



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA AERONÁUTICA Y DEL ESPACIO

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA

TRABAJO FIN DE MÁSTER

Reservoir Neural Networks and time series forecasting: application to failure prediction and predictive maintenance in aerospace.

AUTOR: Juan Manuel RODRÍGUEZ RIESGO

INTENSIFICACIÓN: Vehículos Espaciales

TUTOR DEL TRABAJO: Juan Luis CABRERA FERNÁNDEZ

Julio de 2024

Abstract

Predictive maintenance is a budding field in the aerospace industry with certain caveats that have made implementation sluggish, specially considering the need for data availability and sensor technologies which lack wide commercial implementation industry-wide. This thesis proposes a ESN-GWO system, a Reservoir Neural Network whose configuration is optimized by means of a Grey Wolf Optimizer that explores the hyperparameters in search of the configuration best suited to forecast a given input signal. Once an accurate prediction is generated, the analysis to determine whether or not maintenance is required could begin. The necessary requirements for the system to generate this predictions are defined, with specific suggestions as to how a prediction can be improved through reservoir computing. Three different datasets are studied: the Mackey-Glass equation, the CMAPSS dataset and the power consumption from a resort facility. The preliminary results from these time series assist to determine certain common rules that improve the quality of the predictions and focus the optimization towards hyperparameter solutions that may allow for a faster approach to predictive maintenance.

Acknowledgements

"Yes, I began my journey alone, and I ended it alone.
But that does not mean that I walked alone."
Brandon Sanderson, Oathbringer

Este Trabajo Fin de Máster ha sido un año lleno de descubrimientos y aprendizaje, encontrando un campo en el que no me imaginaba trabajando nunca pero que ahora se ha vuelto una pasión que quiero seguir explorando. Quisiera agradecer a mi tutor Juan Luis Cabrera la oportunidad brindada, enseñándome y teniendo paciencia conmigo.

También tengo que agradecer a mis amigos Luis Enrique Abarca, Manuel Falagan y Diego Vicente por siempre darme una opinión sincera y una crítica que a veces no era muy constructiva pero era de agradecer. Siempre están ahí cuando se les necesita y eso es lo más importante.

A mis hermanos, Alba y Jorge, y a mis padres, María José y Manuel, por tener que aguantarme cuando no podía más y apoyarme siempre con las decisiones que tomo. Quisiera dar mención especial a mi padre, ya que algunos datos incluidos en este trabajo son suyos.

Y, por último, gracias a ti Teo. Has sido una parte fundamental de este trabajo y me has ayudado y animado hasta el final para poder terminarlo. El paso más importante es siempre el siguiente.

Contents

1 Introduction **11**
1.1 Predictive maintenance . . . 11
1.2 Undesired fluctuations in complex systems . . . 14
1.3 Prediction of unstable systems . . . 16

2 Reservoir Computing and Optimization **17**
2.1 General information on Neural Networks . . . 17
2.1.1 Activation functions . . . 18
2.1.2 Network topology . . . 19
2.1.3 Training algorithm . . . 20
2.2 Reservoir Computing . . . 21
2.2.1 Echo State Network . . . 22
2.3 Meta-heuristic Optimization Algorithms . . . 24
2.3.1 Grey Wolf Optimizer . . . 25
2.4 ESN-GWO System . . . 30
2.4.1 System workflow . . . 31

3 Experimental data **34**
3.1 Mackey-Glass time series . . . 34
3.2 CMAPSS dataset . . . 35
3.3 Resort facility power consumption (PC dataset) . . . 44

4 Experimental results **47**
4.1 Mackey-Glass dataset . . . 47
4.2 CMAPSS dataset . . . 56
4.2.1 Fourth set of hyperparameters for the CMAPSS dataset . . . 56
4.2.2 Validation using the fourth set of hyperparameters with Unit 93 . . . 65
4.3 Punta Cana dataset . . . 69
4.3.1 Third set of hyperparameters for the PC dataset . . . 71

5 Conclusions **78**

6 Future Steps **81**

A Additional experimental results **87**
A.1 CMAPSS dataset . . . 87
A.1.1 First set of hyperparameters for the CMAPSS dataset . . . 87
A.1.2 Second set of hyperparameters for the CMAPSS dataset . . . 94
A.1.3 Third set of hyperparameters for the CMAPSS dataset . . . 101
A.1.4 Fifth set of hyperparameters for the CMAPSS dataset . . . 108
A.2 Punta Cana dataset . . . 115

A.2.1 First set of hyperparameters for the PC dataset 115
A.2.2 Second set of hyperparameters for the PC dataset 122
A.2.3 Fourth set of hyperparameters for the PC dataset 128

B Published article **135**

List of Figures

1.1 Overview of the maintenance types [6] . . . 12
1.2 Trend on number of papers anually on predicitive maintenance [6] . . . 13
1.3 Full Authority Digital Engine Controls (FADEC) from Safran [11] . . . 13
1.4 With surge the flow stalling at the blades means that the accumulation of air downstream of the compressor can no longer be sustained. The accumulated air discharges in the opposite direction and flows backwards through the compressor. When the pressure level behind the compressor has dropped sufficiently, the flow reattaches on the blades and the pressurization starts over again.[16] . . . 15
1.5 Rotor downwash flow field, (a) out-of-ground effect, (b) in-ground effect. [18] . . . 15

2.1 Signal pathway inside of a neuron in a human brain [20] . . . 18
2.2 Signal pathway inside of a node in a neural network [20] . . . 18
2.3 Most commonly used activation functions in Neural Networks [20] . . . 19
2.4 Example of a simple multilayered network [20] . . . 20
2.5 The basic Echo State Network architecture. Dashed arrows indicate connections that are possible but not required. [23] . . . 22
2.6 Hierarchy of grey wolfs in a pack (dominance decreases from top down). [38] . . . 25
2.7 Position updating according to α, β and δ in a 2D search space. The final position is a random location withing a circle defined by where the lead wolves are in the search space. [38] . . . 27
2.8 Attacking (a) or searching for prey (b) depending on the value of $|A|$ [38] . . . 28
2.9 ESN-GWO System flowchart . . . 33

3.1 Mackey-Glass series with a time delay n=17 . . . 35
3.2 Simplified diagram of the turbofan engine simulated in CMAPSS [41] . . . 35
3.3 CMAPSS dataset Test 1, Unit 49: Altitude, Mach, TRA and T2 . . . 38
3.4 CMAPSS dataset Test 1, Unit 49: T23, T30, T50, P2 . . . 39
3.5 CMAPSS dataset Test 1, Unit 49: P15, P30, Nf, Nc . . . 40
3.6 CMAPSS dataset Test 1, Unit 49: EPR, Ps30, ϕ, NRf . . . 41
3.7 CMAPSS dataset Test 1, Unit 49: NRc, BPR, farB, htBleed . . . 42
3.8 CMAPSS dataset Test 1, Unit 49: Nf_dmd, PCNfR_dmd, W31, W32 . . . 43
3.9 Direct solar radiation in Punta Cana (Dominican Republic) during 2021 [44] . . . 44
3.10 From top to bottom: differential values in 1, 10 and a 100 time steps for the PC dataset . . . 45
3.11 Resort facility power consumption from (top to bottom): January 2021 to July 2021, August 2021 to February 2022, March 2022 to September 2022 and October 2022 to April 2023. In cyan the unfiltered signal and in brown the filtered signal. . . . 46

4.1 Mackey-Glass time series along with the predicted signal (above), the $|\Delta|$ variation in time (middle) and the predicted signal during the first 500 time steps (below) for the reservoir with the first set of hyperparameters 48
4.2 Mackey-Glass time series along with the predicted signal (above), the $|\Delta|$ variation in time (middle) and the predicted signal during the first 500 time steps (below) for the reservoir with the second set of hyperparameters 50
4.3 Reservoir internal states at various moments during training for the Mackey-Glass time series for the first set of hyperparameters 52
4.4 Histogram of reservoir internal states at various moments during training for the Mackey-Glass time series for the first set of hyperparameters 53
4.5 Reservoir internal states at various moments during training for the Mackey-Glass time series for the second set of hyperparameters 54
4.6 Histogram of reservoir internal states at various moments during training for the Mackey-Glass time series for the second set of hyperparameters 55
4.7 nRMSE value of each of the signals for the prediction generated by the reservoir with the fourth set of hyperparameters. 58
4.8 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 59
4.9 Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 60
4.10 Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 60
4.11 Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 61
4.12 Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 61
4.13 Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters 62
4.14 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters 63
4.15 Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters 64
4.16 RMSE value of each of the signals for the prediction generated by the reservoir with the fourth set of hyperparameters with the data from Unit 93. 65
4.17 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters with the data from Unit 93 66
4.18 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters with the data from Unit 93 . . . 67
4.19 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters with the data from Unit 93 . . . 68
4.20 PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the third set of hyperparameters 71

4.21 PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the third set of hyperparameters 72
4.22 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the third set of hyperparameters 74
4.23 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the third set of hyperparameters 75
4.24 Reservoir internal states at various moments during training for the PC time series for the third set of hyperparameters 76
4.25 Distribution of the reservoir internal states at various moments during training for the PC time series for the third set of hyperparameters 77

A.1 RMSE value of each of the signals for the prediction generated by the reservoir with the first set of hyperparameters. Average RMSE: 1.2784 88
A.2 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 89
A.3 Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 89
A.4 Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 90
A.5 Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 90
A.6 Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 91
A.7 Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters 91
A.8 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the first set of hyperparameters 92
A.9 Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the first set of hyperparameters 93
A.10 nRMSE value of each of the signals for the prediction generated by the reservoir with the second set of hyperparameters. 94
A.11 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters 95
A.12 Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters 96
A.13 Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters 96
A.14 Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters 97

A.15 Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters . . . 97
A.16 Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters . . . 98
A.17 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the second set of hyperparameters . . . 99
A.18 Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the second set of hyperparameters . . . 100
A.19 nRMSE value of each of the signals for the prediction generated by the reservoir with the third set of hyperparameters. . . . 101
A.20 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 102
A.21 Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 103
A.22 Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 103
A.23 Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 104
A.24 Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 104
A.25 Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters . . . 105
A.26 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the third set of hyperparameters . . . 106
A.27 Distribution of the reservoir internal states various moments of training cycle C for the CMAPSS time series for the third set of hyperparameters . . . 107
A.28 RMSE value of each of the signals for the prediction generated by the reservoir with the fifth set of hyperparameters. . . . 108
A.29 Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 109
A.30 Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 110
A.31 Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 110
A.32 Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 111
A.33 Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 111

A.34 Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters . . . 112
A.35 Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fifth set of hyperparameters . . . 113
A.36 Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fifth set of hyperparameters . . . 114
A.37 PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the first set of hyperparameters . 116
A.38 PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the first set of hyperparameters . . . 117
A.39 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the first set of hyperparameters . . . 118
A.40 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the first set of hyperparameters . . . 119
A.41 Reservoir internal states at various moments during training for the PC time series for the first set of hyperparameters . . . 120
A.42 Distribution of the reservoir internal states at various moments during training for the PC time series for the first set of hyperparameters . . . 121
A.43 PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the second set of hyperparameters 122
A.44 PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the second set of hyperparameters . . . 123
A.45 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the second set of hyperparameters . . . 124
A.46 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the second set of hyperparameters . . . 125
A.47 Reservoir internal states at various moments during training for the PC time series for the second set of hyperparameters . . . 126
A.48 Distribution of the reservoir internal states at various moments during training for the PC time series for the second set of hyperparameters . . . 127
A.49 PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the fourth set of hyperparameters 128
A.50 PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the fourth set of hyperparameters . . . 129
A.51 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the fourth set of hyperparameters . . . 131
A.52 PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the fourth set of hyperparameters . . . 132
A.53 Reservoir internal states at various moments during training for the PC time series for the fourth set of hyperparameters . . . 133
A.54 Reservoir internal states histogram at various moments during training for the PC time series for the fourth set of hyperparameters . . . 134

List of Algorithms

1 Grey Wolf Optimizer with bounds enforcement 29

1 Introduction

The ability to glimpse the future is trivial, to do so with any degree of certainty is nothing short of miraculous. A profound knowledge of a complex systems may not guarantee nor facilitate an understanding of its real-world behaviour, as an insurmountable amount of phenomena is interacting constantly. Methods exists that break down and simplify these inner workings, but the fidelity is lost when trying to translate multitudes into individuals, loosing the context in which each unit operates.

Unfortunately, this conundrum has never stopped engineers from trying to understand how their own creations behave in the wild. The biggest foe any designer can face is not whether it will start, but if it will pass the test of time and how long before something critical fails. Maintenance is the solution to a problem as old as time. Once again, how to maintain it is not a problem. When to do it so as to be efficient, that is when stuff gets dicey.

1.1 Predictive maintenance

Experience is the foundation for any modern maintenance process, all prognostics on complicated systems are either model-based or data-driven. The former depends on the use of physics-based models of the degradation processes and are typically applied to safety-critical and slow-degrading equipment whose degradation mechanisms have been extensively studied. The latter, only used when no accurate physics model is available, requires enormous amounts of data for training, only being applied to non-safety critical systems with short mean time to failure [1].

Model-based prognostics have been the preferred choice during the last century, resting the required actions to be taken as much from the physical knowledge behind the model as from the experience of the engineers tasked to maintain it. At the dawn of this millennia though, new technologies rose which provided a far more powerful process to predict failure given enough data on the workings of a mechanical system. The advent of Machine Learning brought a whole new set of tools for engineers to predict, simulate and diagnose events that would be catastrophic if occurring on a real-life scenario, allowing an increase in reliability across all modern industrial fields.

Data-driven models are therefore gaining traction on various engineering fields, specially Aerospace, as even reduced improvements through computing-intensive simulations can yield millions through efficiency savings on the long run. Not only Machine Learning is being used extensively, but digital models are being widely employed so as to combine both technologies and speed up greatly project development. Industrial applicability of data-driven frameworks [2] have been developed due to several causes among which are prominent: the unprecedented increase in data availability; the advancements in high performance computation; improved data storage and sensor technologies; greater academic focus on algorithms and applied mathematics; and a flood of investment from the industry.

Aerospace engineering is, for instance, a great study case on how the advancement on computation has brought forth valuable tools for quick development, testing and optimization of aerodynamics

and propulsion. Aerodynamic Shape Optimization (ASO) [3], combines Computational Fluid Dynamics (CFD) along with Machine Learning to speed up shape optimization in wings, fuselage and nacelles. Instead of performing high-cost wind tunnel trials, CFD simulations are run to feed data to a neural network so it can be trained, creating surrogate models also known as "metamodels". Another choice is to train neural networks with physical laws, also known as Physics-Informed Neural Networks (PINN) [4], and then have the neural network optimize the design based on the desired parameters.

Preliminary engine design is also a field susceptible to the benefits posed by Machine Learning techniques. There are extensive databases on engine performance and design which can be exploited to speed up the design and sizing of axial-compressors, [5], which in turn reduces the time spent on the early stages of turbofan development.

Predictive maintenance, not exclusive to the aeronautics industry, is also gaining traction through the combination of increased capabilities in sensor networks as well as computational power available. The ability to pour through high dimensional and multivariate data and classify its internal relationships is key to determine the potential failures bound to happen.

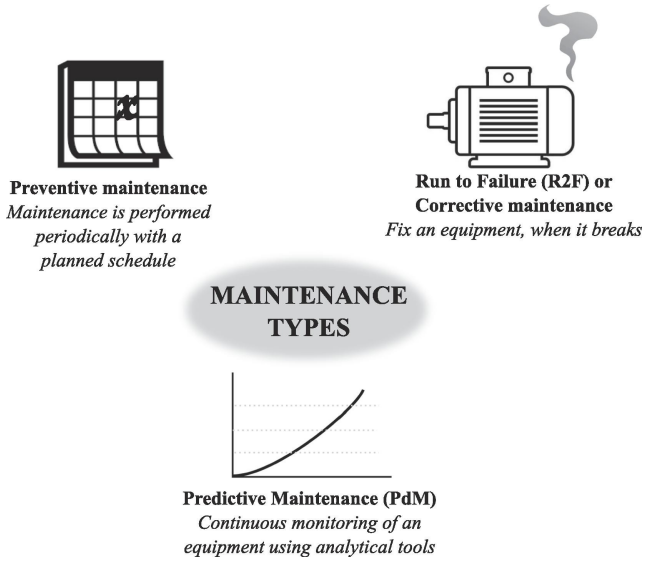


Figure 1.1: Overview of the maintenance types [6]

Three main categories have been presented to classify the potential maintenance frameworks available to the industry, [7]:

- **Run-to-failure:** only acting after the failure appears, which is the simplest and least efficient as servicing downtime cannot be scheduled or accounted for in advance.
- **Preventive maintenance:** acting repeatedly to avoid failures, setting a planned schedule based previous experience. This increases operation costs as resources are sometimes expended on needless maintenance.
- **Predictive maintenance:** acting right before the element fails, therefore requiring to estimate the remaining life at any given moment based on the historical information available through the use of "health" metrics.

The potential for *Predictive Maintenance*, once again, depends greatly on the correct implementation of appropriate Machine Learning techniques, to calculate within practical certainty those "health" metrics required to determine when to act.

The uptick in interest on this field has grown since 2010, [6], as academic interest on this area has increased, as shown in Figure 1.2, due to the industrial necessity for an ever increasing reduction

in service downtime, no matter whether we are referring to an airplane or a large-scale facility.

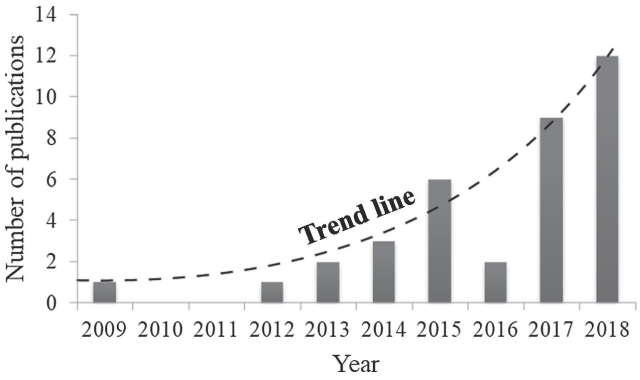


Figure 1.2: Trend on number of papers anually on predicitive maintenance [6]

The actual reduced amount of investigation on Predictive Maintenance has been due to the complexity of implementing effective strategies in production environments, [8]. The root cause is two-fold: professionals need to be formed both in data science and maintenance strategies and it is vital to have access to a solid and complete historical record to develop the models.

Some of the latest developments on the field come from the application of Reservoir Computing to emulate the future behaviour of certain components and thus predict the remaining life and how the failure will occur. Certain elements are of greater interest when studying critical components in aerospace, such as engines and avionics, as they contain Electronic Control Units capable of storing operational data. For instance, modern turbofan engines are governed by Full Authority Digital Engine Controls (FADEC), which since the 1990s are capable of running diagnostics and life tracking algorithms, storing in-flight data for later analysis if necessary, [9]. Three decades of historical data allows for the development of precise models, being favoured against more novel approaches such as Structural Health Monitoring, which are now being developed commercially. Structural Health Monitoring (SHM) is defined as "the process of acquiring and analysing data from on-board sensors to evaluate the health of a structure", [10], which seeks embedding permanent sensors on an airplanes fuselage and facilitate early fault detection or severe damage. This technology unfortunately has only gained relevance this last decade, which has prevented its commercial implementation sooner, and thus making structural data lacking for most Predictive Maintenance research.



