neural Architectures

Theorem 1: Time-variant discrete-time weight-update IPLNAs system

$$\mathbf{w}(k+1) = \mathbf{A}(k)\cdot\mathbf{w}(k) + \mathbf{B}(k)\cdot\mathbf{u}(k)$$

is BIBS stable if there exist constants L_u, M_A and M_B for which

$$\sup_{k>k_0}\{||\mathbf{u}(k)||\} = L_u < \infty, \quad \sup_{k>k_0}\{||\mathbf{A}(k)||\} = M_A < 1, \; \sup_{k>k_0}\{||\mathbf{B}(k)||\} = M_B < \infty.$$

Recall the weight updates for IPLNAs using gradient descent learning rule result in

$$\mathbf{w}(k+1) = \left(\mathbf{I} - \eta(k)\cdot\mathbf{g}(\mathbf{x})\cdot\mathbf{g}(\mathbf{x})^T\right)\cdot\mathbf{w}(k) + \eta(k)\cdot y(k)\cdot\mathbf{g}(\mathbf{x})$$

Using notation $\mathbf{A}(k) = \left(\mathbf{I} - \eta(k)\cdot\mathbf{g}(\mathbf{x})\cdot\mathbf{g}(\mathbf{x})^T\right)$, $\mathbf{u}(k) = y(k)\cdot\mathbf{g}(\mathbf{x})$ we have

$$\mathbf{w}(k+1) = \underbrace{\left[\prod_{j=0}^{k-k_0}\mathbf{A}(k-j)\right]}_{=\,\mathbb{A}_i}\mathbf{w}(k_0) + \sum_{i=k_0}^{k-1}\underbrace{\left[\prod_{j=1}^{k-i}\mathbf{A}(k-j+1)\right]}_{=\,\mathbb{C}_{k-1,i}}\mathbf{B}(i)\mathbf{u}(i) + \mathbf{B}(k)\mathbf{u}(k).$$

Applying a norm and using the triangle inequality we obtain

$$\|\mathbf{w}(k+1)\| \leq \|\mathbb{A}_k\|\cdot\|\mathbf{w}(k_0)\| + \sum_{i=k_0}^{k}\|\mathbb{C}_{k,i}\|\cdot\|\mathbf{B}(i)\|\cdot\|\mathbf{u}(i)\|$$

[I. Bukovsky, G. Dohnal, P. M. Benes, K. Ichiji and N. Homma (2020): *Letter on Convergence of In-Parameter-Linear Nonlinear Neural Architectures With Gradient Learnings.* IEEE Transactions on Neural Networks and Learning Systems]

Convergence of Neural Architectures

Corollary: An IPLNA, for which there exists a constant $0 < q < 1$ such that

$$\rho\left(\mathbf{I} - \eta(k) \cdot \mathbf{g}(\mathbf{x},k) \cdot \mathbf{g}^T(\mathbf{x},k)\right) \leq q$$

for all $k > k_0$, is BIBS stable.

Recall that $\mathbf{A}(k)$ is a Hermitian matrix and therefore $||\mathbf{A}(k)|| \leq \rho(\mathbf{A}(k) \leq q$.

Theorem 2: Time-variant discrete-time weight-update IPLNAs systems with $||\mathbf{A}(i)|| \leq q < 1, \quad i > k_0$ are ISS.

We had $$\|\mathbf{w}(k+1)\| \leq \underbrace{\|\mathbb{A}_k\| \cdot \|\mathbf{w}(k_0)\|}_{\beta\left(||\mathbf{w}(k_0)||.k\right)} + \underbrace{\sum_{i=k_0}^{k} \|\mathbb{C}_{k,i}\| \cdot \|\mathbf{B}(i)\| \cdot \|\mathbf{u}(i)\|}_{\gamma\left(||\mathbf{u}||\right)}$$

$\mathcal{K}_{\mathcal{L}}$ $\qquad$ $\mathcal{K}_{\infty}$

(I.e., $\beta(x,k)$ is strictly increasing in x, $\lim_{x\to\infty} \beta(x,k) = \infty$, $\beta(0,k) = 0$ and decreasing to 0 in k, $k \to \infty$, function $\gamma(x)$ is strictly increasing in x, $\lim_{x\to\infty} \gamma(x) = \infty, \; \gamma(0) = 0$)

Stability of learning entropy based detection

In the case of classical stability (e.g., Ljapunov), we ask $\rho(\mathrm{A}(k)) < 1$.

- In this case: all eigenvalues of $\rho(\mathbf{A}(k))$ equal to 1, but one which equals to

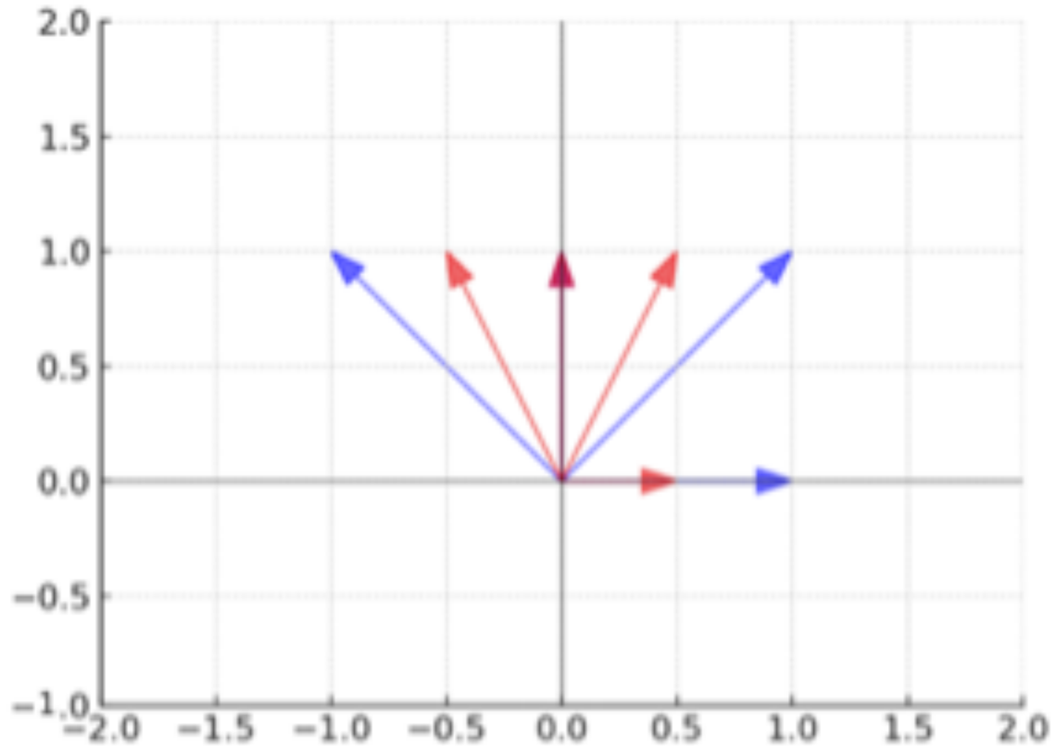
$$1 - \eta(k)\frac{||\mathbf{g}(\mathbf{x}(k))||^2}{||\mathbf{g}(\mathbf{x}(k))||^2 + \epsilon}$$

- therefore $\rho(\mathbf{A}(k)) = 1$

<= there exists a direction in which the weights do not change (and don't diverge)

=> the weight dynamics is not strictly contractive - it is **marginally stable**.

Geometrical illustration for **A**=**I**−$\mu g g^\top$ (μ=0,5; g=[1;0]$^\top$)
blue = original,
red = transformed



VI. Conclusions

Conclusions

- The condition $\rho(\mathbf{A}(k)) = 1$ is too strong, in real learning we observe $\rho(\mathbf{A}(k)) \approx 1$ or $\rho(\mathbf{A}(k)) = 1 + \varepsilon$ for some small $\varepsilon > 0$.

- This corresponds to: even if immediately $\rho(\mathbf{A}(k)) = 1$, in the long run the average mass energy is damped because in the direction of the error the eigenvalue decreases to $1-\eta(k)$.

Theorem: If the learning rate function $\eta(k)$ satisfies the following inequality

$$0 < \eta(k) \leq \frac{2}{\mathbf{g}(\mathbf{x}(k))\mathbf{g}^T(\mathbf{x}(k))},$$

then the IPLNA system is BIBS.

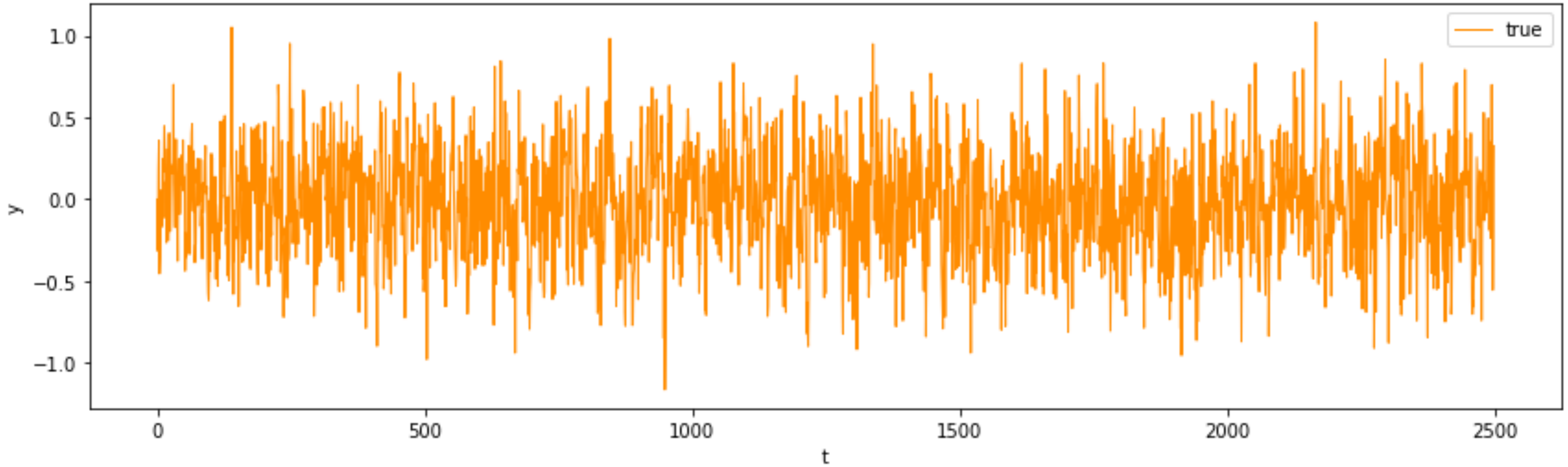
- In other words: the system is BIBS stable because the weights remain bounded and do not explode even if the strict condition $\rho < 1$ is not met.