Figure 1.3: Full Authority Digital Engine Controls (FADEC) from Safran [11]

The advantages of using Reservoir Computing for Predictive Maintenance will be further explored later on, but certain examples are noteworthy by the influence on this thesis. For instance, employing an Echo State Network to predict on the medium and long-term fluctuations on avionic

modules, [12], and thus allowing for preventive maintenance in case it strides off from its nominal value. Another implementation is the use of an ESN to predict the Remaining Useful Life from the operational data of several turbofan engines, analysing operational settings and sensor information to determine how many cycles are left before catastrophic failure, [13].

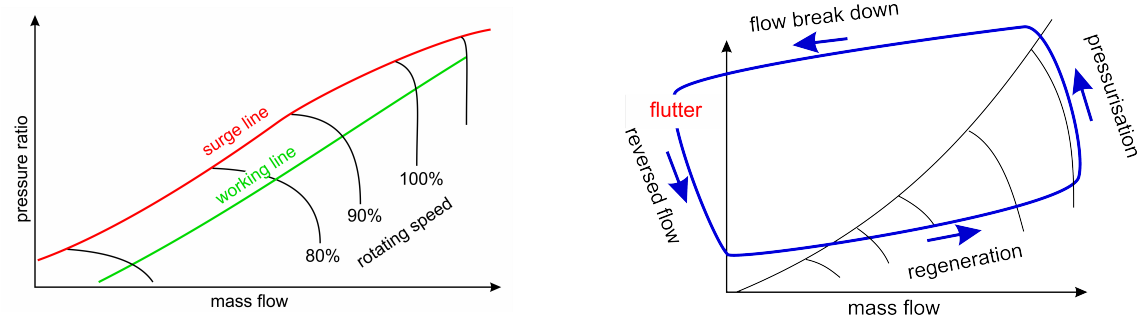
1.2 Undesired fluctuations in complex systems

The questions therefore are, what can be learned from a dynamical system? What are the limits to our understanding of its behaviour? And how long can it be predicted? The answers require a deep knowledge of its dynamic properties and the multitude of interactions amongst its various components. The power consumption of any given device, for instance, is a multivariate time series depending on the demanded output, inefficiencies and the inputs given as instructions. Even a rather simple system can suffer from fluctuations, rapid changes from the global mean that may affect its behaviour. Most of the time those changes are key to learn how the components are interacting and how it may fail on the long run.

Unfortunately these fluctuations are, for the most part, completely undesired from a design standpoint. No amount of modelling will completely cover every and all fluctuations caused by the interaction of the system with its surroundings. Take for instance the behaviour of a turbofan engine, designed for a certain flight envelope and conditions, when it starts suffering from the dynamic instability known as surge. Surge is a phenomenon that occurs when the compressor is working under high pressure ratios and low mass flow rates at a constant rotating speed. When this incident happens, the angle of attack of the flow around the blades increases causing its stall and the compressor starts working at an unstable position. Given these circumstances, mass flow reverses and starts discharging through the compressor. It seriously affects the performance and safety of an engine, especially when high temperature gases from the combustion chamber go through the compressor which is not prepared to withstand them. When the flow has been discharged, the compressor starts to increase the air's pressure level until the flow goes backwards and discharges once again [14].

Surge results in high amplitude and low frequency blade vibrations that may cause engine failure. It has a frequency ranging between 0.5 and 10 Hertz and an amplitude of 40 to 85 percent of the compression ratio. Certain indicators can be used to predict the starting point of this instability, even though all of them are short-term, indicating its imminent appearance. Most modern engines have multitude of safe-guards to prevent known surge regions in the compressor maps (e.g., Figure 1.4a), but may still occur in flight. This phenomenon is repeated periodically until the engine operating conditions are changed. However, since it may cause big structural damages to the compressor, it is necessary to initiate a controlled shutdown of the engine and a restart, which is against all flight safety tenets and needs to be avoided. Modern methods to predict surge are based on numerical approximations of each of the components (fan, compressor, combustion chamber and turbine) and then stacked together, which lacks the ability to fully capture the engines dynamic process. Therefore, the capacity to simulate and predict the behaviour of a turbofan engine fully and observe under which conditions it occurs is necessary for increased safety. Surge seriously affects the performance and safety of an engine, as the mass flow and pressure at the outlet of the compressor fluctuate greatly at low frequency, with the speed and thrust decreasing slowly and the exhaust temperature increasing rapidly, [15].

Another event in aerospace of a similar nature is the instability of drones low-altitude flight dynamics. Multi-rotor drones are difficult to fly precisely in near-ground trajectories due to the complex aerodynamic interactions between the wakes of the propellers and other aerodynamic surfaces with the ground, [17]. Compensating for ground-effect has been under performing, most methods are based on mathematical modelling of aerodynamic forces through steady-flow conditions or adap-



(a) Compressor map

(b) Surge cycle in the compressor map

Figure 1.4: With surge the flow stalling at the blades means that the accumulation of air downstream of the compressor can no longer be sustained. The accumulated air discharges in the opposite direction and flows backwards through the compressor. When the pressure level behind the compressor has dropped sufficiently, the flow reattaches on the blades and the pressurization starts over again.[16]

tive control methods. The first solution is simplistic, as most life scenarios show unsteady flow conditions, and the second suffer from slow responses and delayed feedback.

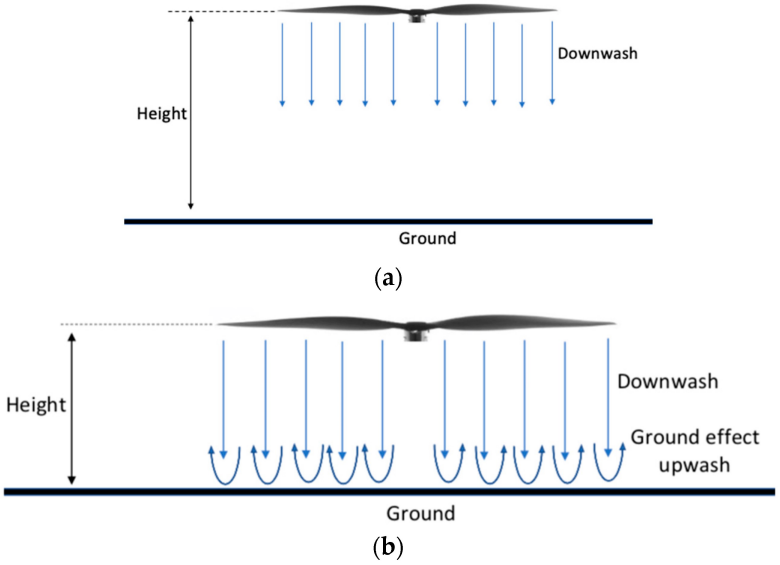


Figure 1.5: Rotor downwash flow field, (a) out-of-ground effect, (b) in-ground effect. [18]

Both these events are existing dangers to commercial technology, as it can affect flight safety without warning. Surge is a well studied phenomenon that can be prevented most of the time, but still requires pilots to avoid certain maneuvers or operational conditions to be on the safe side. Drones flight stability, on the other hand, is still a developing field which is growing commercially at neckbreacking speed. A severe instability in-flight when a drone is about to land may well cause casualties if it happens in a populated area, as we are not only talking about the run-of-the-mill miniature drones but of the autonomous quad-copters that some companies are developing to deliver goods and people in cities, [19]. It is therefore a safety necessity to develop tools in aeronautics that may help in predicting accurately these phenomena.

Accurate signal prediction opens up the door for efficient and safe control, automatically adjusting and correcting any kind of output signal whether the operational settings of an engine, the flight-path of a drone or the coordination of a bipedal robot. If fully integrated into a system, it will prevent from failures caused by sluggish control algorithms that simply react one the action has happened, rather than adjusting the system to prevent it.

1.3 Prediction of unstable systems

As mentioned previously, Machine Learning has become a powerful tool in any engineer´s arsenal to quickly categorize and predict signals coming from complicated systems. Several methods are becoming widely available, such as Deep Learning Neural Networks, Recurrent Neural Networks, Random Forest (ensemble methods), Support Vector Machine (kernel based methods) and K-nearest Neighbours.

A specific subset of the Reservoir Neural Networks also known as Echo State Networks is the main study of this thesis, with the objective to predict a real-life time series describing the behaviour of a complex system. The advantages and characteristic of this Neural Network will be further discussed in the next chapter, but one of the key aspects to focus on is in its adaptability to any kind of multivariate time series. This can be used to build a reservoir which its hyperparameters are automatically detected to be the best match through a genetic algorithm. By combining this two elements, the reservoir and an optimization algorithm, we get a system that can be fed any kind of time dependent signal and generate a prediction in the most efficient way.

2 | Reservoir Computing and Optimization

The thesis will present an ensemble of a Reservoir Neural Network along with an optimization algorithm, this will allow for the selection of the most appropriate setting for any kind of input signal. The fundamentals of Reservoir Computing (RC) and the optimization algorithm, a Grey Wolf Optimizer, are presented separately. The theory behind both entities is explored and the selection criteria explained, as several options were considered, specially in the case of the optimization algorithm.

As seen in the previous chapter, Machine Learning has been applied before on Predictive Maintenance without much impact, as safety regulations do not still support this effort until technological demonstrators prove their sturdiness. The following development attempts to create a tool capable of easily adapting to one of the many types of signals that may be generated on an aircraft, or for that matter on any kind of machinery, so as to predict its behaviour and readily detect when it may be about to fail.

2.1 General information on Neural Networks

Neural Networks may be referred sometimes in engineering as black box processes, in which you give it an input and sometime later obtain an output. This may be due to the mathematics involved in the process, but to avoid blindly plunging into unintended results it is necessary to at least skim over what are the involved fundamentals. An Artificial Neural Network models the relationship between a set of input signals and output signals using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs [20].

An ANN is assembled from neurons, or nodes, in a similar fashion to the human brain architecture, albeit at a much smaller scale. A total of 86 billion neurons make up the human brain [21], which is several times the order of magnitude of the most advanced ANN. For the last 50 years rudimentary neural networks have been used to simulate the human's approach to solving problems, but recently it has been successfully employed in speech and handwriting recognition programs, automation of smart devices and physical sophisticated models such as weather, fluid dynamics or tensile strength. ANNs are best applied to problems where the input data and output data are well-defined or at least fairly simple, yet the process that relates the input to output is extremely complex [20].

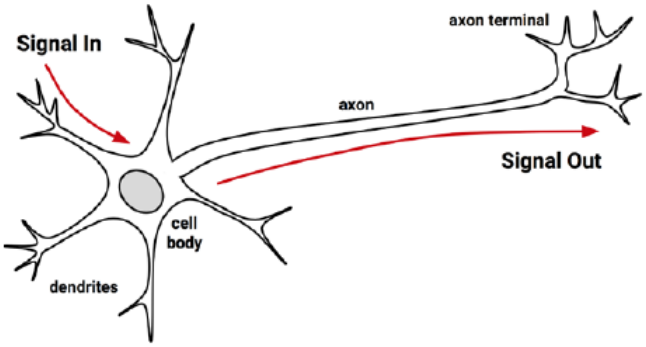


Figure 2.1: Signal pathway inside of a neuron in a human brain [20]

A neuron processes incoming signals, received by dendrites through biochemical processes, by weighing it according to importance and frequency criteria. It then accumulates the information until a threshold is reached and then fires up the output through the axon to its terminal, processed chemically there and passed to the next neuron through the synapse (see Figure 2.1).

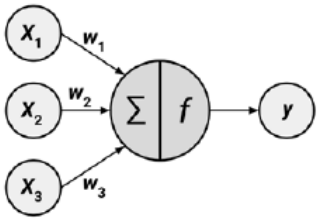


Figure 2.2: Signal pathway inside of a node in a neural network [20]

On the other hand, a neural network node works in a similar fashion than its biological counterpart. The diagram in Figure 2.2 shows that incoming signals, X_i, are weighted, w_i, according to a predetermined relative importance to each other. Those weighted inputs are then passed to the activation function, which eventually passes the output if the necessary threshold is reached, y. As it can be noted, the node tries to mimic the behaviour of a neuron in the way it synthesizes and reacts to the inputs it receives. Mathematically, it can be modelled simply by:

$$y(x) = f\left(\sum_{n}^{i=0} w_i X_i\right) \tag{2.1}$$

Therefore what defines a node from a neural network are the three following elements [20]:

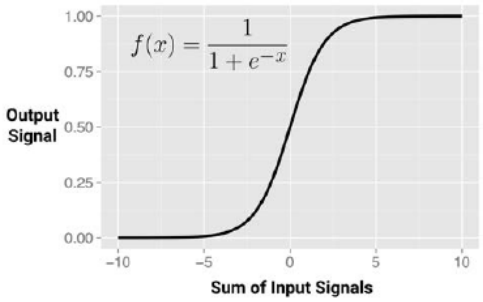
- **Activation function:** transforming the weighted inputs into an single output and then broadcasting it further into the network.
- **Network topology:** which determines the number of nodes, its interconnection and the number of layers.
- **Training algorithm:** specifies the manner in which weights affect the inputs by inhibiting or exciting them with respect to each other.

2.1.1 Activation functions

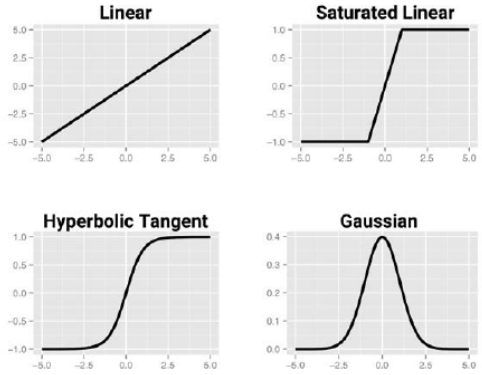
The activation function determines how the incoming information is transformed and passed on through the network. A threshold activation function needs a certain total input to be reached

for the output to be fired off, otherwise nothing is transmitted forth to the network. Although similar to what biological neurons work on, this threshold activation functions are not used in Neural Networks, using instead activation functions that can introduce desirable mathematical characteristics and accurately model relationships in the data.

The sigmoid activation function, Figure 2.3a, is the activation function most commonly used in Machine Learning. Its main incentive is that its output is not pure binary, but can fall in the range between 0 and 1, and is differentiable, a key feature for the creation of efficient Neural Network algorithms [20]. Other activation functions that also serve this purpose, Figure 2.3b, are the linear, the saturated linear, the hyperbolic tangent and the gaussian.



(a) Sigmoid function



(b) Assortment of activation functions

Figure 2.3: Most commonly used activation functions in Neural Networks [20]

Selecting any of these candidates greatly depends on the desired output signal, whether it will vary between $[0, 1]$, $[-1, 1]$ or $(-\infty, \infty)$, as it biases the neural network so it may fit the input data. A problem that may arise from the utilisation of these activation functions is solution squashing. For instance, in a sigmoid function any input lower or equal to -5 will result in an output of 0, while an input higher or equal to 5 will yield a 1. To solve this issue it is always recommended to squash the inputs, or better referred as normalise them, so they fall within a small range around 0. With normalisation, large values will not make smaller ones be lost due to solution squashing, improving and speeding up training.

2.1.2 Network topology

The ability of Neural Networks to learn is rooted in its topology [20], which is basically conditioned by the number of layers, whether the information is allowed to travel backwards and how many nodes are kept inside each of the layers. The more complex the network, the more capable it is to capture complex dynamics and subtle patterns, and is not only a matter of how many nodes are in the network in total but how they are distributed internally.

Nodes are therefore organized in layers, which helps to define the topology of a neural network depending on the number of them and how many nodes they contain. Input nodes form the first layer, as it processes the incoming signal and plugs it into the rest of the network, in which each of the nodes dedicates itself to a single dataset feature and transforms it according to its activation function. A middle layer, often referred to as hidden, processes the signals coming from the input before it reaches the output. Figure 2.4 depicts a network where the nodes of one layer are connected to all the nodes on the next. The output layer then interprets the signals coming from the nodes on the hidden layer and yields an output.

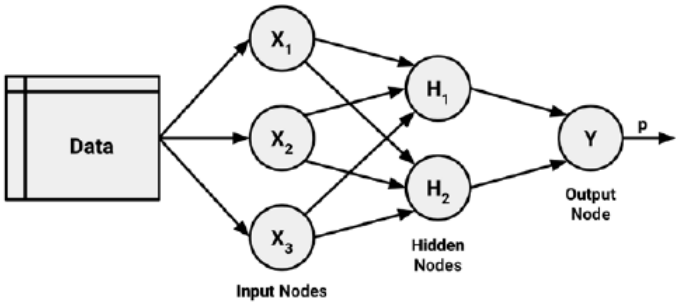


Figure 2.4: Example of a simple multilayered network [20]

How the information travels within the network is important to determine the type of training it requires and what tasks it is better focused at. Networks in which the input signal is fed continuously in one direction from connection to connection until it reaches the output layer are called feedforward networks [20]. Even with this restriction, the number of layers, inputs or outputs can be changed, making them highly adaptable. A Deep Neural Network is, for instance, one with a large number of hidden layers in which the weight between them has to be fine tuned. A recurrent network, also known as feedback network, does not restrict the flow of information in just one direction but allows for loops and turn-backs. This propagation behaviour tries to mimic that of biological brains and can be employed for the classification of extremely complicated patterns. To improve the capability to better understand sequences a delay can be introduced in the feedback loops, improving time series comprehension.

The amount of nodes on each of the layers is highly dependant on the task at hand, for instance the output nodes depend on how many outcomes need to be obtained, same logic applies with the input layer, as its node amount comes from the amount of ingested data. No rule has proven efficient to determine the amount of nodes necessary for the hidden layer, as it is highly dependant on the amount of inputs, training data and general complexity of the task. As a rule of thumb, the more complex the layer the better it will be at capturing the dynamics of the given data. A problem arises when increasing too much the number of nodes, as it may cause overfitting and yield poorly on following datasets as well as be computationally expensive and slow to train.

2.1.3 Training algorithm

The process of training a neural network is perhaps the most arduous and computationally intensive phase, as weights need to be adjusted for an specific task. In the late 1980s a technique appeared that allowed for a more efficient training, making neural network research leave the niche it had been in until then. The algorithm, which used a strategy of back-propagating errors, is now known simply as backpropagation [20]. This method is still relatively slow compared to other machine learning algorithms, but it is the most common for data mining. Its strength lies in its adaptability to numeric or classification problems, capacity to model complex patterns or avoid making assumptions of the underlying relationships in the data. On the other hand, it can be extremely slow for a complex topology, is prone to overfitting and leads to a difficult to interpret black box model.

The backpropagation algorithm iterates through many cycles of two phases, with each cycle being known as an epoch [20]. The weights are initially set at random, and then fine tuned trough the epochs until a stopping criteria is met. In the first phase, forward, the nodes are activated in sequence from the input layer with a certain weight until reaching the output layer and generating an output signal. For the second phase, backward, the output is compared to the target value and an error is obtained, which is propagated back through the network to diminish the error in the next run. Gradient descent is applied at each of the epochs to vary accordingly the weights of the

nodes. The algorithm uses the derivative of each node´s activation function to find the gradient of each of the incoming weights, therefore indicating by how much the error will be increased or reduced by a change in that weight. The learning rate is the amount in which the algorithm will attempt to change the weights to obtain the greatest reduction in error.

2.2 Reservoir Computing

A reservoir computer consists essentially of two components: a dynamical system that accepts an input signal and a trainable readout layer which is used to produce the output. The key claim is that if the system is complex enough, in principle, any computation can be achieved by simply training the readout layer for a desired functionality [22]. This definition encompasses a framework derived from Recurrent Neural Networks and first introduced by the works of Herbert Jaeger [23] and Wolfgang Maass et al. [24], as both these papers introduced respectively the Echo State Networks (ESN) and Liquid State Machines (LSM). The main focus of this type of Neural Network is time series processing, as input is applied over a period of time.

A reservoir is a random combination of activation functions that create a mathematical construct capable of simulating dynamics through the activation of internal states with the same or even more complexity as the original. Any system can be considered a reservoir given an input, an output and an interpretation layer to transform into readable information. Take for instance the use of Brain Organoids, brain-like 3D *in vitro* cultures, as a reservoir due to its capacity for non-linear dynamics and fading memory properties [25]. In the case of Reservoir Computing, the reservoir comes as an adjacency matrix that represents the relationships between the activation functions, building specific paths which are activated under specific inputs. Training a reservoir is nothing but observing during enough time how it reacts to a given set of information and making a linear correlation with a desired output, not delving in the complex mathematics that develop inside and only skimming the surface reaction for answers.

Computation on a reservoir is a two-step process, in which during the first an input is ingested to obtain the dynamic of the internal states over a period of time and during the second a readout layer is used to interpret those states. This first phase is computationally free, as there is no requirement to precisely design the topography and only needs to match certain criteria. The second phase is the one that requires an specific design so the results can be interpreted accordingly to the training data.

From the information processing point of view, the system operates as an abstract machine that changes the representation of the input [22]. The reservoir interprets the data coming from the input layer as changes over time of its internal states, which the readout layer then transforms back from those states into the output signal. This means the output layer does not perform any significant computation, as it only translates the internal state at that moment into real numbers it assigned through training.

A dynamical system is called computationally universal if it can be programmed through its initial conditions to perform any digital computation [26]. Therefore the reservoir must show a minimal amount of complexity within its internal states so it can be called universal computing. This dynamical system is then a placeholder of states, and what is known as reservoir is nothing by the collection of these internal states through a period of time.

Assuming a reservoir can achieve universal computation then to build one should be just as easy as to mimic an existing dynamical system. Most of the technological challenges that might surface in the reservoir computer building process are contained within the procedures of building an interface that can assess the states of the system [22]. The basis lying on the fact that the output layer just needs to interpret and is free, from an engineering standpoint, of constrictions. The capacity of a

reservoir to capture complexity is within its topography, as a universal computer it must show a complex and state rich configuration space. The higher the degree of complexity, the better the reservoir. This unfortunately, comes with a caveat, as it also requires to separate inputs so as to activate different regions of the configuration space.

The field of reservoir computing was born after research concluded that during the incremental weight improvement processes mostly the weights associated with the outmost layer are altered [22]. The internal weights are not as influential during training, and for instance this caused turmoil in Deep Neural Networks as backpropagation was not as useful. Fortunately, this problem does not apply to reservoir computing, as this is the characteristic that is exploited to enable for a faster training phase.

2.2.1 Echo State Network

Echo State Networks (ESN) provide an architecture and supervised learning principle for Recurrent Neural Networks (RNNs). The main idea is firstly to drive a random, large, fixed recurrent neural network with the input signal, thereby inducing in each neuron within this "reservoir" network a nonlinear response signal, and then combine a desired output signal by a trainable linear combination of all of these response signals [27].

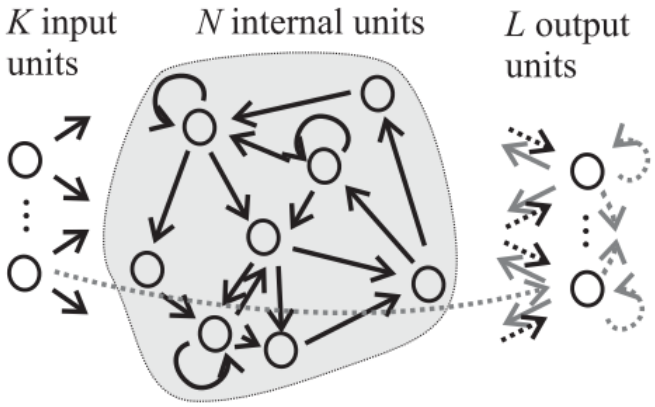


Figure 2.5: The basic Echo State Network architecture. Dashed arrows indicate connections that are possible but not required. [23]

On Figure 2.5 the architecture of an Echo State Network is depicted: showing an input layer made up of K units; a reservoir of N internal neurons randomly distributed and interconnected; and finally the L units output layer to interpret the internal states. To properly set this framework Jaeger [27] points out three basic steps:

- **Generate a random RNN:** a random dynamical reservoir is created by using a neuron model, based on the activation functions seen previously, with size N. All neurons within the reservoir are connected randomly to every input unit and an output layer is set up. In case output feedback is required, once again random all-to-all connections are set up between the reservoir and the output units.
- **Harvest reservoir states:** the reservoir is dynamically excited with the training data, a T length time series, and the internal states stored. The resulting $N \times T$ matrix stores this configuration changes for later, being the nonlinear transformations of the input which solely drives the reservoir. With output feedback, along with the training data the output signal is also fed back in.
- **Compute output weights:** a linear regression is employed to correlate the matrix of

internal states with the training data, which results in the weights required for the output layer.

The main foundation of the ESN is the Echo State Property (ESP), which specifies the required mathematical properties of a valid reservoir. A random complex system may not be appropriate for a reservoir. A good reservoir is one that is able to separate inputs; different external inputs should drive the system to different regions of the configuration space. This is called the separability condition [28]. This is not the only aspect of the property, but vital for the reservoir to be able to generate sufficiently different internal states given a set of inputs. The other is quite more abstract, and relates to the ability of the reservoir to hold the echos of previous inputs.

In order for the ESN principle to work, the reservoir must have the Echo State Property, which relates asymptotic properties of the excited reservoir dynamics to the driving signal. Intuitively, the ESP states that the reservoir will asymptotically wash out any information from initial conditions. The ESP is guaranteed for additive-sigmoid neuron reservoirs, if the reservoir weight matrix (and the leaking rates) satisfy certain algebraic conditions in terms of singular values. For such reservoirs with a tanh sigmoid, the ESP is violated for zero input if the spectral radius of the reservoir weight matrix is larger than unity. Conversely, it is empirically observed that the ESP is granted for any input if this spectral radius is smaller than unity. This has led in the literature to a far-spread but erroneous identification of the ESP with a spectral radius below 1. Specifically, the larger the input amplitude, the further above unity the spectral radius may be while still obtaining the ESP [27].

The property, mathematically developed by Jaeger [23], establishes that for a reservoir to hold the echo state it must resonate information of all the inputs through the consequent internal states. The reservoir is built from N neurons, whose internal weights are randomly generated in the form of a $N \times N$ matrix within the range $[-1, 1]$ from a uniform distribution. In practice, three basic ideas need to be considered for effectively constructing a reservoir that holds true to this principle [28].

- **The size N of the reservoir:** as the more complex the incoming signal the bigger it should be.
- **Sparsity of the reservoir:** or how much neurons are connected among each other. Connectivity should be low so as to improve computation speed as well as to guarantee the echo state property.
- **Spectral radius:** the spectral radius, $\rho(\mathbf{W})$, needs to be calculated and the matrix of internal weights $\mathbf{W}$ normalized accordingly. An spectral radius lower than one is recommended to guarantee the echo state property no matter the input. According to Lukoševičius [29], $\rho(\mathbf{W})$ should be selected to maximize performance, with the value 1 serving as an initial reference point.

Reservoir optimization following Lukoševičius' [29] advice will be explained when delving into the ESN-GWO System. Mathematically, the process starts by having an input signal $\mathbf{u}(t) \in \Re^M$, being a time series of length T with $t = 1, ..., T$. This input signal is introduced to the reservoir by means of another randomly generated input layer, $\mathbf{W}_{in} \in \Re^{N \times (1+M)}$. The input layer is generated following the same random generation as used for the internal weight matrix, only for this matrix all elements should be non-zero and does not require normalization. The reservoir, as shown previously, is described by an adjacency matrix $\mathbf{W} \in \Re^{N \times N}$.

Once $\mathbf{W}_{in}$ and $\mathbf{W}$ have been set up, the internal states for each of the inputs are harvested. Typically the neurons used for this reservoir are leaky-integrated discrete-time continuous-value units, with the update equations:

$$\hat{\mathbf{x}}(t) = \tanh(\mathbf{W}_{in} \cdot [1; \mathbf{u}(t)] + \mathbf{W} \cdot \mathbf{x}(t-1)) \tag{2.2}$$

$$\mathbf{x}(t) = (1-\alpha)\mathbf{x}(t-1) + \alpha\hat{\mathbf{x}}(t) \tag{2.3}$$

Where:

- $\mathbf{x}(t) \in \Re^N$ is a vector of internal states of the neurons in the reservoir.
- $\hat{\mathbf{x}}(t) \in \Re^N$ is its update vector for new inputs, all at a time step t.
- $\alpha \in (0, 1]$ is the leaking rate.
- $[1; \mathbf{u}(t)]$ is the vertically concatenated unitary value with the input at time step t.

The output $y(t)$ being:

$$\mathbf{y}(t + \Delta t) = \mathbf{W}_{out} \cdot \mathbf{x}(t) \tag{2.4}$$

The computation of $\mathbf{W}_{out}$ is a linear equation system. It is fixed by means of a ridge-regression, also known as *Tikhonov* regularization.

$$\mathbf{Y}^{target} = \mathbf{W}_{out} \cdot \mathbf{X} \tag{2.5}$$

$$\mathbf{W}_{out} = \mathbf{Y}^{target} \cdot \mathbf{X}^T \left(\mathbf{X} \cdot \mathbf{X}^T + \beta\mathbf{I}\right)^{-1} \tag{2.6}$$

Where:

- $\mathbf{X}$ is the internal states matrix $\mathbf{X} \in \Re^{N \times T}$ in which the columns are state vectors $\mathbf{x}$(t) for $t = 1, ..., T$.
- $\mathbf{Y}^{target}$ is the output objective $\mathbf{X} \in \Re^{M \times T}$ used for training.
- $\mathbf{W}_{out}$ is the output weight matrix $\mathbf{X} \in \Re^{M \times N}$ needed to interpret the internal states back to a usable output.
- $\mathbf{I}$ is the identity matrix.
- β is the regularization coefficient.

Once the training is finished, the internal states have been harvested and the ridge-regression applied, the prediction phase runs on Equations 2.2, 2.3 and 2.4 but with the input $\mathbf{u}(t)$ being the forecasted output of the preceding step $\mathbf{y}(t + \Delta t)$.

2.3 Meta-heuristic Optimization Algorithms

Meta-heuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality [30]. In the quest on finding the most fitting reservoir parameters in a fast and efficient way, the best answer is to apply a meta-heuristic optimization algorithm to find the best topography for a certain input signal.

These kind of algorithms do not guarantee that an optimal solution is going to be found, but do excel at exploring the probability space to find near-optimal solutions [30]. This property is essential for finding the best match for a given signal, as it is not a conventional optimization of a continuous function but the exploration of the topography of the reservoir.

Meta-heuristics is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [30]. Genetic algorithm (GA) is the best known example of this field, consisting on generating an initial random population which contains individual chromosomes which will evolve through iterations called generations. The best fitted chromosomes are selected at each generation and the new candidates, offsprings, are created by either combining existing chromosomes, splicing, or by altering a chromosome by mutation [30]. This approach has been successfully integrated along with a three-layer feedforward neural network to tune the hidden layers [31].

Simulated Annealing (SA) is an analogy on how an optimal atom configuration is found in statistical mechanics. During the search random solutions are generated and as the simulation progresses the requirements are stricter, mimicking the slow cooling of metallic annealing [30]. This algorithm has been used to improve the performance of a Convolutional Neural Network by finding suitable configurations [32].

Many more exist which are based on nature for their exploration logic, such as: Migrating Birds Optimization [33], Particle Swarm Optimization [34], Firefly Optimization [35] or Jellyfish Search [36]. In this work, the meta-heuristic selected has been Grey Wolf Optimizer, which already has been successfully employed along with a Long-Short Term Memory (LSTM) neural network for wind speed forecast [37].

2.3.1 Grey Wolf Optimizer

The Grey Wolf Optimizer is a meta-heuristic method proposed by Mirjalili et al. [38], based on the hunting patterns of grey wolves, mimicking their leadership hierarchy and hunting procedures. It is based on the fact that wolves have a rigid social structure, with a lead individual, known as alpha, making the decisions. It comes followed by the beta wolf, which enforces the alphas' decisions and gives feedback. The omega wolf is the lowliest of all, being the last to eat and basically acting as an scapegoat to prevent infighting.

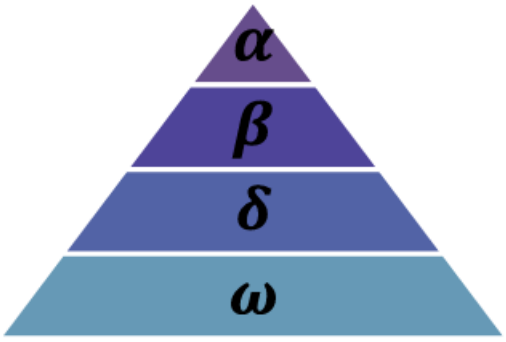


Figure 2.6: Hierarchy of grey wolfs in a pack (dominance decreases from top down). [38]

The main phases of grey wolf hunting are [38]:

- Tracking, chasing and approaching prey.
- Pursuing, encircling and harassing the prey until it stops moving.
- Attack towards the prey.

The mathematical model of this collective behaviour is what drives the optimization algorithm, candidates sharing information about the drive space, assisting in avoiding locally optimal solutions and exploring the objective space.

The first step is to establish that the fittest solution is the α wolf, followed respectively by the β and δ. Whatever remains of the population at that point are considered ω wolves, and follow the three lead wolves.

Wolves when hunting encircle prey and so it is modelled as:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{2.7}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2.8}$$

Where:

- t is the current iteration.
- $\vec{A}$ and $\vec{C}$ are coefficient vectors.
- $\vec{X}_p$ is the position vector of the prey.
- $\vec{X}$ indicates the position vector of the grey wolf.

$$\vec{A} = a(2\vec{r}_1 - 1) \tag{2.9}$$

$$\vec{C} = 2\vec{r}_2 \tag{2.10}$$

$$a = 2(1 - t/T) \tag{2.11}$$

With a (Eq. 2.11) linearly decreasing from 2 to 0 over the course of the iterations until the maximum number of them is reached, T. In the original algorithm this value is a vector too, but it does not specify anything of its characteristics except its initial and final value. Therefore, and to seek simplicity, it has been turned into a parameter. The randomly generated vectors, (r_1, r_2) are random vectors uniformly distributed in $[0, 1]$ and are of a dimension matching that of the wolves position vectors.

To hunt for the prey we need the location of the optimum, unfortunately this is an unknown. To simulate the hunting behaviour [38], the α, β and δ wolves are believed to have the most information on where that optimum is. Whenever a new population needs to be spawned, the three lead wolves are taken as a reference and the rest are forced to update their positions accordingly. In this process the distance between the α, β and δ wolves and the prey is modeled as:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \tag{2.12}$$
$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \tag{2.13}$$
$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{2.14}$$

Where:

- $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$ are the locations of leading wolves.
- $\vec{C}_1$, $\vec{C}_2$ and $\vec{C}_3$ represent random vectors.
- $\vec{X}$ specifies the location of the present solution.

The estimation of the distances allows for the update rules:

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \tag{2.15}$$
$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \tag{2.16}$$
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \tag{2.17}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2.18}$$

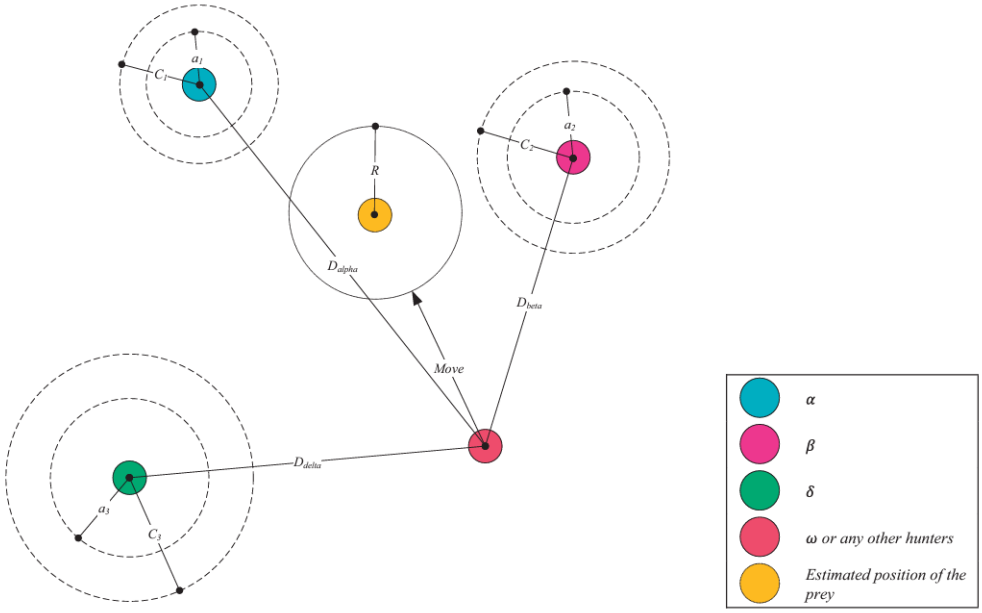


Figure 2.7: Position updating according to α, β and δ in a 2D search space. The final position is a random location withing a circle defined by where the lead wolves are in the search space. [38]

The parameter a given by Eq. 2.11 is modelled to signify the end of the hunt once the prey stops moving. $\vec{A}$ fluctuates in the interval $[-2a, 2a]$, when $\vec{A}$ is within the range $[-1, 1]$, the next position will be towards the prey.

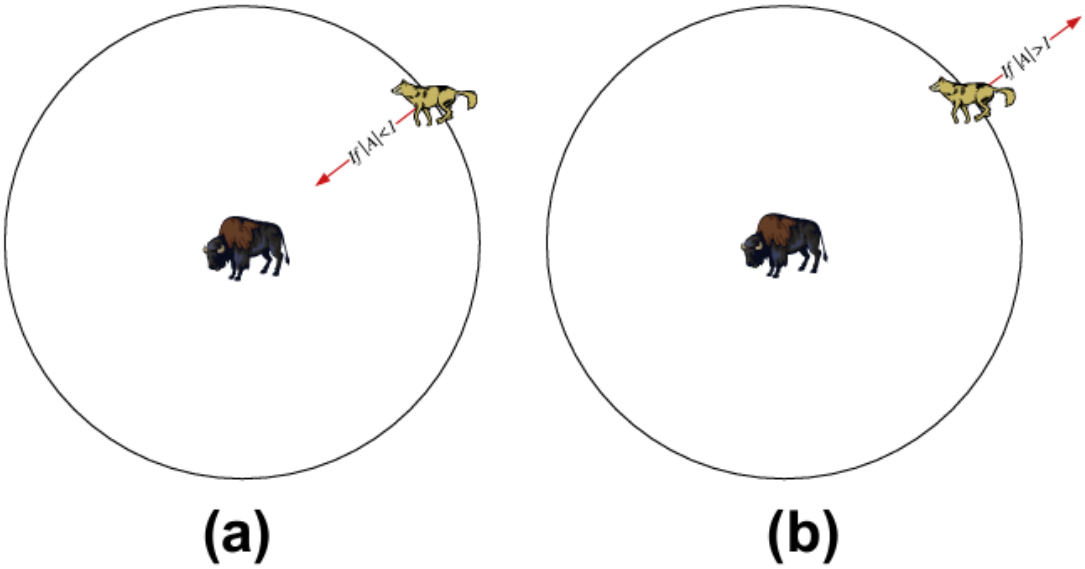


Figure 2.8: Attacking (a) or searching for prey (b) depending on the value of $|A|$ [38]

To promote exploration $\vec{A}$ may take different values: $|A| < 1$, which as mentioned previously focuses the wolves on the hunt; or $|A| > 1$, which drives them to diverge from the prey and find a better solution. Random vector $\vec{C}$ also plays part in the exploration phase of the optimization, according to Mirjalili et al. [38] this component provides random weights for prey in order to stochastically emphasize ($C > 1$) or deemphasize ($C < 1$) the effect of prey in defining the distance in Equation 2.7. And opposite to $\vec{A}$, $\vec{C}$ does not linearly decrease with a in order to always enforce exploration [39].