VII. Simulation study

Simulation study

Process: $y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$

where u(t) is white noise, $\epsilon \sim N(0, 0.06)$

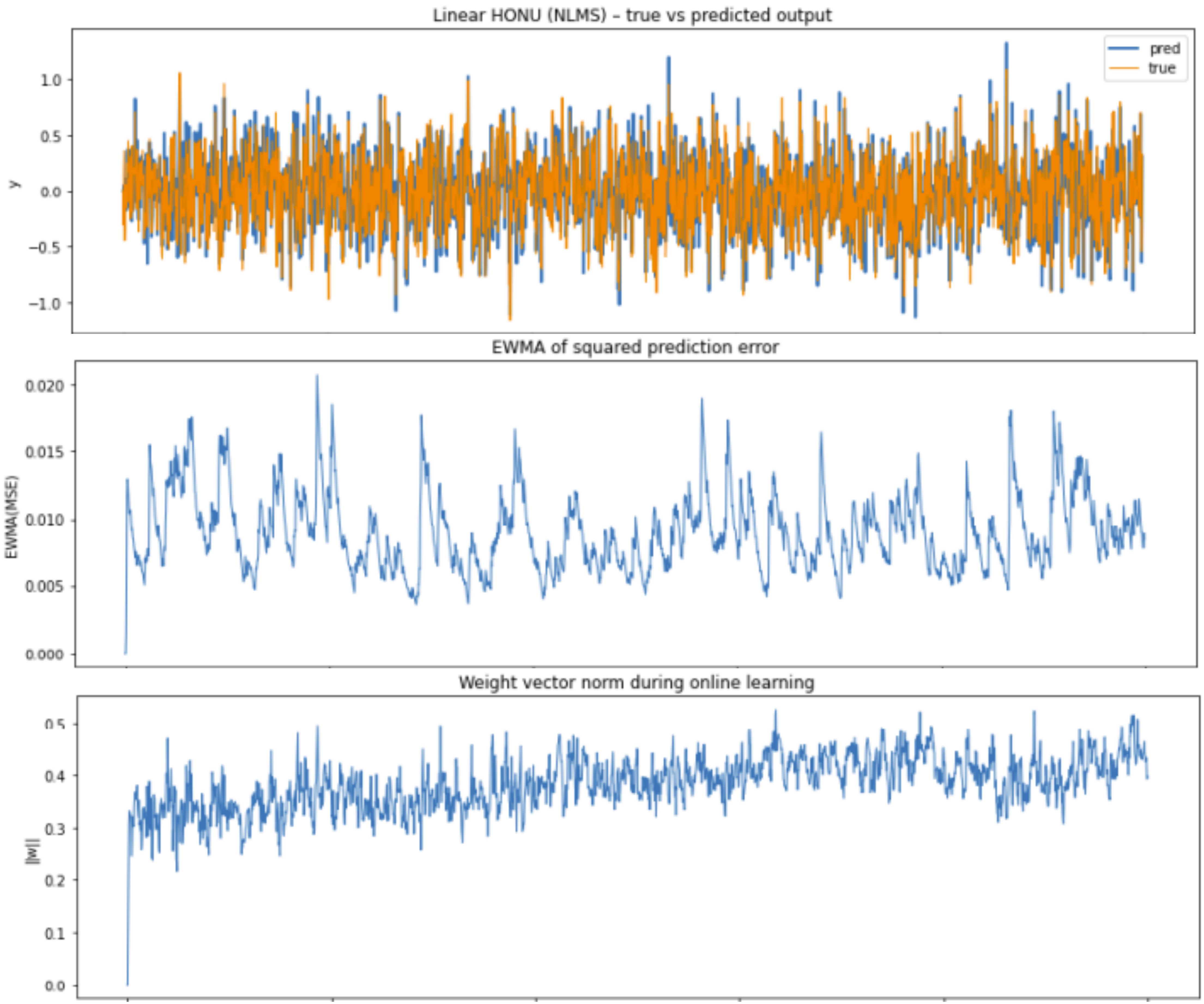


predictor: LNU (Linear HONU)

learning rate: $\eta(k) = \dfrac{\mu}{\mathbf{g}(\mathbf{x}(k))\mathbf{g}^T(\mathbf{x}(k))}$ with different μ

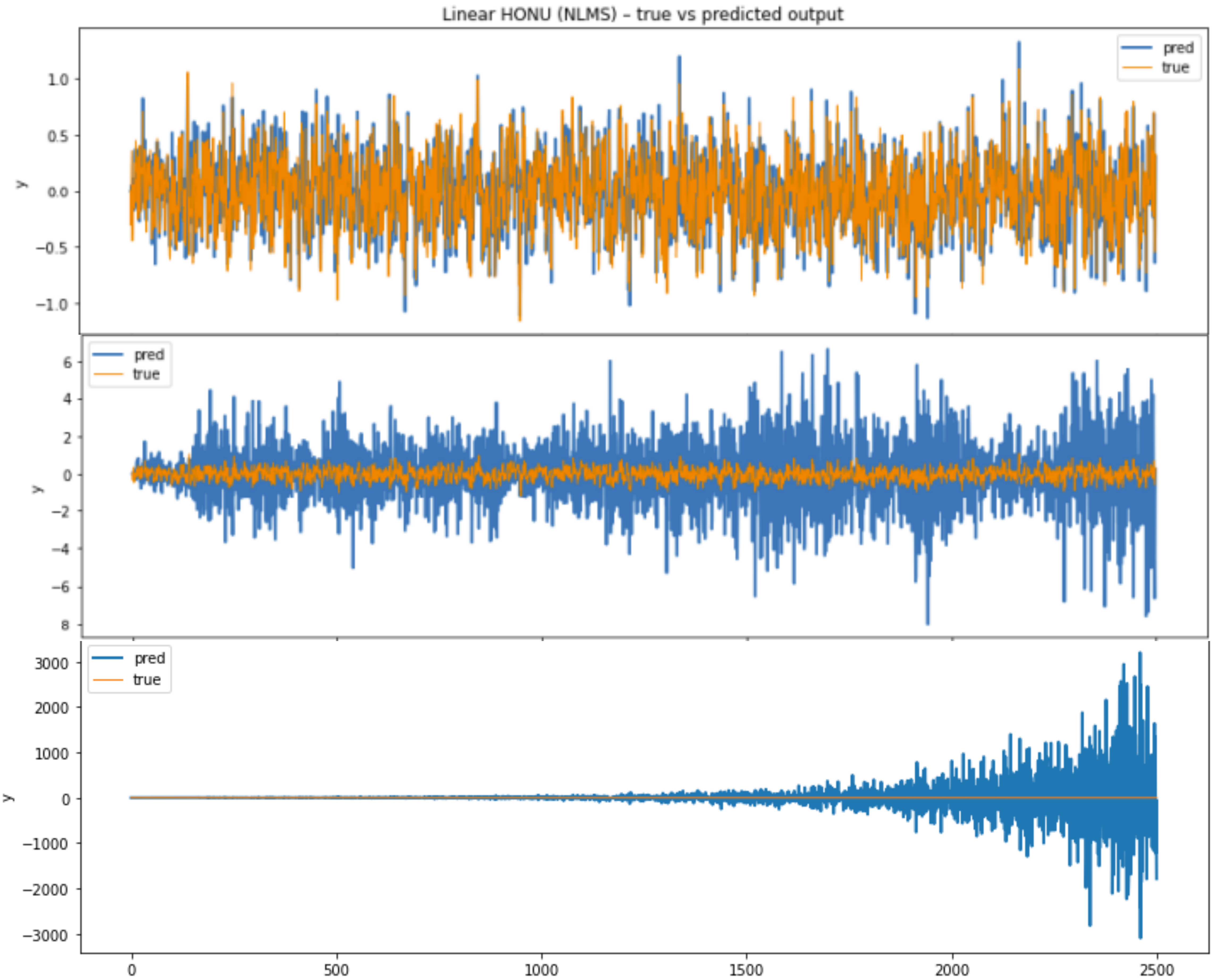
Linear HONU (NLMS) – true vs. predicted output for $\mu = 1.9$

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



Linear HONU (NLMS) – true vs. predicted output

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



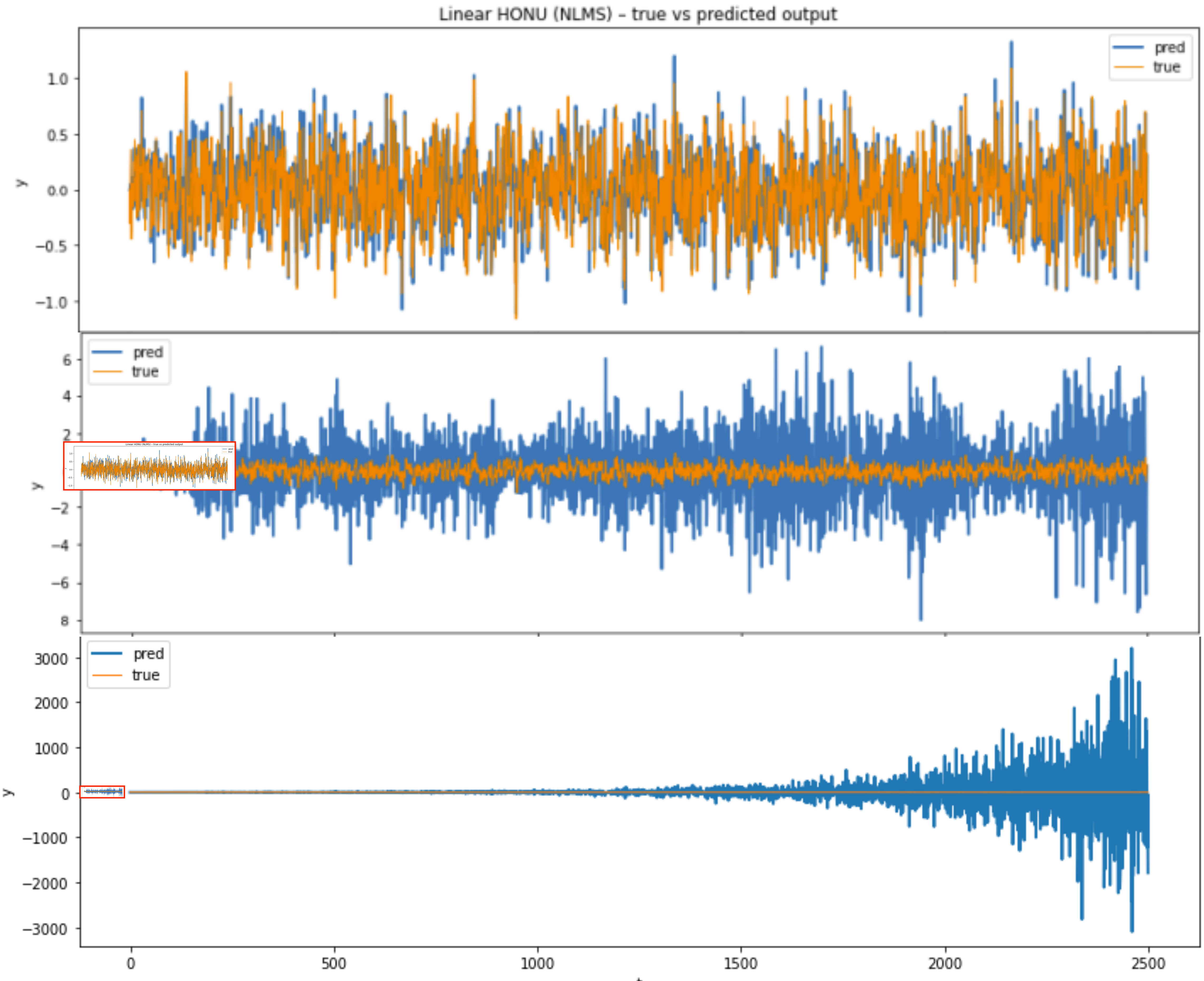
$\mu = 1.9$

$\mu = 2.0$

$\mu = 2.01$

Linear HONU (NLMS) – true vs. predicted output

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



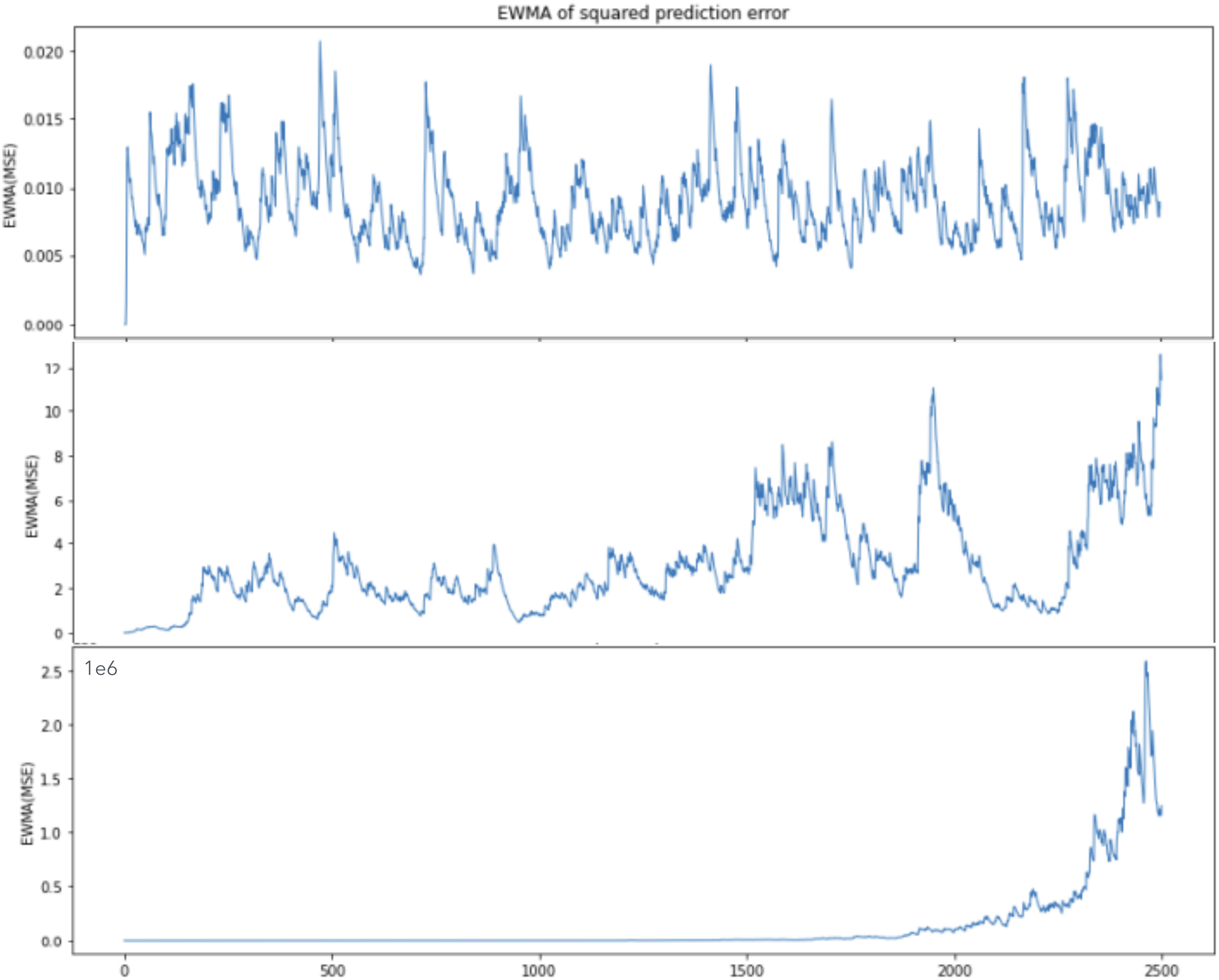
$\mu = 1.9$

$\mu = 2.0$

$\mu = 2.01$

EWMA of squared prediction error

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$,   where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