The pseudo code for the Grey Wolf Optimization has been altered with respect to the original [38], as in this case some bounds need to be set up for the exploration phase, please refer to the pseudocode at Algorithm 1. When the new population is generated their location is checked to see whether they are within bounds or not. If the wolves are not within bounds, the generation is repeated again and again until they are. It has been proven through testing that, in some fringe cases, the new population created is always out of bounds. When this happens a countdown is activated, and once a wolf has been repopulated a certain amount of times the out of bounds coordinates in the exploration space are changed to ones randomly within the sphere the three lead wolves create. This allows for exploration while guaranteeing the wolves will not remain stuck in an out of bounds position.

Algorithm 1 Grey Wolf Optimizer with bounds enforcement

Maximum number of iterations T
Maximum number of hunt attempts n_{hunts}
Total population individuals w
Initialize upper coordinate bounds $UB_i \quad (i = 1, 2, ..., n)$
Initialize lower coordinate bounds $LB_i \quad (i = 1, 2, ..., n)$
Initialize the grey wolf population $X_i \quad (i = 1, 2, ..., n)$
Check that $LB_i < X_i < UB_i$
Initialize a, A, C
Calculate the fitness of each search agent
X_α = best search agent
X_β = second best search agent
X_δ = third best search agent
$t = 0$
while $t <= T$ **do**
    $n_{attempts} = 0$
    for $i = 0 : w$ **do**
        Update the position of current search agents by Equation 2.18
        if $LB_i > X_i$ OR $X_i > UB_i$ **then**
            Update again the position of current search agents by Equation 2.18
            $n_{attempts} = n_{attempts} + 1$
            if $n_{attempts} > n_{hunts}$ **then**
                Update only the out of bound coordinates from the search agents to ones randomly located within the α, β and δ locus
            end if
        end if
    end for
    Update a, A, C
    Calculate the fitness of each search agent
    Update X_α, X_β, X_δ
    $t = t + 1$
end while
return X_α

2.4 ESN-GWO System

The Grey Wolf Optimizer-Echo State Network System, GWO-ESN System, receives an input signal and runs through the hyperparameter space to find the topography best suited for that specific prediction. Once the system finds the optimal settings for a reservoir it is not necessary to run it again for generating new predictions off an updated dataset. For instance take the example of the power consumption of a device during three years, the training data could be everything but the last two months and the validation data those last two months. When the appropriate topology for the reservoir has been found, new results can be quickly generated bimonthly without having to optimize the network. This can be repeated over time until the mean square error of the predictions diverges from a tolerance and the process begins anew.

To optimize the topography of a reservoir certain hyperparameters must be defined. Lukoševičius' tricks of the trade [29] serves as a foundation for defining the optimal configuration vector which the wolves will try to hunt down. As mentioned previously the internal weight matrix, $\mathbf{W}$, and the input weight matrix , $\mathbf{W}_{in}$, are randomly generated following some requirements so as to follow the echo state property. The following reservoir parameters are the ones that build up the configuration space:

- N: size of the reservoir.
- α: leak rate.
- a_{Win}: input matrix column(s) scaling factor.
- c: internal weight matrix density.
- K_ρ: spectral radius scaling factor.
- β: Tikhonov regularization coefficient.

The size of the reservoir, N, is fundamental to appropriately capture the inherent complexity of the input signal. As a rule of thumb, the more challenging the task the bigger the reservoir should be. Some limitations should be established too: firstly to avoid overfitting and secondly to avoid unnecessarily computationally expensive calculations. Lukoševičius [29] recommends starting with smaller reservoirs and then scale to bigger ones, always taking into consideration that N should be at least equal to the estimate of independent real values the reservoir has to remember from the input to solve its task.

The leak rate, α, is the speed of the reservoir update dynamics discretized in time. The closer to 1 it gets the more it will forget about the previous internal states the reservoir went through. The leaking rate needs to match the speed of the dynamics of the predicted signal and the training data [29], usually being preferred to remain close to 0 so slow dynamics on the internal states are retained and the duration of the short-term memory increases.

The input matrix column scaling factor, a_{Win}, needs to be as many as different inputs are ingested in the reservoir, always being at least two for this system. Each of the scaling factors apply to a specific column of the input weight matrix, with the first always being the one for the unitary value that accompanies the input and from the second onward for the input itself. In case the input signal is more than a single time series it is recommended to group them and apply the same factor for those that contribute differently to the task. Input scaling regulates the amount of nonlinearity of the interal states of the reservoir and how much the current state is influenced by the input against the memory it holds of previous states [29]. Linear tasks benefit from small scaling values, as the activation function can operate in the 0 range where it is mostly linear. For nonlinear tasks a bigger scaling forces the activation function to act as a binary switch, closer to -1 and 1 values.

The sparsity of the reservoir, c, represents the connectivity among the neurons in the reservoir. Lukoševičius [29] states that when connecting each reservoir node to a small number of other nodes (e.g., 10) on average, irrespective of the reservoir size, computation speeds up. Sparse matrix representations are recommended as they give slightly better performance, but in general this hyperparameter does not affect much the quality of the prediction. As Jaeger indicates [27], some authors have developed fully connected reservoirs which work as well as sparsely connected ones. This goes against the initial believe that sparsity promotes the decomposition of reservoir dynamics into loosely coupled subsystems which in turn gives the desirable trait of large variation in internal states. The main advantage to use sparse matrices is the reduction in computational cost, as with a fixed low number of connections the cost increases linearly with size instead of quadratically.

The spectral radius scaling factor, K_ρ, is used to enforce the echo state property through establishing the maximal absolute eigenvalue of the internal weight matrix, $\mathbf{W}$. A random sparse matrix is generated, its spectral radius computed ($\rho(\mathbf{W})$), then $\mathbf{W}$ is divided by the spectral radius to yield a matrix with unitary spectral radius and scaled with $\mathbf{K}_\rho$ to adapt to the specific input ingested in the reservoir (Eq. 2.19). When $\rho(\mathbf{W}) < 1$ the echo state property is ensured in most situations, and even can hold for values greater or equal to 1 in some non-zero inputs. This happens due to the activation-squashing nonlinearities, strong inputs "squeeze out" the autonomous activity from the reservoir activations [29]. The spectral radius should be selected to maximize the performance, with 1 serving as a reference point and taking into consideration that it should be greater in tasks requiring longer memory of the input.

$$\mathbf{W}_{scaled} = \mathbf{W}_{initial} \frac{\mathbf{K}_\rho}{\rho(\mathbf{W}_{initial})} \tag{2.19}$$

The last hyperparameter to be tuned is the Tikhonov regularization coefficient, β, which serves to obtain the output layer through ridge-regression. A $\mathbf{W}_{out}$ matrix with high values is an indication of a very sensitive and unstable solution, which poses a serious problem when the reservoir is making predictions on its generative mode, where the next input is the prediction from the previous step. To ensure the output´s stability the regularization coefficient is also optimized in the same manner as the rest of hyperparameters, even though Lukoševičius [29] recommends trying different values along the same reservoir configuration. The inclusion of this parameter in the optimization seeks reducing the necessary steps for optimizing a reservoir as the wolves will hunt the value that best serves the output while exploring the topography of the reservoir. It is imperative to adjust properly the regularization coefficient as it prevents overfitting and feedback instability.

2.4.1 System workflow

Once the system is initialized it will determine autonomously the best suited set of hyperparameters for a given input signal with minor tweaking necessary code-side. Figure 2.9 shows the workflow of the system, as the data is processed, the GWO algorithm is called and then the reservoir runs for evaluating the wolves. The particulars for how to prepare each of the inputs is given along the exposition of the experimental data, as this section focuses on the generalities of the process and how the previously explained optimization algorithm and reservoir interact with each other.

Once the data is ingested it gets normalized according to Eq. 2.20, where $x_{norm}(t)$ is the normalized signal value at time t, $x(t)$ is the signal value at time t, μ is the mean of the signal and σ the standard deviation. The data needs to be normalized, as per Lukoševičius [29] and Zio [13], to guarantee the outliers do not "squash out" the rest of the signal when applied to the activation function.

$$x_{norm}(t) = \frac{x(t) - \mu}{\sigma} \tag{2.20}$$

As for the training and testing data selection there is a convention for the optimal relation between both, 70% training and 30% testing, but depends on the task at hand. Once again the specifics will be shown during the experimental data section but for the most part this convention holds true. Another vital aspect is establishing the initial transient to be discarded during training, as Lukoševičius [29] states the initial internal states are affected by the fact that the first internal state is null. How much of those initial internal states is discarded falls under the "find out" doctrine, as sometimes with short sequences not much can be discarded without nothing being left. For this system the initial transient for the training to ignore has been fixed at 10% of the total input data, so that the reservoir may forget the unnatural states that are not often activated, while keeping the "warmed-up" states that hold better information on the dynamics of the system.

Once the GWO has been initialized, the initial population of wolves is created by placing each of the hyperparameters randomly between the upper and lower bounds. This allows for an increased coverage of the configuration space so the best set up can be found for the given input signal. Then the wolves are run through the reservoir and ranked according to the normalized Root Mean Square Error (nRMSE) [39] calculated along the length of the testing phase, Eq. 2.21.

$$\delta_i \equiv y_i - Y_i^{target} \tag{2.21}$$

$$\Delta_i \equiv \frac{\delta_i}{\max|\delta_i|} \tag{2.22}$$

$$nRMSE = \sqrt{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \Delta_i^2}. \tag{2.23}$$

Now the process of optimization begins, churning through potential wolves as long as the maximum number of iterations has not been reached. The process of selecting the new wolf population, and discarding those out of bounds, has been explored previously. Suffice to say that it is during the latest stages of the optimization where most out of bounds wolves are generated, as those that try to explore the configuration space further from the prey begin probing the limits set to the topography.

Once the maximum number of iterations has been reached, the GWO provides the hyperparameter combination best suited to the input data and it is run once more. This time the predicted signal is stored, as when called during the optimization the output is just the RMSE computation, and generates the graphs and files to depict the predicted signal during the testing phase for validation. When more forecasts on updated data are required, the GWO is not needed and the reservoir generates new predictions working on the reservoir obtained by the optimal hyperparameters.

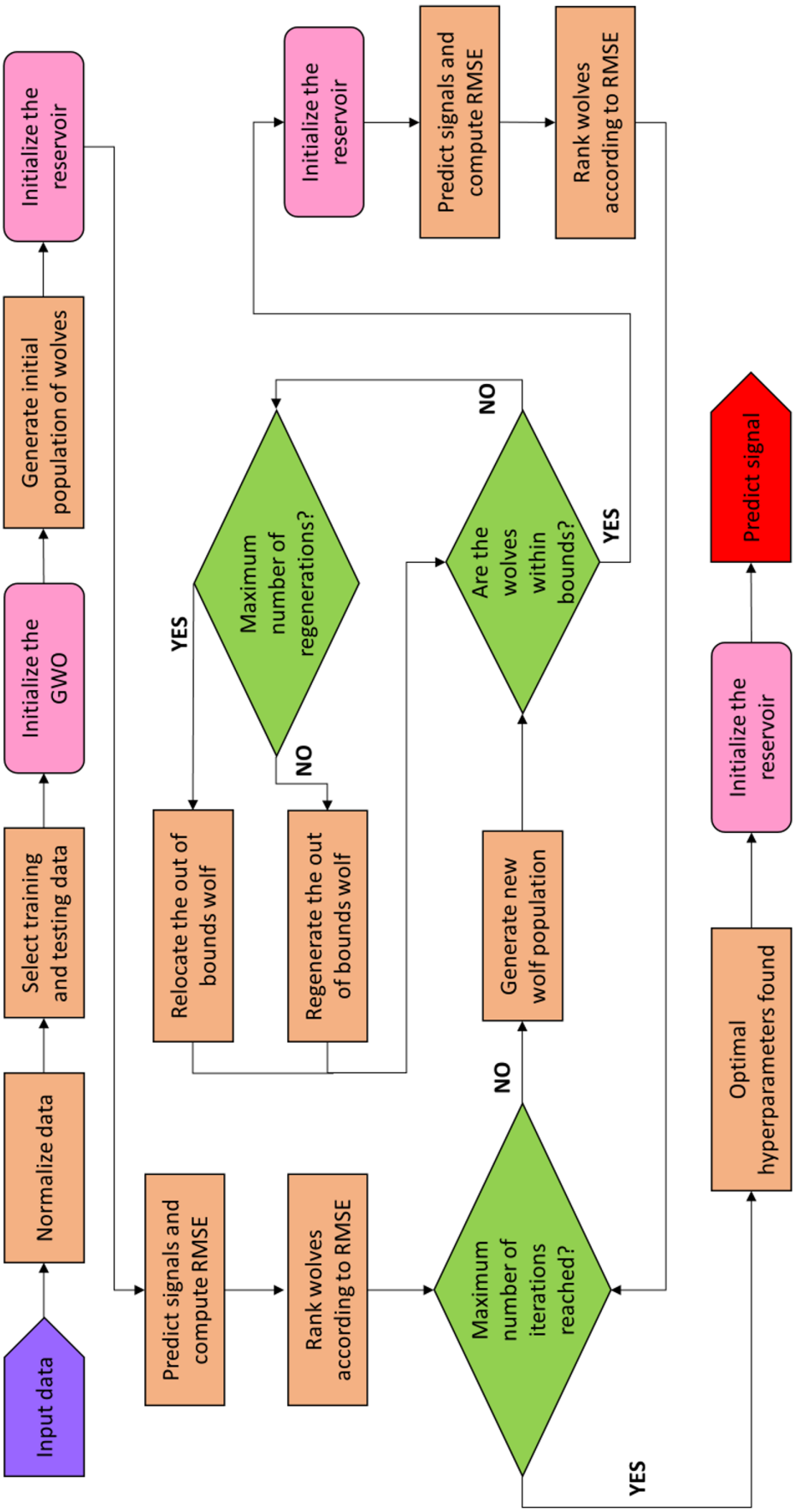


Figure 2.9: ESN-GWO System flowchart

3 | Experimental data

Three datasets have been employed to test the system under different circumstances, covering a wide range of potential possibilities for usage. The main requirement when looking for data to evaluate performance was to predict time series data which has a straightforward relationship between input and output but the journey between them is obscured by the system it comes from. For instance, a large scale facility has a certain power consumption that varies through time, so to predict how much power is consumed in a conventional manner all devices to the grid should be taken into account and how themselves behave over time. This is a complicated system, with multitude of individual elements that fluctuate themselves and with a global power requirement almost impossible to study unless simplifications are introduced. The relationship between time and power consumption is not evident a priori but can be processed by the reservoir.

The first dataset, the Mackey-Glass function, is nothing more than a commonly adopted time series used to benchmark in reservoir computing. The most relevant are the remaining two, the sensor data from turbofan engines and the power consumption of a large resort facility. The first is valuable for predicting the behaviour of engines during service, specially to analyze what can be taken as indicators of the remaining useful life. The second could be seen as unrelated, but power consumption prediction is relevant no matter if it is from avionics, an airport or any other kind of facility.

3.1 Mackey-Glass time series

The Mackey-Glass equation is the nonlinear time delay differential equation theorized by Glass et al. [40] which displays a range of periodic and chaotic dynamics. The mathematical formulation, Equation 3.1, is composed of three positive parameters, γ, β, n and y_r which is the value of y at time $(t - \tau)$. The time series depicted in Figure 3.1 is replicated by using the parameters $\gamma = 0.1$, $\beta = 0.2$, $n = 10$ and $\tau = 17$.

$$\frac{dy}{dt} = \beta \frac{y_r}{1 + y_r^n} - \gamma y \qquad \gamma, \beta, n > 0 \tag{3.1}$$

The prediction of this time series was just part of the testing for the development of the reservoir, and is a functional example to test the ensemble too. The upper and lower bounds of the hyper-parameters are generic, without much fine tuning unlike the CMAPSS and power consumption datasets. The size of the reservoir, N, is limited by really loose bounds with the maximum of a 2000 neurons being inefficient computationally for a time series without the chaotic dynamic of the other two datasets. In this case there are two input matrix scaling factors, a_{Win}, as one is for the unitary value and the other for the y value. The reasoning behind the connectivity, c, is to guarantee sparseness with any of the potential reservoir sizes. As for the spectral radius scaling factor, K_ρ, and the Tikhonov regularization coefficient, β, the widest range has been set up for both to explore the effects on the prediction.

	N	α	$a_{Win}(1)$	$a_{Win}(2)$	c	K_ρ	β
Lower bound	100	0.01	0.01	0.01	0.001	0.01	$1{\cdot}10^{-8}$
Upper bound	2000	0.9	100	100	0.3	0.99	100

Table 3.1: Upper and lower bounds for the Mackey-Glass dataset hyperparameters

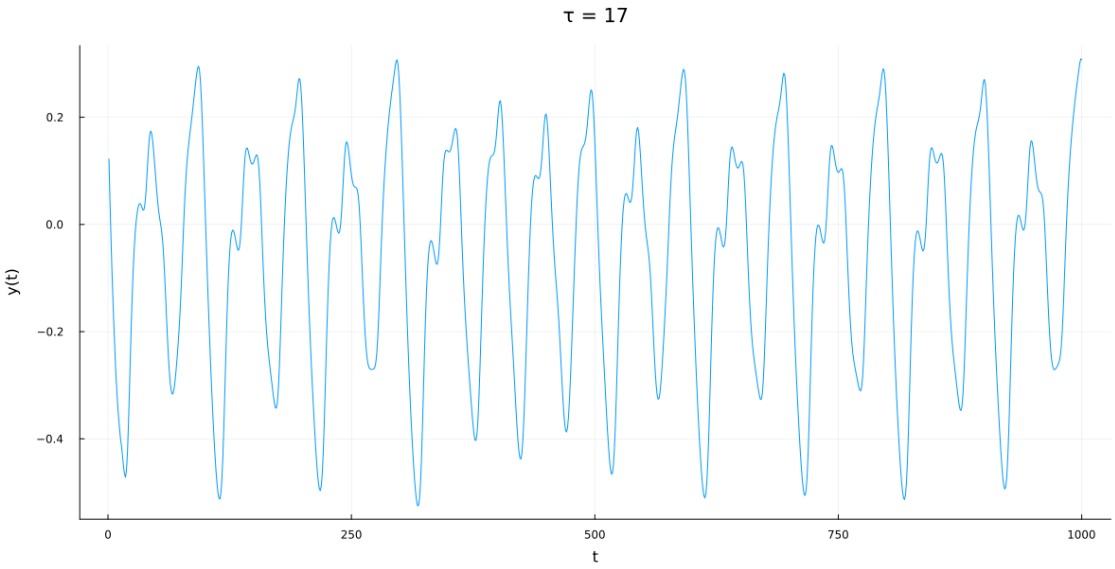


Figure 3.1: Mackey-Glass series with a time delay n=17

3.2 CMAPSS dataset

The *Commercial Modular Aeropropulsion System Simulation* dataset (CMAPSS), was developed by Saxena et al. [41] as part of NASA's Prognostic Center of Excellence Data Repository [42]. This dataset contains sensor information from a turbofan engine obtained from a thermo-dynamical simulation model.

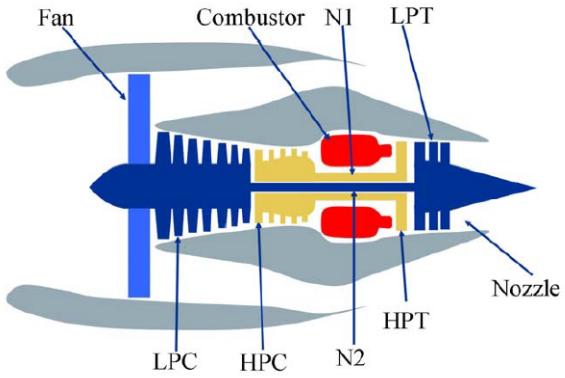


Figure 3.2: Simplified diagram of the turbofan engine simulated in CMAPSS [41]

Figure 3.2 shows the simplified diagram of the turbofan engine used to run the CMAPSS simulations. A turbofan engine is a dual flow engine, with a bypass flow only propelled by the fan and a main flow which is compressed, combusted and later passed through a turbine which extracts the power required for all moving components, including the fan. The fan and the Low Pressure Compressor (LPC) are fed from the Low Pressure Turbine (LPT) through the N2 axis. The High Pressure Compressor (HPC) is fed from the High Pressure Turbine (HPT) through the N1 axis.

Several runs are contained in this dataset, each considered to be different turbofan engines of the same make and model with different initial degradation levels in their components. This

degradation will affect how the engines behave over time, as damage and backlash can severely impact performance. These time series are measured in cycles, C, an unspecified unit of time that does not defeat the purpose of the thesis as the state on the previous time-step affects the next.

For the simulations three operational settings determine the turbofan behaviour: altitude, Mach and TRA. Altitude provides the environmental conditions of the flow coming into the engine as also does the flight Mach, while the TRA (Throttle Resolver Angle) is the fraction from maximum amount of fuel injection into the combustion chamber. In Table 3.2 the nominal operating conditions are also shown, as Altitude and Mach are given as the difference from nominal conditions during design cruise (the design objective for the engine to achieve during the cruise phase).

Symbol	Description	Units	Nominal value	Dataset ID
A	Flight cruise altitude	ft	42000	Operational Setting 1
M	Flight Mach number	-	0.84	Operational Setting 2
TRA	Throttle Resolver Angle	%	100	Operational Setting 3

Table 3.2: Operational settings and nominal values for the CMAPSS dataset [41]

Twenty-one signals comprise the output for the CMAPSS simulation, Table 3.3, in which data such as temperatures and pressures, both dynamic and total, are registered at different stages of the turbofan. Some of these outputs are stable through time for design reasons, for instance EPR (Signal 10) is the fraction between the pressure at fan inlet and the pressure at the exhaust nozzle (P2/P5). As a design principle it needs to remain stable through the operation of the engine given that the turbine remains critical, no more flow rate can get through it, and isentropic efficiency is also stable [14]. Any deviation in the EPR may be caused by a change in the internal conditions, as aging components ultimately lead to higher combustion temperatures, which in turn need to be offset by the pressure ratio to guarantee the design cruise thrust.

Symbol	Description	Units	Dataset ID
T2	Total temperature at fan inlet	º R	Signal 1
T24	Total temperature at LPC outlet	º R	Signal 2
T30	Total temperature at HPC outlet	º R	Signal 3
T50	Total temperature at LPT outlet	º R	Signal 4
P2	Pressure at fan inlet	psia	Signal 5
P15	Total pressure in bypass-duct	psia	Signal 6
P30	Total pressure at HPC outlet	psia	Signal 7
Nf	Physical fan speed	rpm	Signal 8
Nc	Physical core speed	rpm	Signal 9
EPR	Engine Pressure Ratio (P50/P2)	-	Signal 10
Ps30	Static pressure at HPC outlet	psia	Signal 11
ϕ	Ratio of fuel flow to Ps30	pps/psi	Signal 12
NRf	Corrected fan speed	rpm	Signal 13
NRc	Corrected core speed	rpm	Signal 14
BPR	Bypass Ratio	-	Signal 15
farB	Burner fuel-air ratio	-	Signal 16
htBleed	Bleed Enthalpy	-	Signal 17
Nf_dmd	Demanded fan speed	rpm	Signal 18
PCNfR_dmd	Demanded corrected fan speed	rpm	Signal 19
W31	HPT coolant bleed	lbm/s	Signal 20
W32	LPT coolant bleed	lbm/s	Signal 21

Table 3.3: Output signals for the CMAPSS dataset [41]

In the CMAPSS dataset the 21 output signals are directly affected by each other with the operational settings determining the conditions under which the turbofan engine is working. The altitude and Mach number are fundamentally indicators of the intake, total and dynamic, pressure and temperature. The TRA depicts how much fuel is fed to the engines, even though the rela-

tionship between this setting and ϕ (Signal 12) is not correlative, and is essentially the amount of demanded thrust. None of the signals act independently, as for instance a minor change in the conditions of the compressor affects the whole system. Predicting how the system is going to react under certain conditions, and specially how each of the components behave, can assist in detecting a specific failure in the engine when certain signals are off from their nominal values.

In Figures 3.3, 3.4, 3.5, 3.6, 3.7 and 3.8 the time series for the 3 operational settings and 21 output signals are depicted, belonging to the Unit 49 in the "Test 1" file. This has been the selected data for this thesis by the available length, as it provides enough information for the reservoir to parse through and learn on the behaviour of the turbofan engine. Other Units are also suitable and will be later shown when the hyperparameters selected by the ensemble for Unit 49 are used to run a prediction on them.

The upper and lower bounds for the hyperparameters are shown in Table 3.4, these values depend mainly on the size of the time series and its complexity. The size of the reservoir, N, should be, following Lukoševičius' [29] advice, enough to capture the dynamics of the signal without compromising computational costs. A reasonable estimate would be that it requires at least 50 nodes to be equal to the estimate of independent real values from the input the reservoir has to remember and 500 as the maximum computational capacity to expend in a time series this short. The input matrix scaling factor, a_{Win}, needs to be threefold for this dataset. The first factor is for the unitary value appended to the input, the second is for the operational settings and the third is for the output signals. The bounds have been established following Xu's et al. [43] example using an ensemble of ESN, in this case reducing the orders of magnitude but allowing the scaling factors to vary wildly between them so the inputs that are more relevant to the reservoir dynamics are present the most.

	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β
Lower bound	50	0.01	0.01	0.01	0.01	0.001	0.01	$1{\cdot}10^{-8}$
Upper bound	500	0.6	100	100	100	0.3	0.9	100

Table 3.4: Upper and lower bounds for the CMAPSS dataset hyperparameters

The connectivity, c, according to Lukoševičius [29] should be fixed so that each neuron is connected to around another 10. As this value is approximate and the reservoir size fluctuates during the optimization process it is determined that the connectivity should be limited between 0.1% and 30% of the size of the reservoir, as between both values the matrix can still be considered sparse and all possible combinations are explored. For the spectral radius scaling factor, K_ρ, the minimum is arbitrary as it just needs to be sufficiently close value to 0 while the maximum depends on how much of the input the reservoir needs to remember. As the degradation of the turbofan engine begins as soon as it is used the maximum has been put at 0.9 so the reservoir can remember the whole input signal. As for the Tikhonov regularization coefficient, β, Lukoševičius [29] states that the optimal value can vary magnitudes of size, depending on the exact instance of the reservoir and length of the training data. Another key point mentioned is that the value should never be null, as it then would be a regular linear regression. For this dataset the lower and upper bounds have been set at $1{\cdot}10^{-8}$ and 100 respectively so as to explore properly the effect on stability and overfitting of the regularization coefficient.