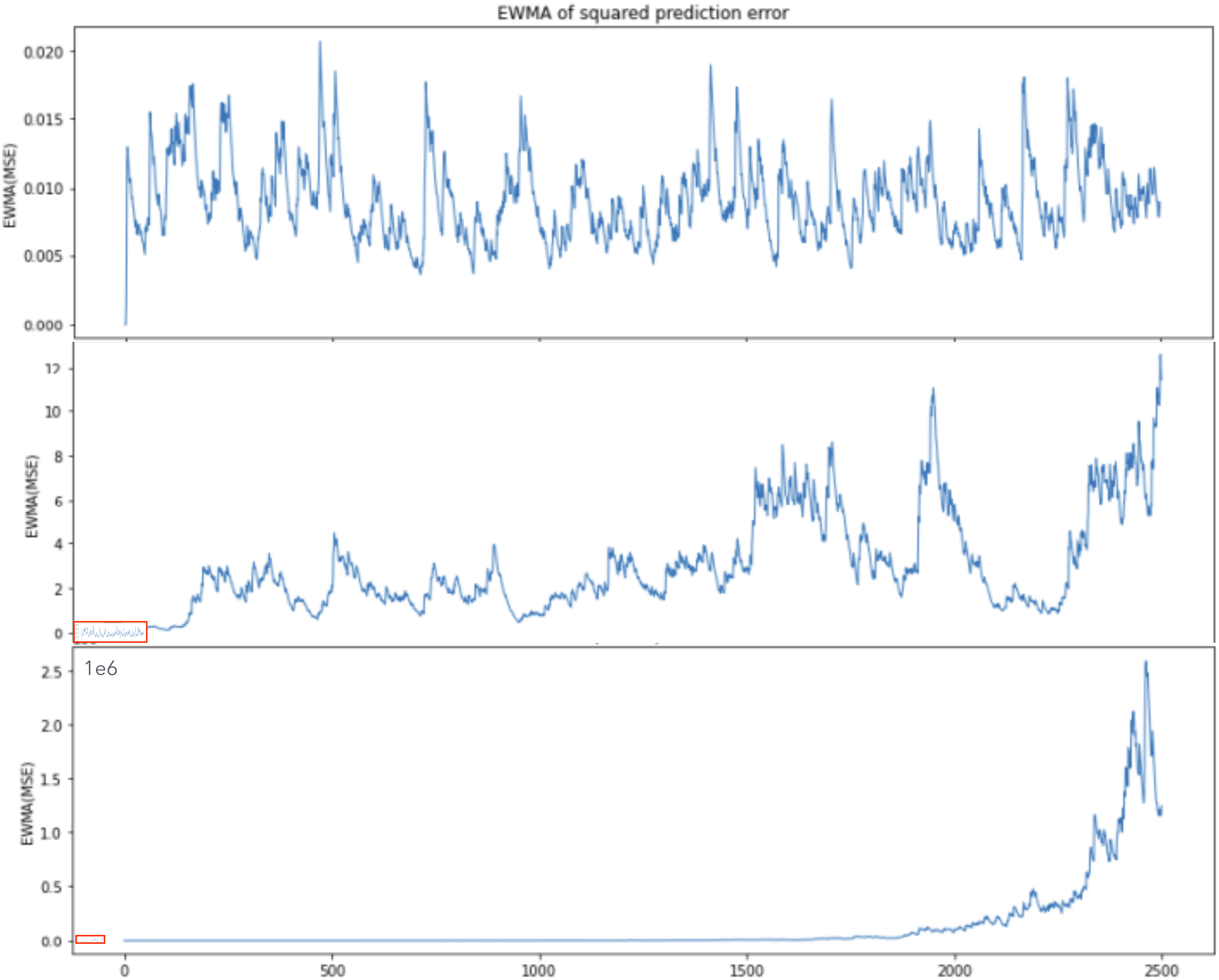
$\mu = 1.9$

$\mu = 2.0$

$\mu = 2.01$

EWMA of squared prediction error

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$,   where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