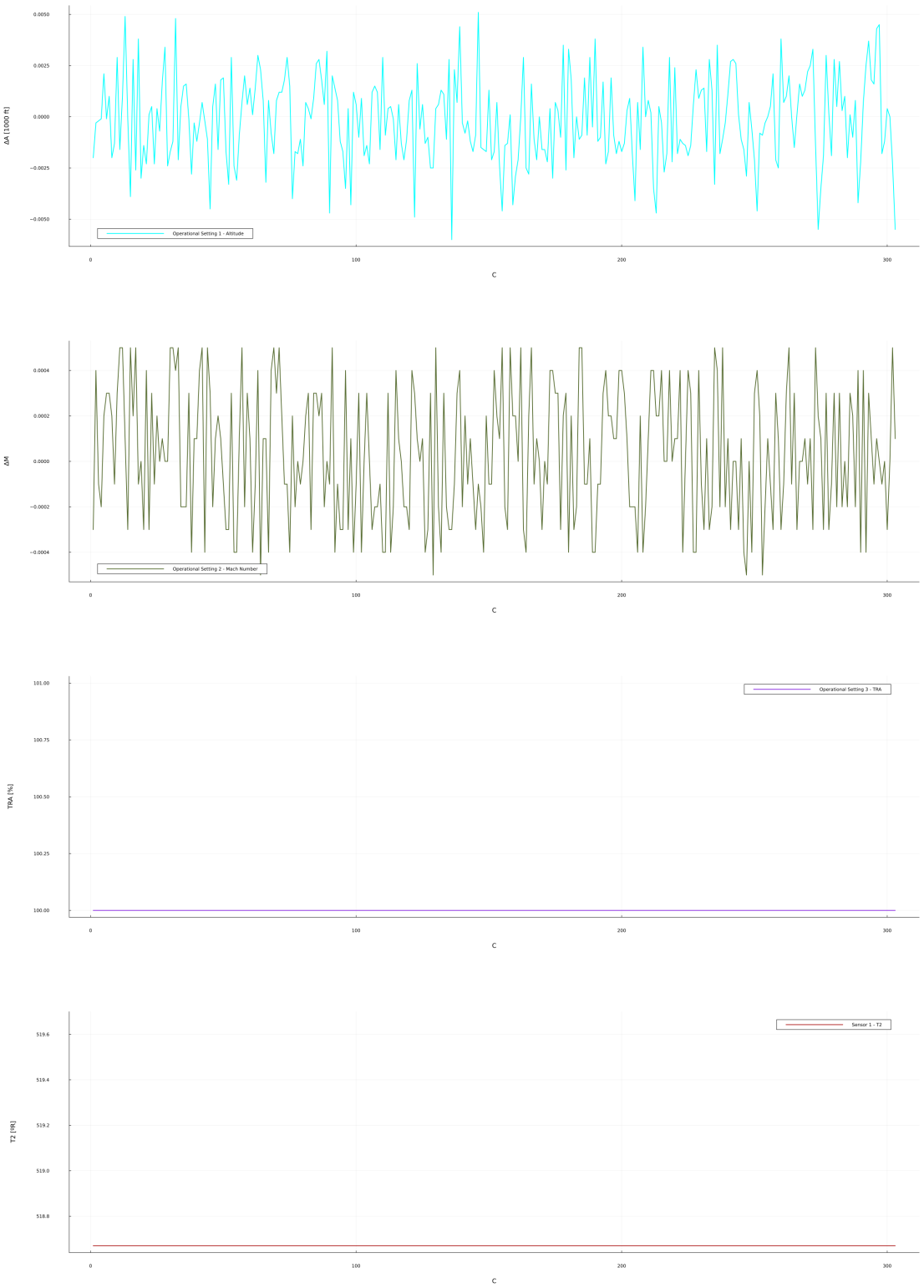


Figure 3.3: CMAPSS dataset Test 1, Unit 49: Altitude, Mach, TRA and T2

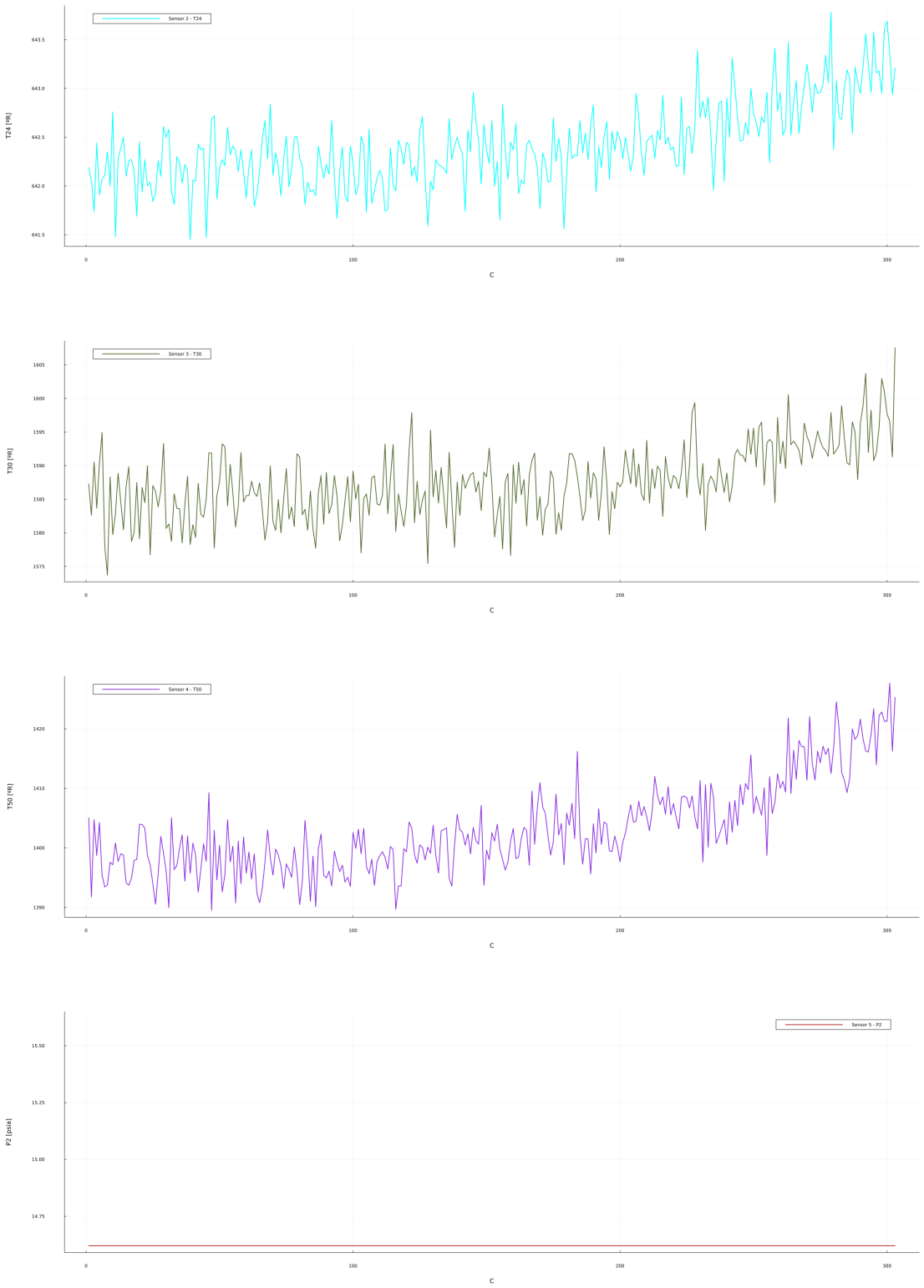


Figure 3.4: CMAPSS dataset Test 1, Unit 49: T23, T30, T50, P2

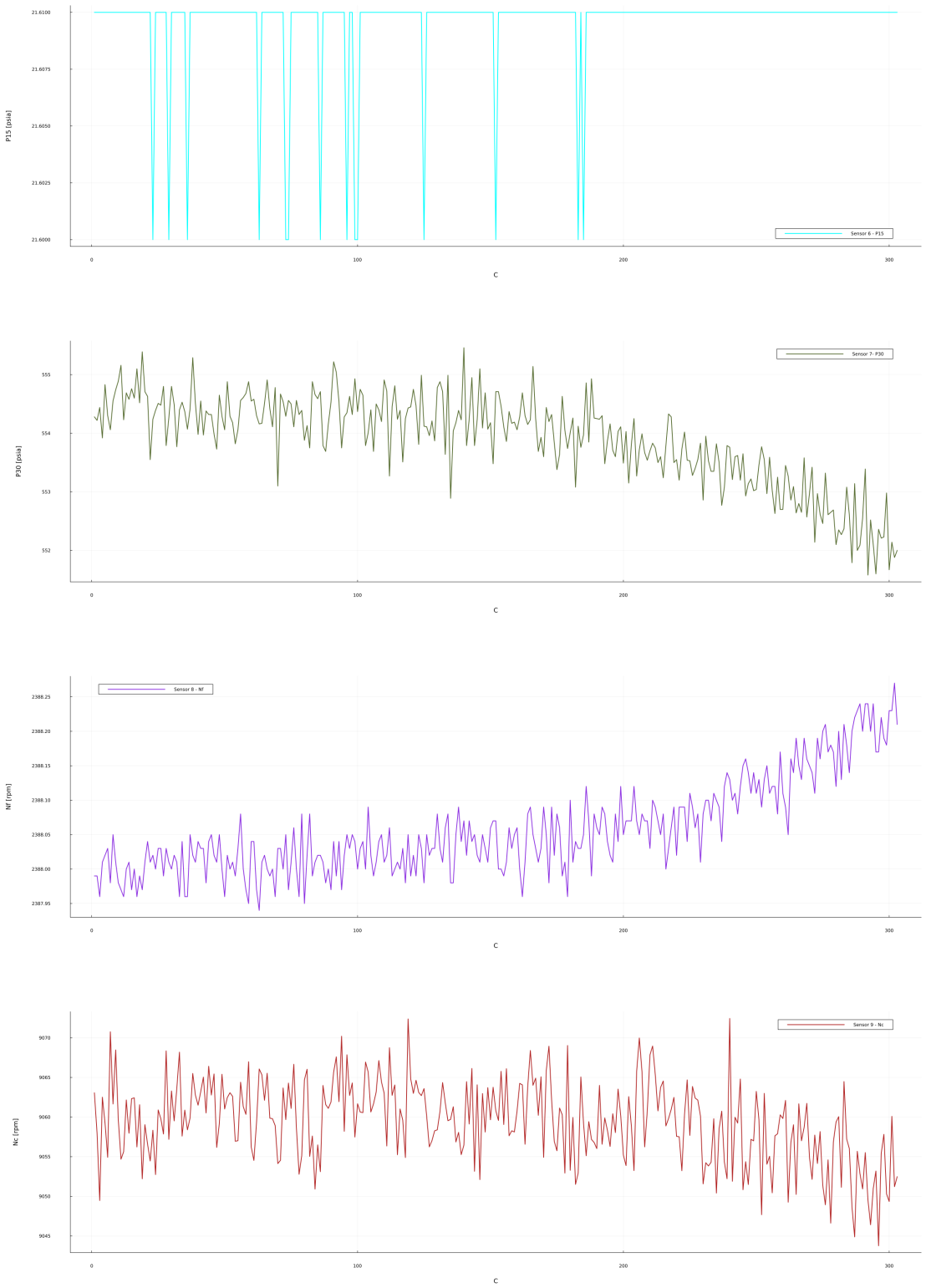


Figure 3.5: CMAPSS dataset Test 1, Unit 49: P15, P30, Nf, Nc

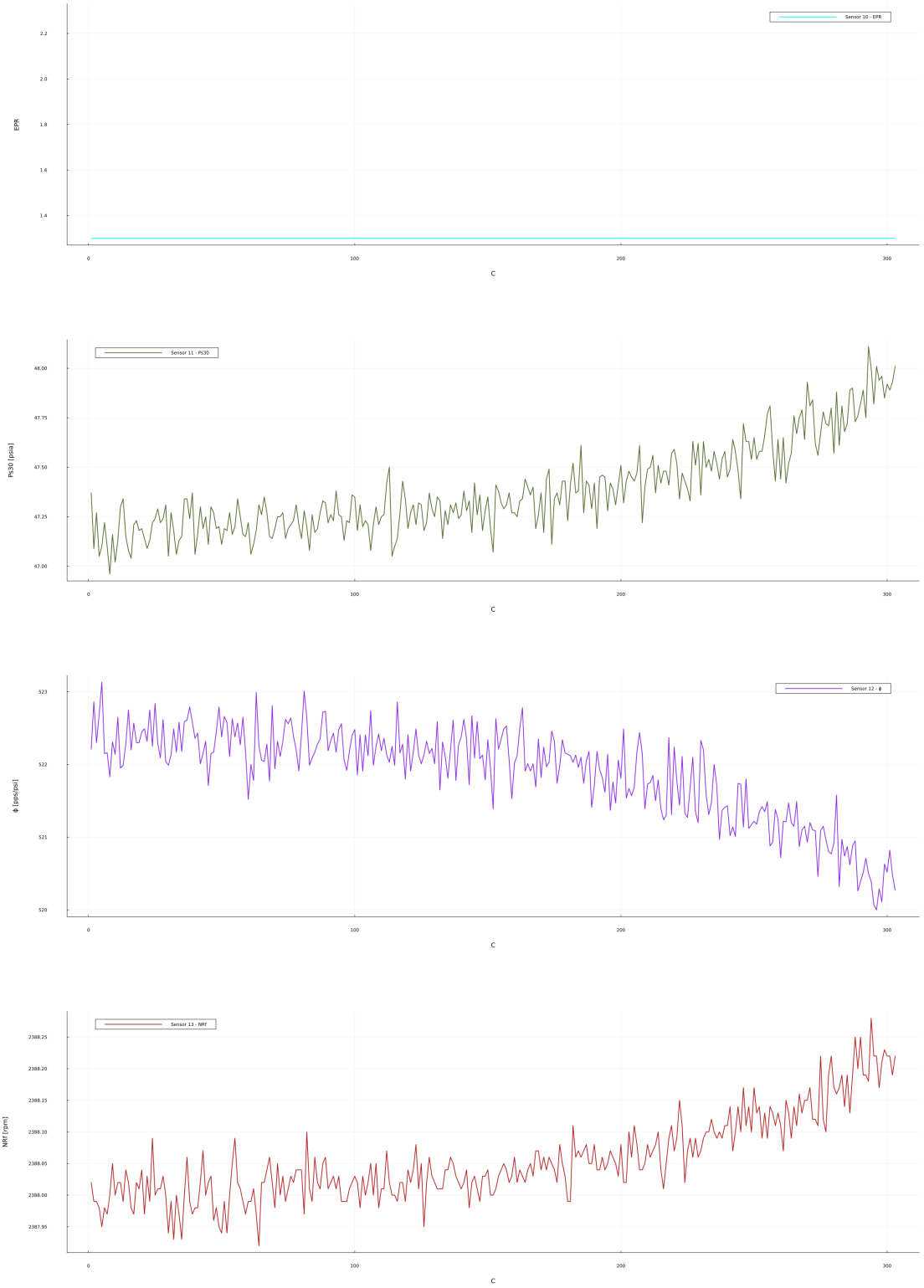


Figure 3.6: CMAPSS dataset Test 1, Unit 49: EPR, Ps30, ϕ, NRf

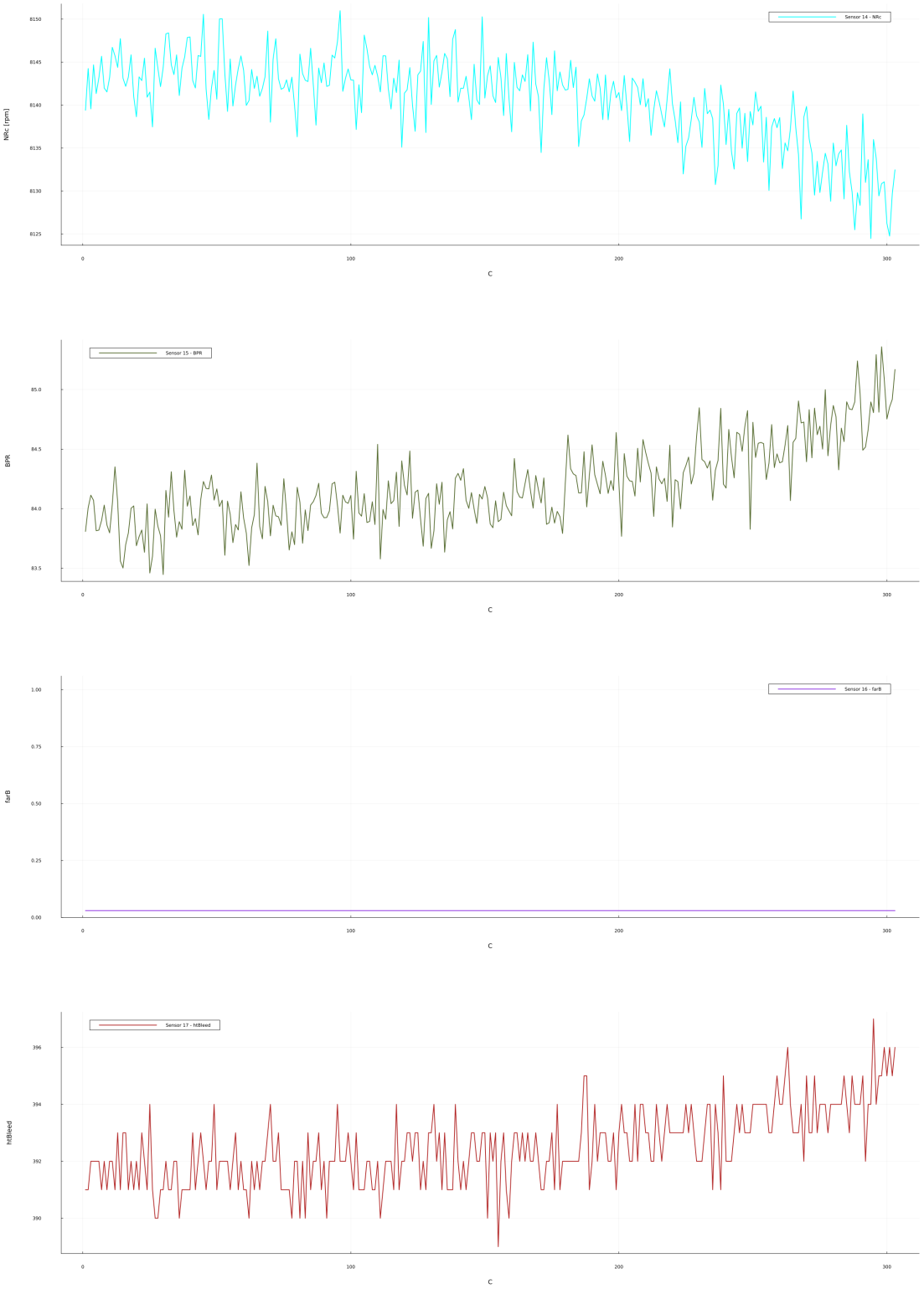


Figure 3.7: CMAPSS dataset Test 1, Unit 49: NRc, BPR, farB, htBleed

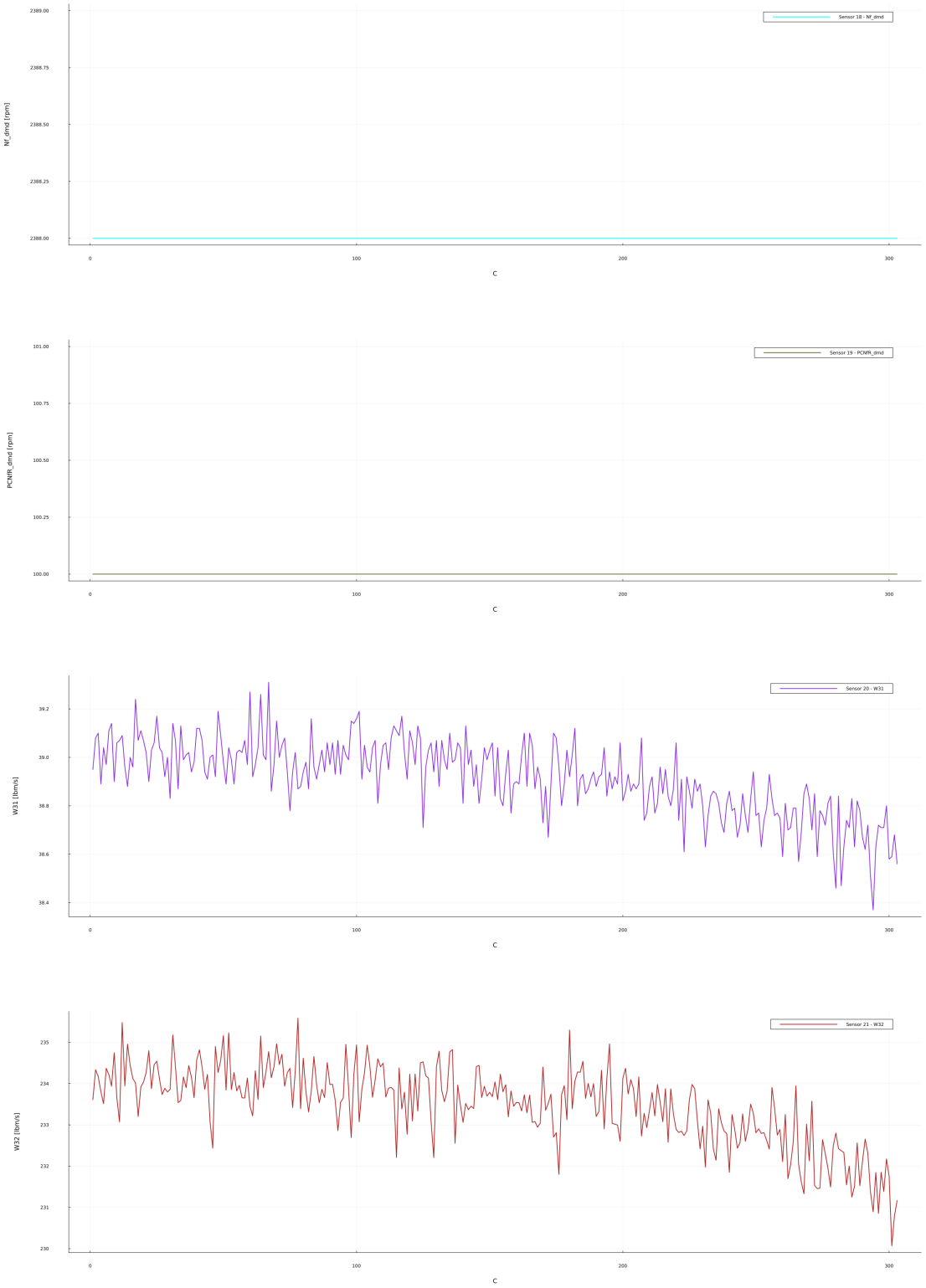


Figure 3.8: CMAPSS dataset Test 1, Unit 49: Nf_dmd, PCNfR_dmd, W31, W32

3.3 Resort facility power consumption (PC dataset)

The third dataset can be considered unrelated to the study at hand, but serves the purpose of studying power fluctuations in a complicated system. The time series in Figure 3.11 represents the power consumption of a large resort facility in the Punta Cana (PC) region, Dominican Republic, at 15 minute intervals. This time series is interesting as it contains data from January 2021 to April 2023, making it excellent for predictions of the power consumption of the following two months.

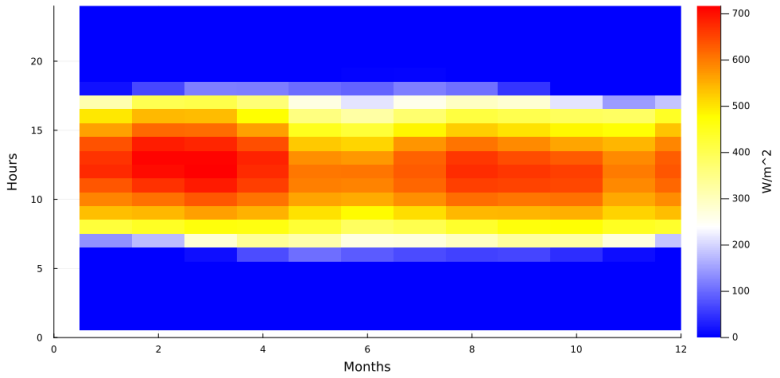


Figure 3.9: Direct solar radiation in Punta Cana (Dominican Republic) during 2021 [44]

The main challenge is to predict the power consumption of the facility, in megawatts, solely from the consumtpion at the previous time step. There were attempts to obtain the occupancy of the facility during the same time period, as it is directly related to the demanded power, but were unsuccessful. With higher occupancy, more power is required for the facility, specially from equipment such as lights and air conditioning units. There will be a fixed amount of power required at any given time, but fluctuations are mostly due to the people inside interacting with any of the equipment that can draw power from the grid. What can be obtained easily is the annual radiation data in the Punta Cana region [44], which is an input that can help the reservoir determine at what times of the day and when during the year the power requirements are higher due to an increased demand of air conditioners. Figure 3.9 is the heatmap for solar radiation in W/m^2 that was employed to generate an input signal that provides the reservoir the radiation at that moment.

An important characteristic of the time series in Figure 3.11 is the noise. Such wild fluctuations may affect the capacity of the reservoir to predict reliably the power consumption, to measure the effect of noise several comparisons will be carried out by filtering the signal with a method called Central Moving Average. The effect on the predictions on applying heavy filtering, that can be seen in the Figures will be discussed so as to determine whether noise is beneficial to the internal states of the reservoir as Jaeger [27] and Lukoševičius [29] claim.

	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β
Lower bound	100	0.01	0.01	0.01	0.01	0.001	0.01	$1{\cdot}10^{-8}$
Upper bound	5000	0.9	100	100	100	0.3	0.9	100

Table 3.5: Upper and lower bounds for the PC dataset hyperparameters

The hyperparameters for the PC dataset are practically the same as for the CMAPSS except for a few details. In the case of the size of the reservoir, N, the total length of the time series is around 80000 time steps. Pairing the length with a complexity due to the change in occupancy and the high amount of noise in the signal implies the necessity for a bigger reservoir. The upper limit has been set at 5000 a reservoir size that may drag the ensemble into a lengthy optimization process.

As for the input matrix scaling factor, a_{Win}, maximum and minimum values remain unchanged from the ones used in previous datasets. The particularity is to what inputs they are applied, as to

assist the reservoir in the prediction of the power consumption three other signals are generated. In Equation 3.2 refers to how the new signals are generated, as they represent the difference between the actual value of power and what will be demanded in 1, 10 and a 100 time steps, Figure 3.10. Therefore the first of the scaling factors is, once again, for the unitary input, the second is for the power itself and the third is for the three δ signals.

$$\delta_1 = p(t+1) - p(t) \qquad \delta_{10} = p(t+10) - p(t) \qquad \delta_{100} = p(t+100) - p(t) \tag{3.2}$$

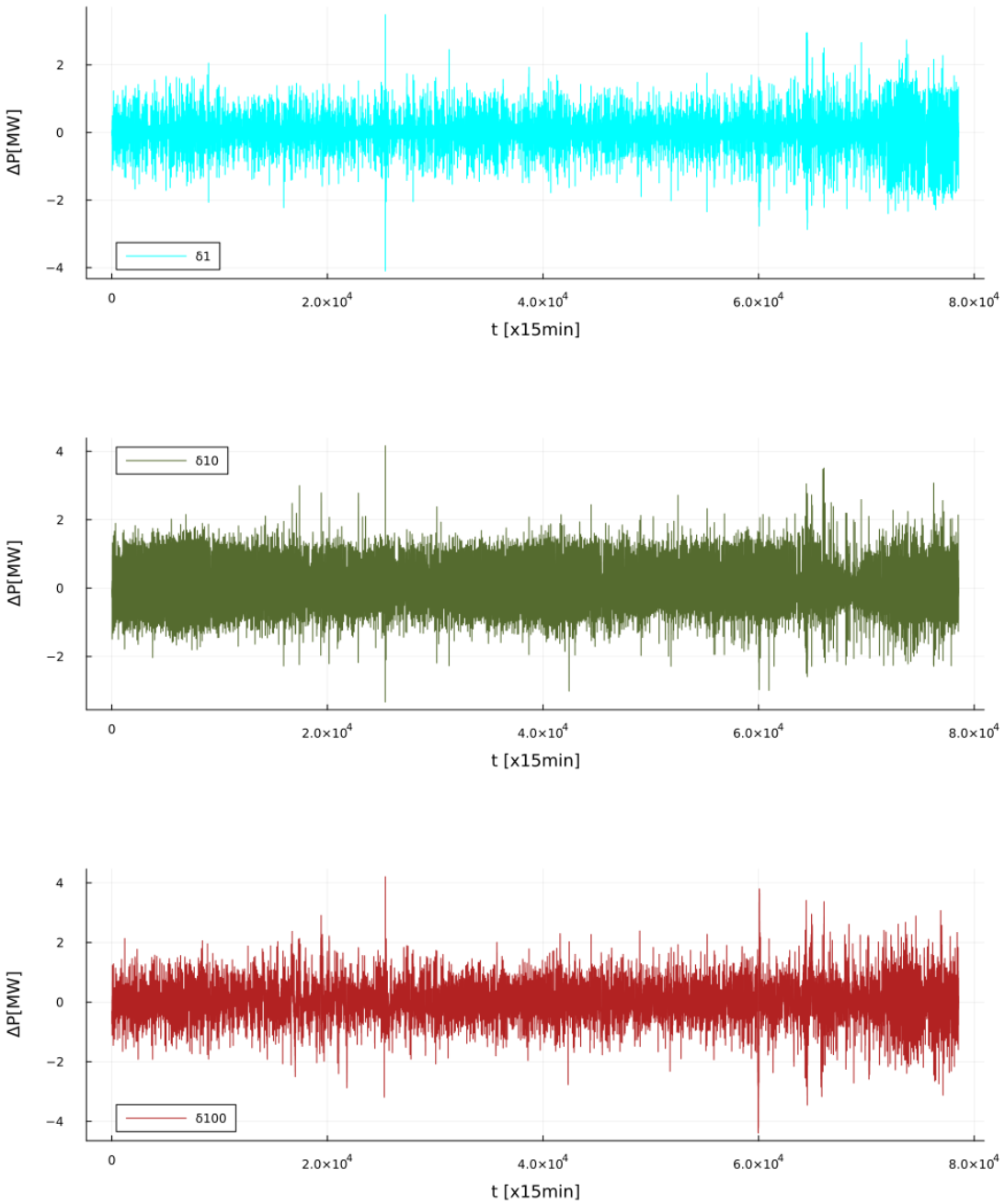


Figure 3.10: From top to bottom: differential values in 1, 10 and a 100 time steps for the PC dataset

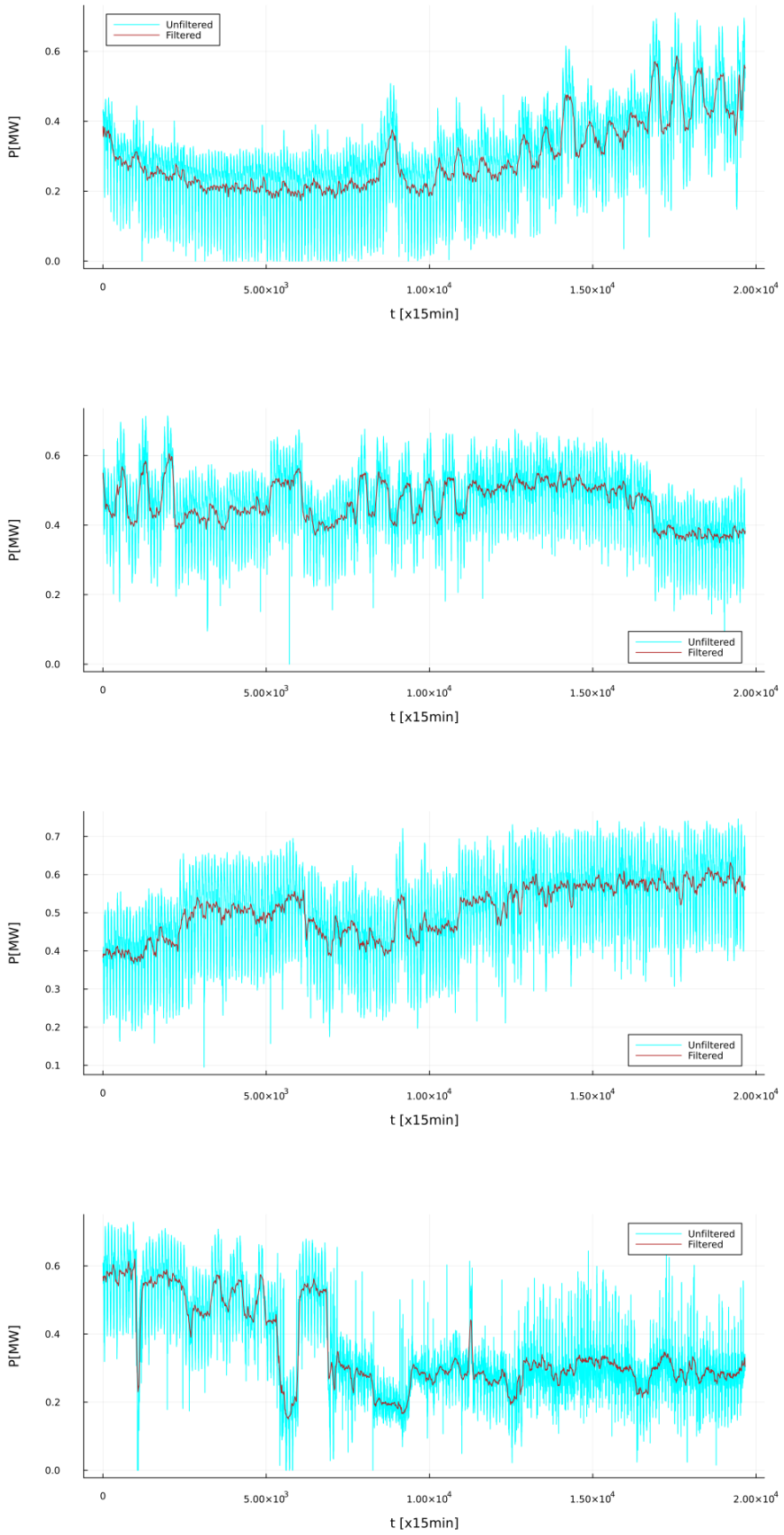


Figure 3.11: Resort facility power consumption from (top to bottom): January 2021 to July 2021, August 2021 to February 2022, March 2022 to September 2022 and October 2022 to April 2023. In cyan the unfiltered signal and in brown the filtered signal.

4 | Experimental results

To demonstrate the capability of the GWO-ESN ensemble the three previously presented datasets will be predicted: Mackey-Glass (MG), CMAPSS and Punta Cana (PC). The key strength of this method resides on its autonomous capacity to find the best reservoir topography for a given set of input data, tailoring its characteristics to generate the best possible prediction while still respecting the echo state property.

The reservoir has been programmed in the *Julia* language [45] with two variants: the first as a serial CPU process, while the second has been adapted to *CUDA* through the library *CUDA.jl* [46]. During the reservoir design process, the architecture was first fixed as a procedural calculation and then adapted for GPU parallel processing through the *CUDA.jl* library, more specifically the *CuArrays.jl* sublibrary [47]. The results shown have all been obtained through the *CUDA* reservoir to speed up the training procedure.

Several runs have been set up for each of the datasets, as the initial hyperparameters are randomly generated, to minimize the effect of chance on finding the optimum. An analysis on the prediction, not only the RMSE value for the test length, but the qualitative aspect of the predicted signal is evaluated. The internal states of the reservoir, at certain time steps, will also be discussed so as to evaluate what may be the fine tuning required to improve the predictions.

4.1 Mackey-Glass dataset

For the Mackey-Glasss dataset just two test runs have been carried out to check what results are generated through the ensemble. Both of them have been executed in the *CUDA.jl* variant of the reservoir, as it was the latest version and these tests were performed solely for benchmarking purposes.

Set	N	α	$a_{Win}(1)$	$a_{Win}(2)$	c	K_ρ	β	**nRMSE[-]**
1	347	0.34	0.39	9.95	0.23	0.53	0.93	0.25
2	1595	0.68	0.93	8.01	0.27	0.65	0.91	0.33

Table 4.1: Sets of hyperparameters for two different runs of the ESN-GWO system along with the RMSE result

As seen in Table 4.1, there are some key differences that contribute to the performance difference of the reservoirs. The size difference is massive, as is the leak rate and the regularization coefficient. When looking at the nRMSE the smaller reservoir seems to outperform the second set, not only in overall error but also on its capacity to assimilate the dynamics of the input signal.

The first of the predictions, Figure 4.1, is characterized by a small reservoir size, N, and leak rate, α. This characteristics leads to a prediction with cheap computational cost and a better short-term memory, inducing slow dynamics in the internal states. The input scaling factors, a_{Win}, are an order of magnitude apart with the one for the unitary input being close to twenty-five times as smaller as the other. This difference forces the reservoir to give more importance to the input

rather than the unitary value, modifying the way the activation function fires up. The fact that both values are close in magnitude to unity will prevent the internal states from saturating, a phenomenon present on several predictions from the other two datasets.

Another characteristic is the spectral radius scaling factor, K_ρ, which contributes to a longer memory of the previous inputs [29] along with the leak factor. Both hyperparameters are fundamental in remembering how the input signal evolves, and as can be seen in Figure 4.1, the fine variations on the chaotic attractor are followed closely during the first 500 seconds as well as the range between 1000 and 2000 seconds.

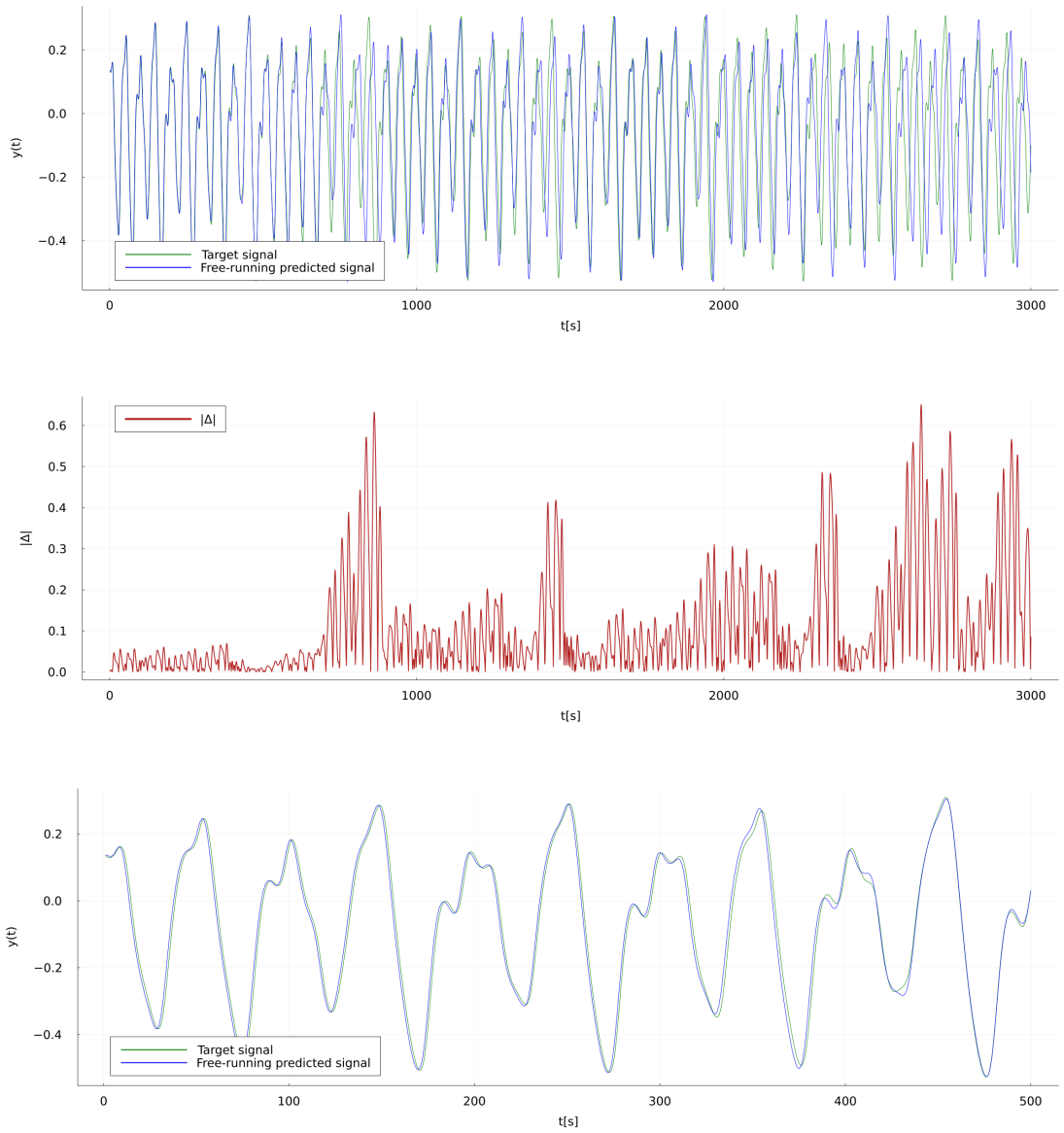


Figure 4.1: Mackey-Glass time series along with the predicted signal (above), the $|\Delta|$ variation in time (middle) and the predicted signal during the first 500 time steps (below) for the reservoir with the first set of hyperparameters

The predicted signal, Figure 4.1, represents accurately the periodic nature of the original with key differences when the Mackey-Glass equation results in local maxima o slight variations in shape, such as the characteristic saddle like peaks, once the first 500 seconds have passed. The prediction is not capable of mirroring these behaviours, and stays contained to a periodic function which coincides with the original but at a slight phaseout. If only the initial range is considered, Figure 4.1 below, the prediction clearly matches the original with an nRMSE value of 0.018. Comparing

the whole testing phase to these first 500 seconds shows clearly that at the beginning the generated dynamics are exactly as in the original signal and later on it starts to diverge, establishing a prediction window where the output of the reservoir is extremely accurate.

The absolute difference value, $|\Delta|$, stays low and mostly under 0.3 except with the appearance of maxima of up to 0.6. This matches the differences observed in Figure 4.1, as the $|\Delta|$ value as time progresses adopts a periodic like behaviour when the prediction gets out of phase with the objective signal, fluctuating between ranges with virtually a perfect match with the original and others where the phaseout causes sudden increases of the absolute difference. In general terms, this specific reservoir performs weel for the Mackey-Glass dataset at the first 500 seconds, fully capturing the required characteristics of the original signal to provide a valid prediction at the initial stages of the prediction.

The second prediction, Figures 4.2, has a much bigger reservoir size and a higher leak rate. Therefore consuming more resources than the first with internal states that do not hold as well the memory of previous inputs. The difference between both input scaling factors is similar to the first set, as an order of magnitude separates both. This contributes to the activation function to take into more consideration the actual varying input rather than the reference unitary value, which will be reflected on the internal states as will be seen later on.

With a spectral radius scaling factor higher than the first case, along with the higher leak rate, the short-term memory of this reservoir is smaller and will not resonate the previous characteristics in its internal states as much. The regularization coefficient is also the same order of magnitude as the first reservoir, which guarantees both output layers contain no high values and therefore avoid overfitting [29].

Figure 4.2 shows a prediction that during the first 100 seconds matches the characteristics of the original, especially the saddle like peaks, and as the test phase progresses the periodicity stops matching. During the first 100 seconds both signals are matched, except the phaseout of the prediction increases steadily as the time passes with an nRMSE of 0.14 for the first 500 seconds. While not all details are captured with the generated output, most of the desirable characteristics are predicted throughout the entire run. Furthermore, the prediction does not fall in a fixed amplitude behaviour and varies according to the original pattern, albeit not reaching the required values to completely mimic it at some specific locations. As the simulation progresses the prediction gets out of phase with the original, which is what contributes to the higher nRMSE.

The absolute difference value, $|\Delta|$, behaves in a periodic manner as did the first reservoir. The objective and the prediction are closely matched during certain ranges in which both signals fall on the same beat but as time goes on the error increases due to the out of phase behaviour of the second. The second set's higher nRMSE is directly tied to its earlier initiation of the phaseout, as its reduced memory capacity is not capable of capturing the dynamics of the original signal as successfully as the first set did.

According to Lukoševičius [29], the best way to understand how the reservoir is behaving is to represent the internal states during training, Figures 4.3 and 4.5. What can be observed at first glance is both reservoirs are unsaturated during training, with the exception of the $t = 1000s$ internal states of both sets. Saturation happens when most of the neurons within the reservoir have their activation functions generate an output close to the maximum or minimum, with a *tanh* function this is either -1 or 1, making the training loose scope of the finer details of the objective function. Having the neurons within the reservoir reach those values is not detrimental when it happens at certain moments in time when the input signal has an uncharacteristic high or low value, not persistently throughout the whole training.

The first reservoir, Figure 4.3 and the histogram at Figure 4.4, is unsaturated at the latter five depicted time steps with most of the neurons reaching activation function values consistent with a

normal distribution. The consequences will be the generic function generated as a prediction that can be seen in Figure 4.1, as it is able to distinguish its characteristic features and replicate the dynamics during the prediction. If it were to be saturated, any input going into the reservoir no matter its value would result in the fringe values of the activation function. This behaviour is only present at the initial stages of training, which is a sign of a transient effect caused by a certain input rather than an erroneous topography.

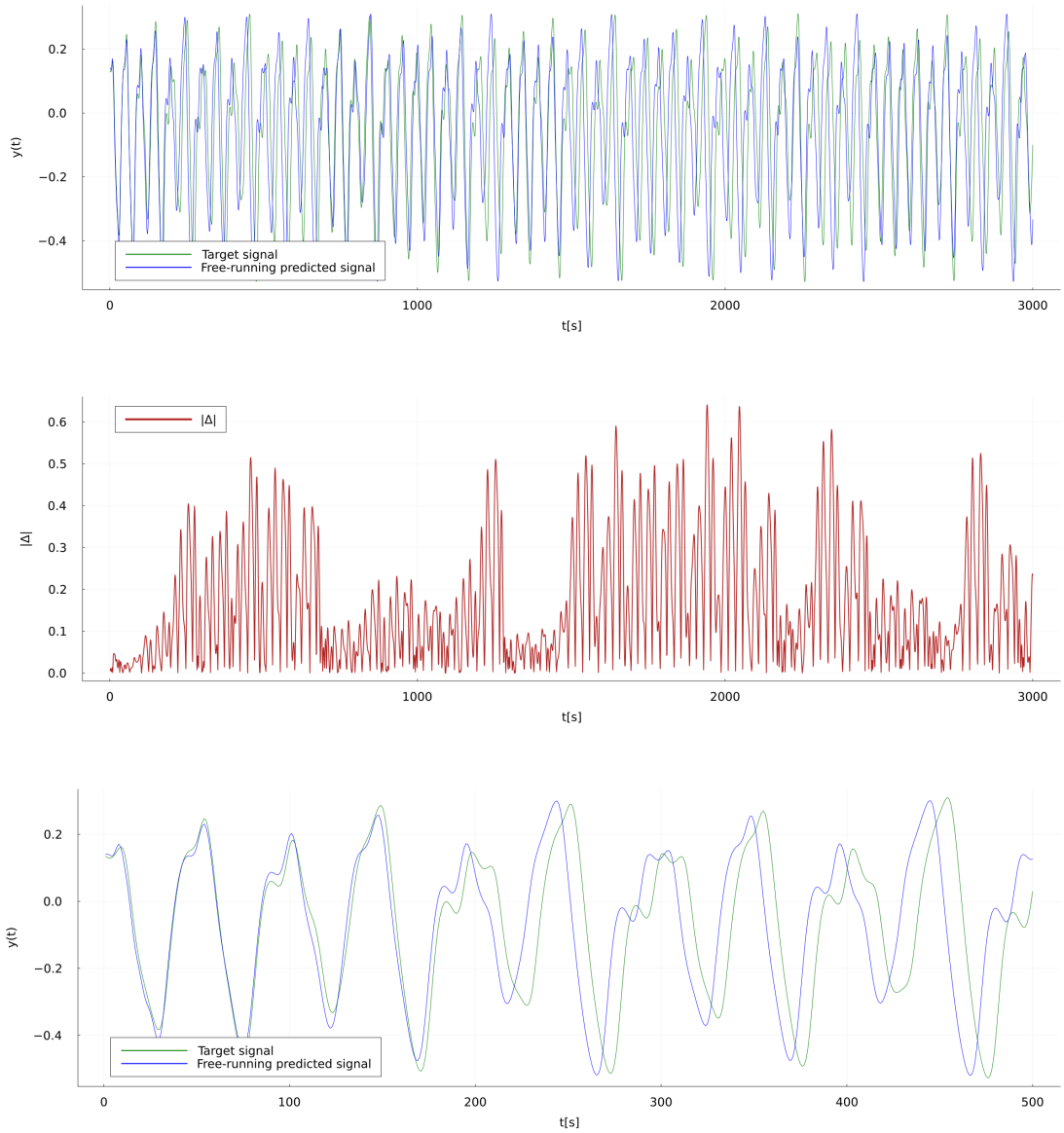


Figure 4.2: Mackey-Glass time series along with the predicted signal (above), the $|\Delta|$ variation in time (middle) and the predicted signal during the first 500 time steps (below) for the reservoir with the second set of hyperparameters

On analysis for the two other datasets, the size of the reservoir, if too small, leads to a lacking capacity to gather the dynamics of the input signal with not enough neurons present to capture the individual characteristics and ultimately flooding them and driving the reservoir to saturation. This phenomena is not present on the first hyperparameter set, as even though the reservoir is small the input scaling factors are scaled appropriately and are a more defining factor when determining if the reservoir is going to be driven to saturation.

As for the second reservoir, Figure 4.5 and the histogram at Figure 4.6, the internal states are

not saturated and vary from one time step to another. This allows for more detail to be captured and allow the training to better discern how it the input signal is evolving. For instance, the internal states at $t = 1000$ present saturation but is due to a transient effect rather than a common occurrence during training. On that time step the absolute value of the input, along with the memory of the previous, caused the reservoir to react in that distinct manner teaching the output layer that a specific event is underway. If the reservoir is saturated, when something remarkable in the signal is happening against a typical behaviour it soon becomes indistinguishable and it will not be replicated in the prediction.

The first set of hyperparameters generates a reservoir with internal states more aptly suited to the Mackey-Glass dataset than the second. At certain steps during training the internal states of the first set concentrate on a narrow band around 0, but still manage to successfully capture the original signal. The second set internal states are also appropriate, with a normal distribution all along training that contributes to an appropriate prediction that unfortunately phases out from the original too early during the testing phase. Avoiding the saturation of the internal states is fundamental to generate predictions that contain the desired characteristics of a particular signal.