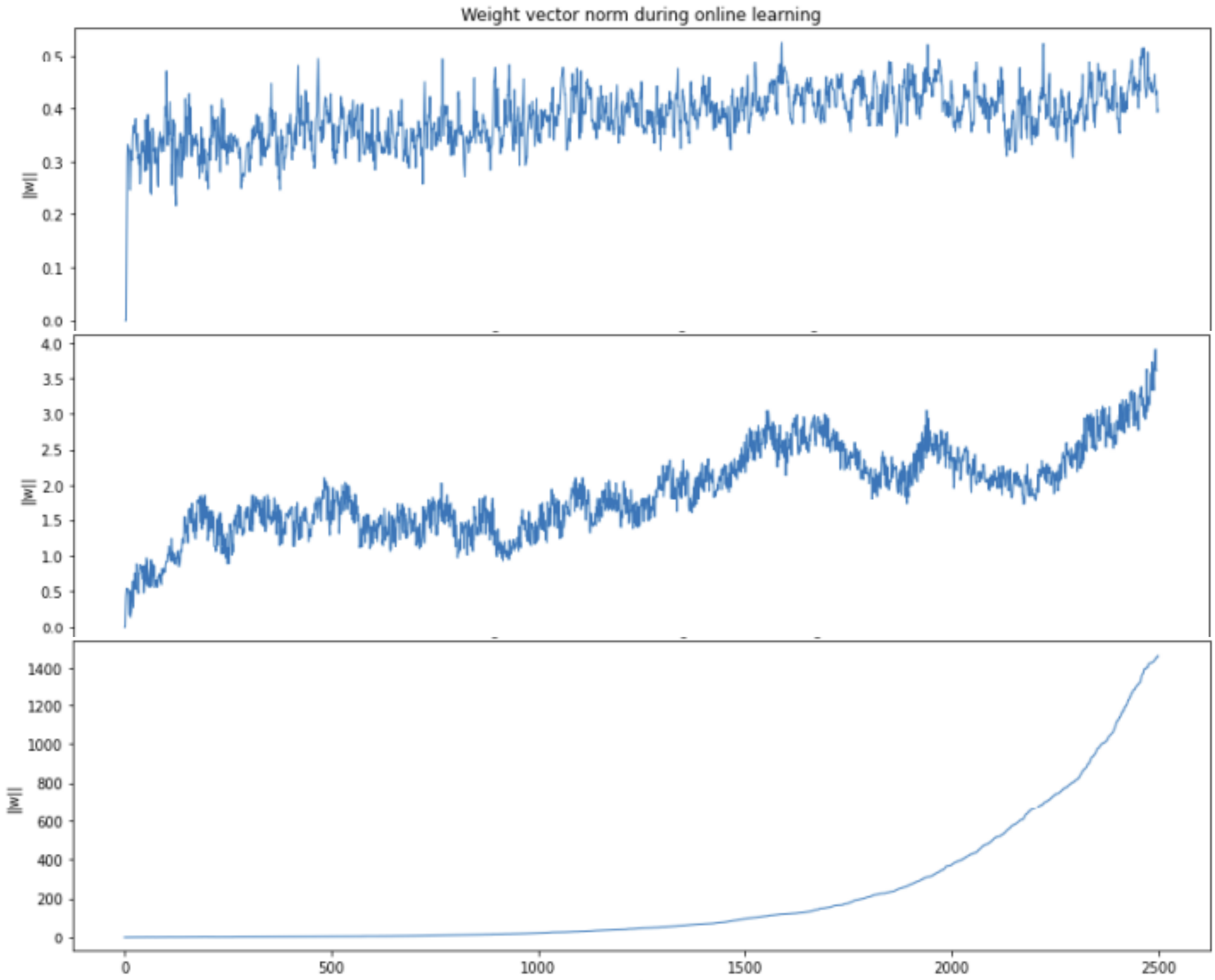
$\mu = 1.9$

$\mu = 2.0$

$\mu = 2.01$

Weight vector norm during online learning

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$



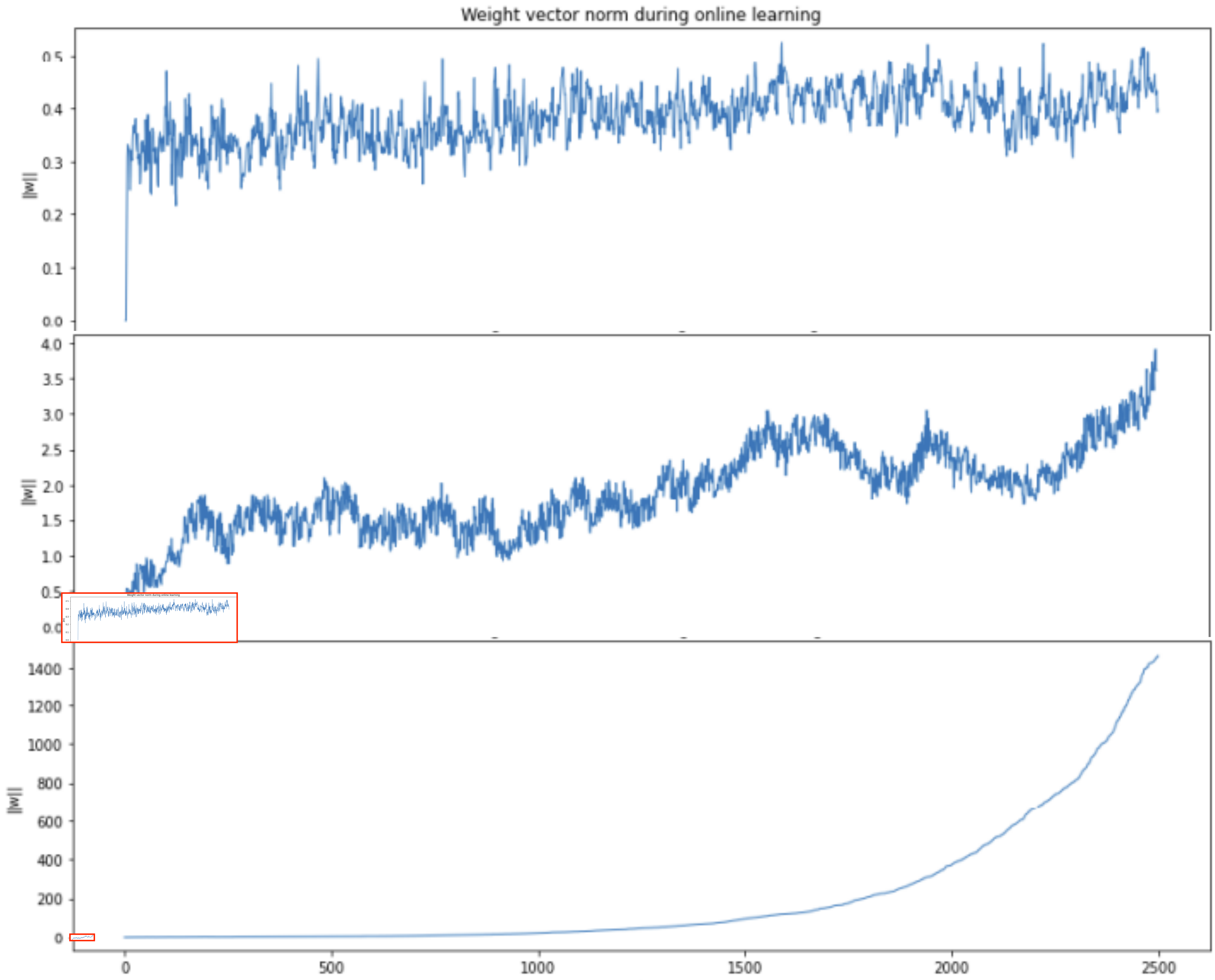
$\mu = 1.9$

$\mu = 2.0$

$\mu = 2.01$

Weight vector norm during online learning

$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$

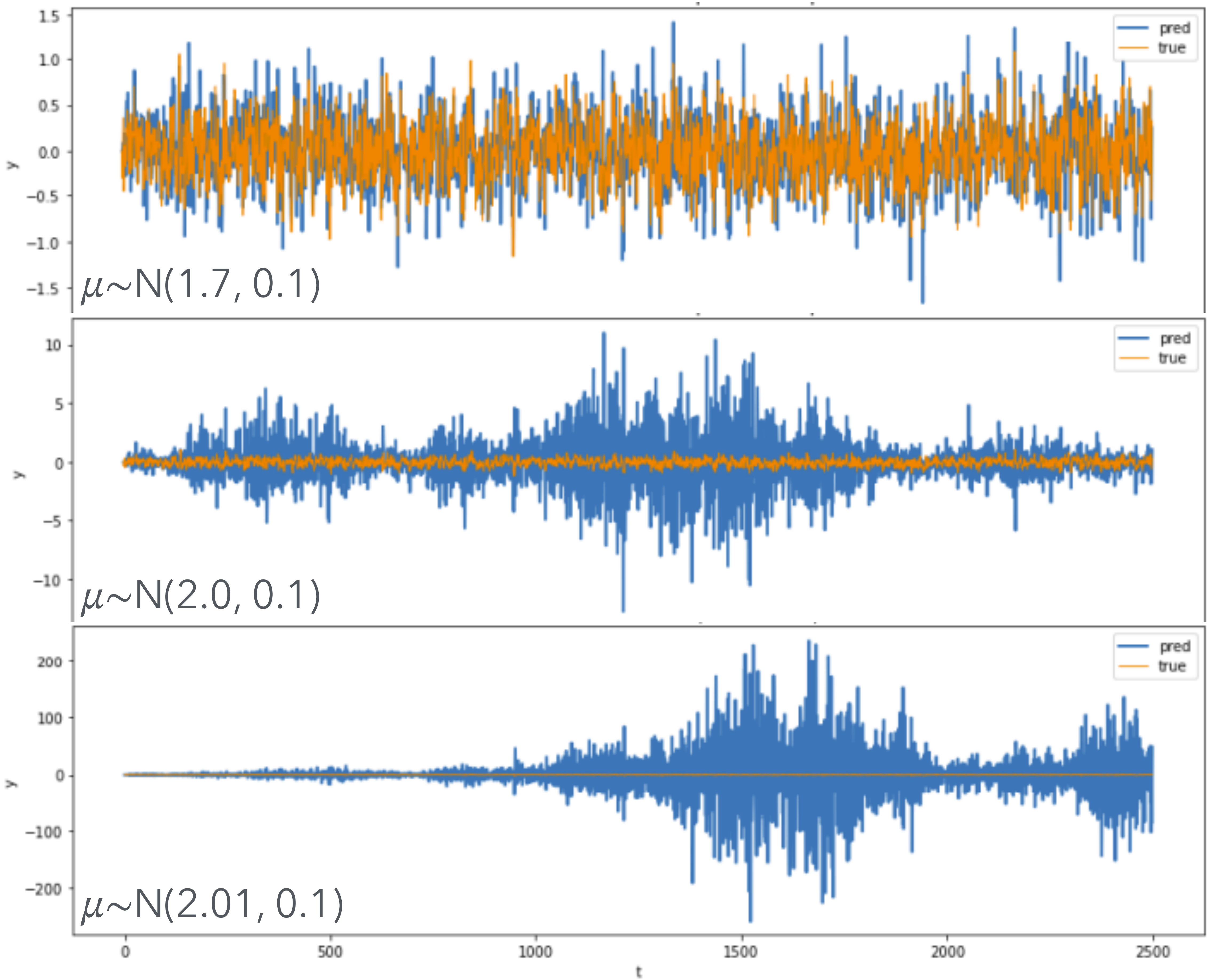


$\mu = 1.9$

$\mu = 2.0$

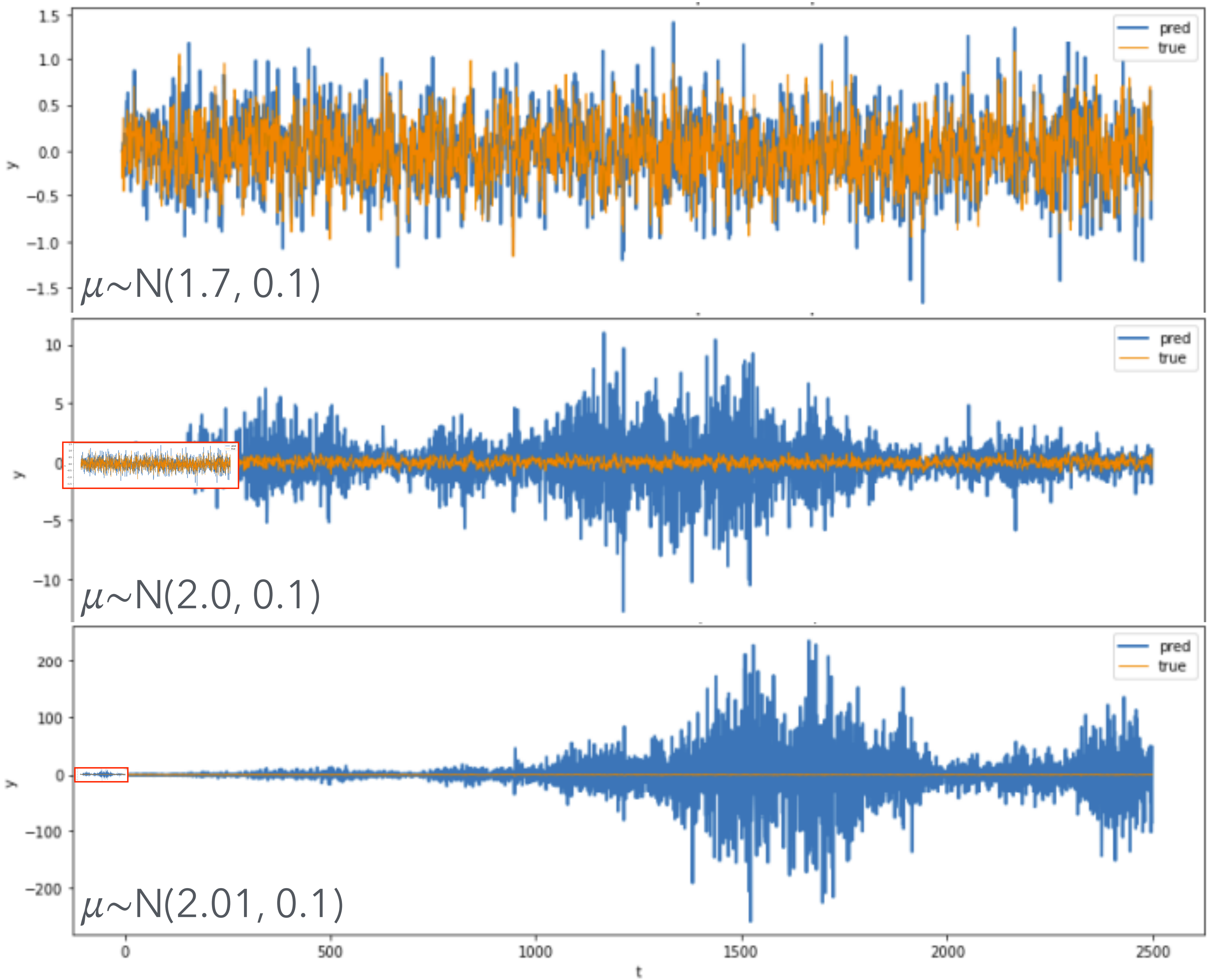
$\mu = 2.01$

Linear HONU (NLMS) – true vs. predicted output



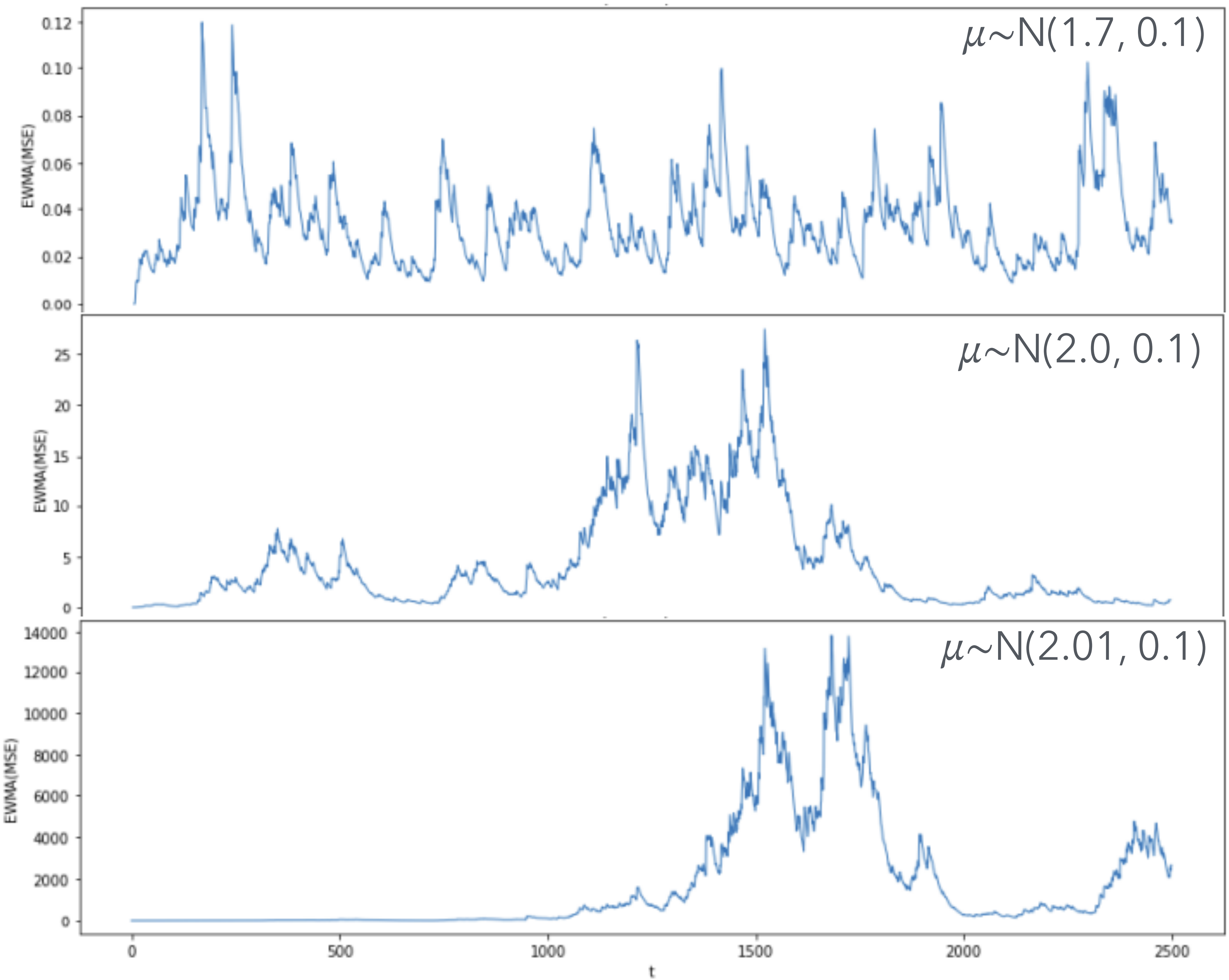
$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where u(t) is white noise, $\epsilon \sim N(0, 0.06)$

Linear HONU (NLMS) – true vs. predicted output



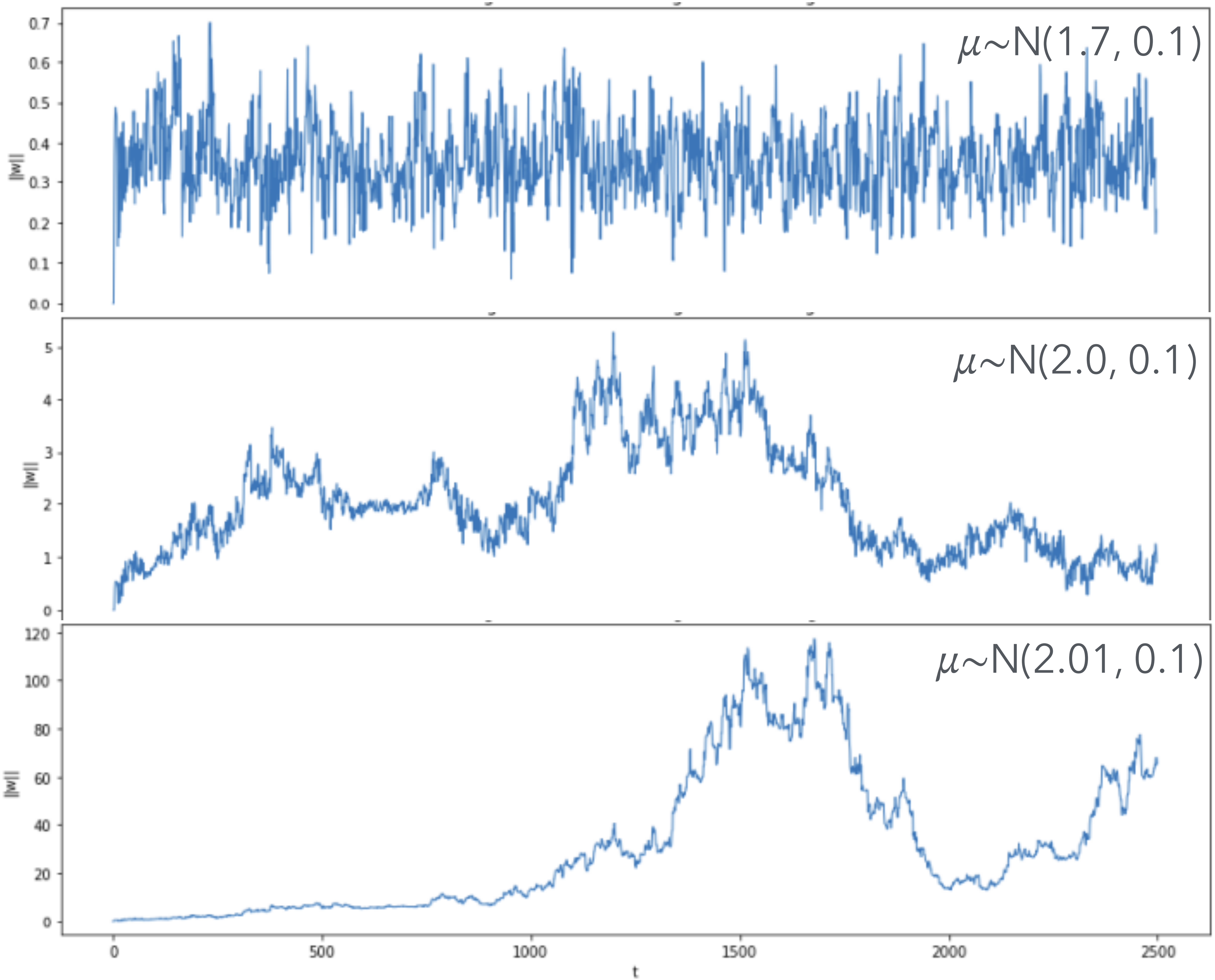
y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+ϵ, where u(t) is white noise, ϵ~N(0, 0.06)

EWMA of squared prediction error



$y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+\epsilon$, where $u(t)$ is white noise, $\epsilon \sim N(0, 0.06)$

Weight vector norm during online learning



y(t) = 0.6*y(t-1)+0.3*u(t)-0.12*u(t-1)+ϵ,   where u(t) is white noise, ϵ~N(0, 0.06)

Thank you for your attention