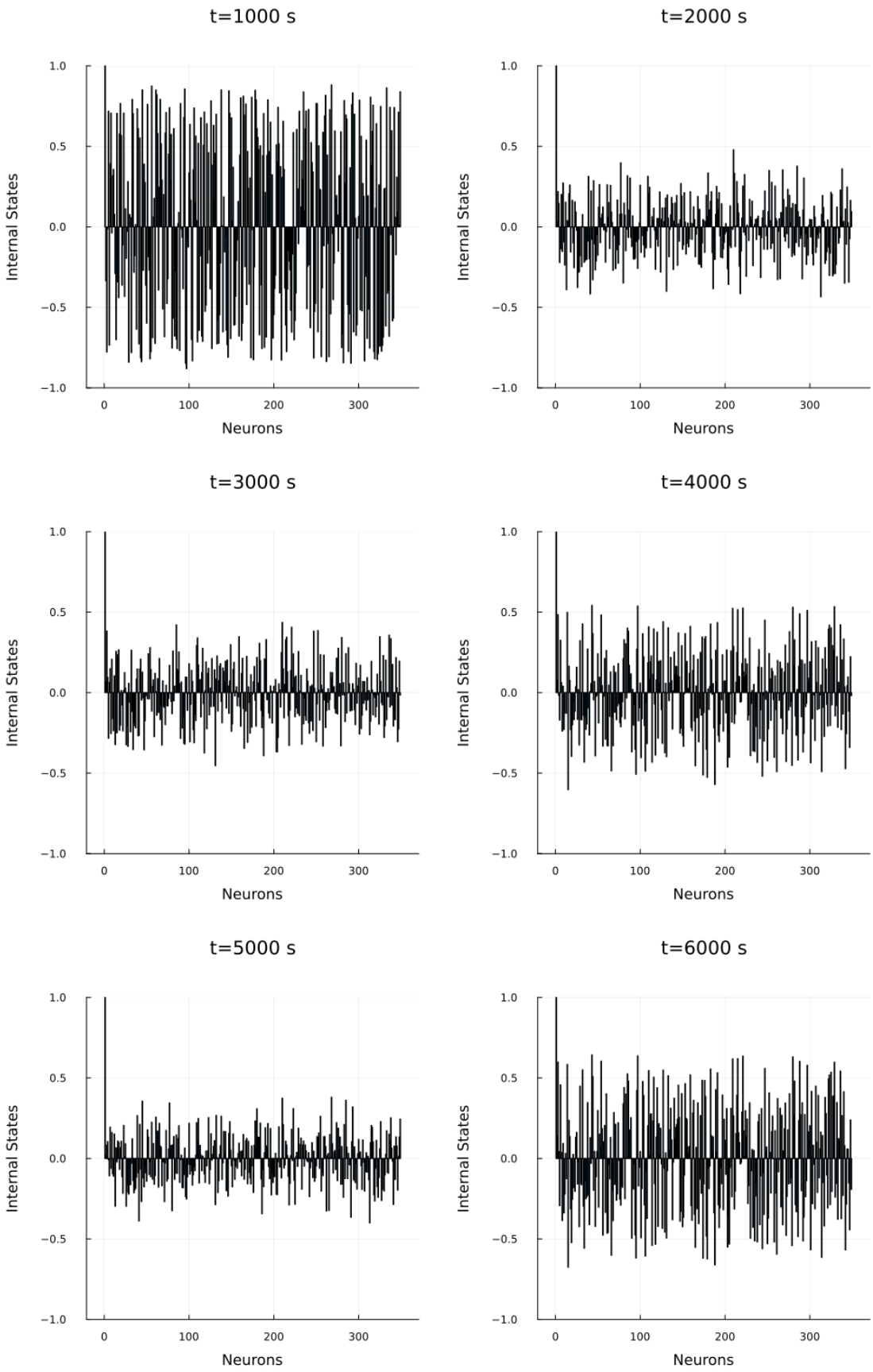


Figure 4.3: Reservoir internal states at various moments during training for the Mackey-Glass time series for the first set of hyperparameters

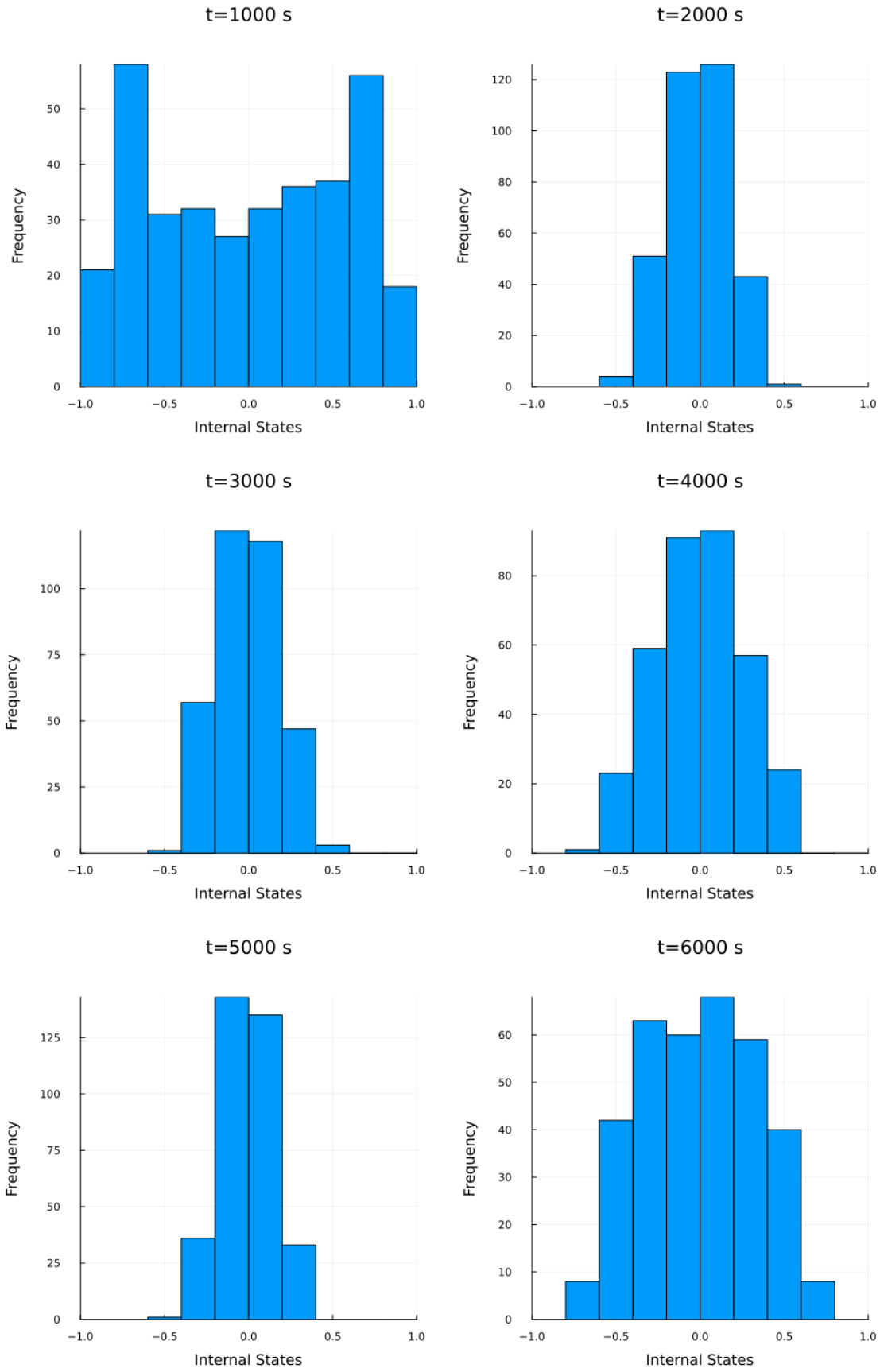


Figure 4.4: Histogram of reservoir internal states at various moments during training for the Mackey-Glass time series for the first set of hyperparameters

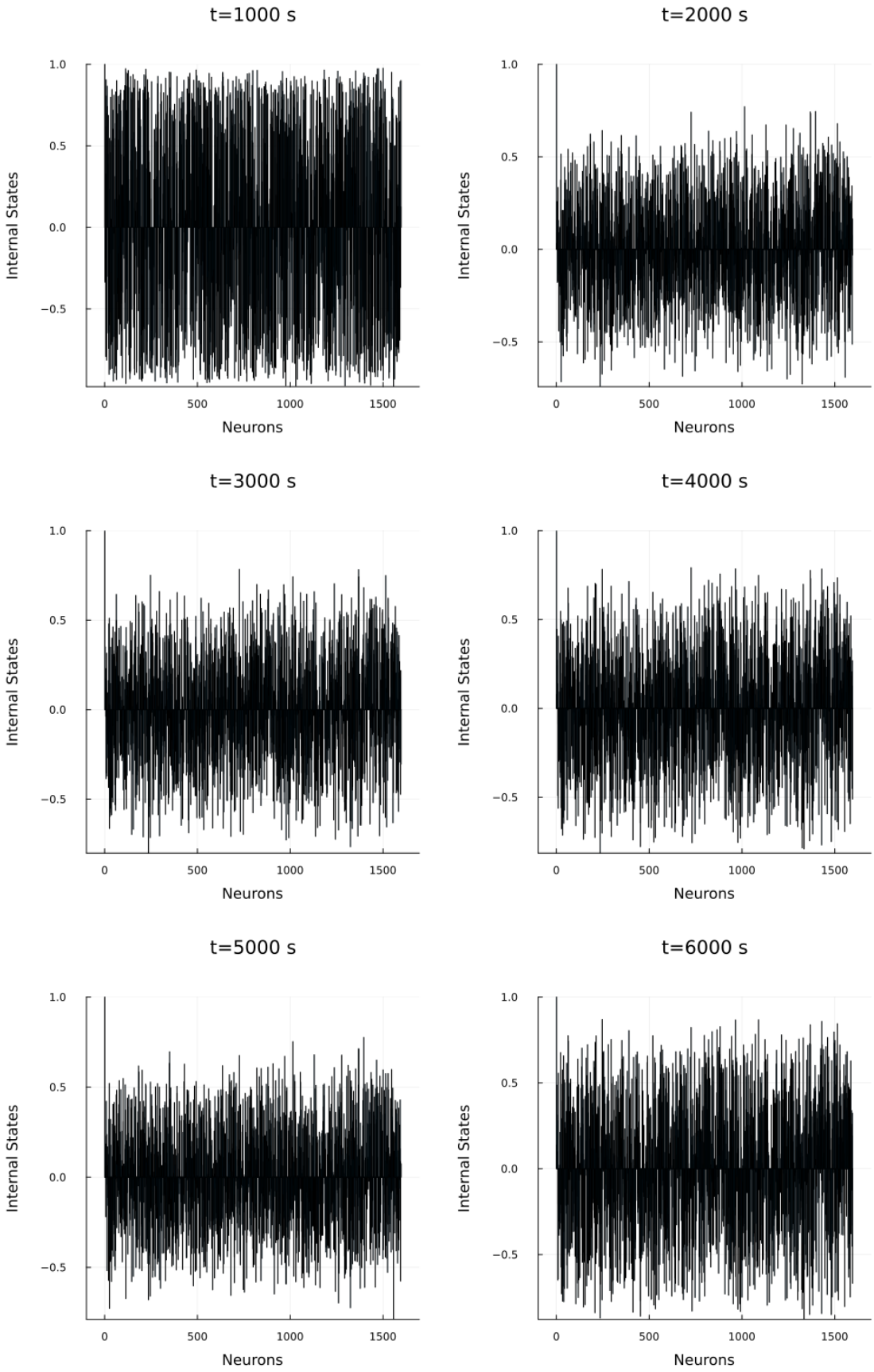


Figure 4.5: Reservoir internal states at various moments during training for the Mackey-Glass time series for the second set of hyperparameters

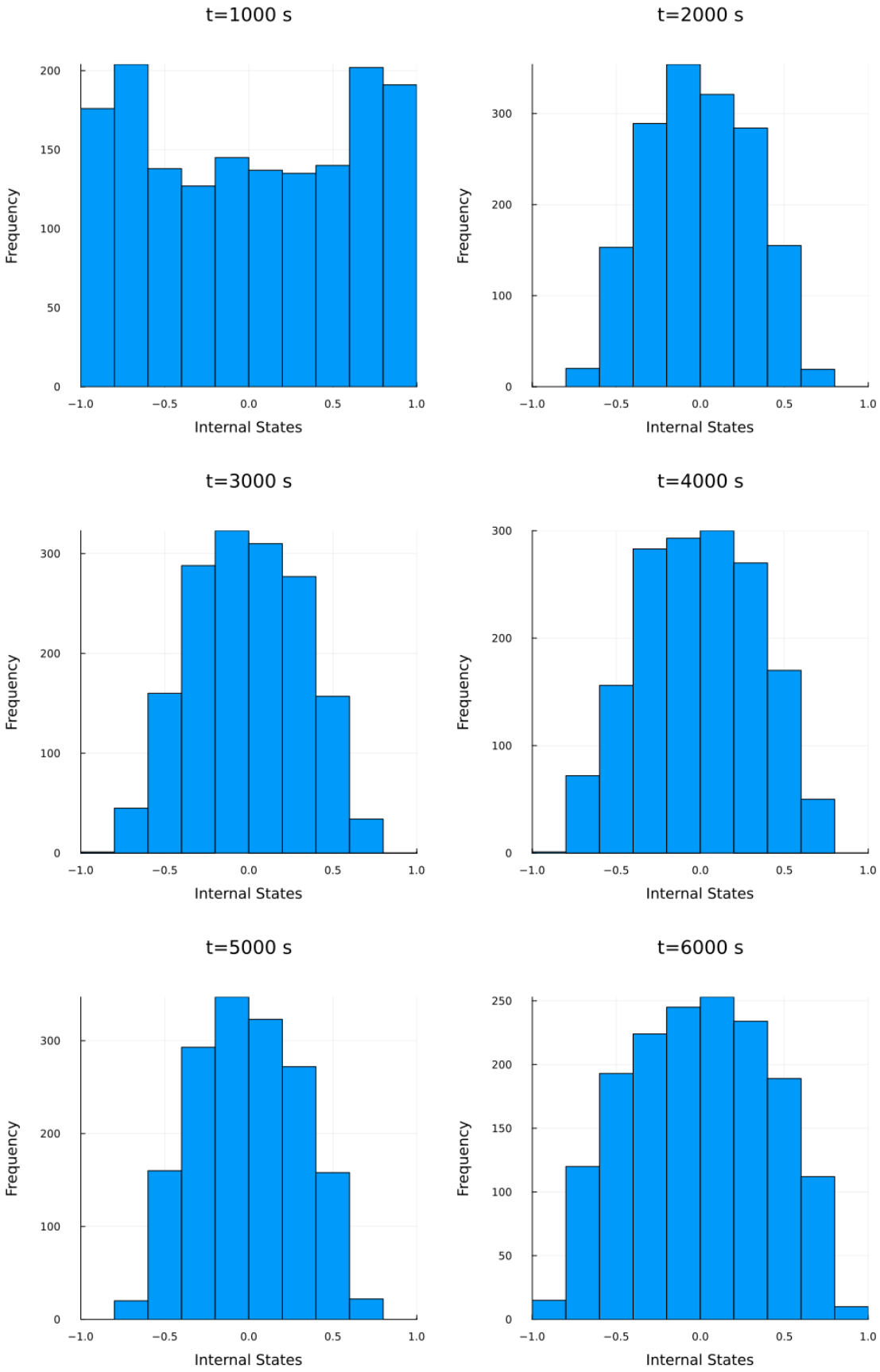


Figure 4.6: Histogram of reservoir internal states at various moments during training for the Mackey-Glass time series for the second set of hyperparameters

4.2 CMAPSS dataset

The CMAPSS dataset grants the option to analyse the impact of having a limited amount of data available for the prediction, only about three hundred discrete values for each signal, and how to minimize it. Table 4.2 shows the reservoir parameters set obtained through five different runs of the ensemble, along with the average nRMSE value of the three operational settings and twenty-one signals. All five have an error value close to each other as the reduced amount of training data affects the capacity of the reservoir to accurately classify the behaviour of the turbofan engine and attempt a sound prediction. Nevertheless, some of the signals are more aptly predicted than others and can provide insight on the turbofan engine.

Trial	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β	**nRMSE**
1	51	0.43	75.76	73.00	71.65	0.08	0.61	0.92	0.36
2	481	0.39	89.33	91.69	82.83	0.13	0.59	0.39	0.37
3	489	0.36	0.86	7.93	7.79	0.23	0.73	0.89	0.39
4	105	0.40	0.49	8.39	1.89	0.25	0.51	0.89	0.36
5	166	0.57	0.91	7.96	9.30	0.29	0.74	0.42	0.42

Table 4.2: Different sets of hyperparameters for the reservoir on the CMAPSS data and the obtained nRMSE after five different runs of the ensemble.

The biggest and most defining difference between each of the four configurations is the size, as two of them are around 500 neurons, one is 50 neurons and the other two are around a 100. The variation in computational cost between them is an order of magnitude, as with a fixed number of connections between neurons the cost increases linearly, and as can be seen increased size does not guarantee better error results. Another key difference are the input scaling factors, which will play a role along with the size in either preventing or promoting the saturation of the internal states.

On Table 4.3 each nRMSE value for the prediction of the twenty-four signals by each of the hyperparmeter sets is depicted. Certain predictions, due to the stationary nature of the original signals, have yielded a null error value. These have to be discarded from being taken into consideration, Operational Setting 3 and Signals 1, 5, 10, 16, 17, 18 and 19, as the focus lies on those whose fluctuations yield information on the health status of the turbofan engine.

The biggest challenge for this data is seeking the optimum of twenty-four signals simultaneously, which is why comparing each of the nRMSE values to each other is also included. Once again, the internal states during training are a valuable metric on the performance of the reservoir and will be of interest for result comparison. The cases will be analysed separately, only showing the fourth hyperparameter set along a validation run. An analysis of the four remaining sets is included on Annex A.1.

The data and results shown of the CMAPSS dataset on this thesis has been published in the proceeding for the *10th International Conference on Time Series and Forecasting* [39], which can also be found on Annex B. The following analysis dives deeper on the prediction characteristics and the reservoir saturation phenomena for all five of the studied datasets.

4.2.1 Fourth set of hyperparameters for the CMAPSS dataset

The second-to-last set of hyperparameters is the one that presents the lowest average nRMSE, a position which shares with the saturated first reservoir, Annex A.1.1, with a moderate amount of neurons available and an average short-term memory capacity. The size is of 105 neurons, which is nearly five times less than the second best configuration, Annex A.1.3, and provides a nearly negligible reduction to the nRMSE. There is a definitive advantage on using such a lightweight

Dataset ID	Set 1	Set 2	Set 3	Set 4	Set 5	**Mean**	Validation
Operational Setting 1	0.35	0.33	0.44	0.39	0.42	0.39	0.38
Operational Setting 2	0.48	0.36	0.31	0.34	0.36	0.37	0.45
Operational Setting 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 1	0.50	0.32	0.43	0.50	0.56	0.46	0.45
Sensor 2	0.40	0.48	0.47	0.40	0.51	0.45	0.33
Sensor 3	0.36	0.46	0.36	0.38	0.43	0.40	0.37
Sensor 4	0.39	0.48	0.52	0.34	0.47	0.44	0.38
Sensor 5	0.50	0.32	0.43	0.50	0.56	0.46	0.45
Sensor 6	0.36	0.37	0.35	0.38	0.42	0.38	0.45
Sensor 7	0.43	0.43	0.53	0.36	0.52	0.45	0.34
Sensor 8	0.43	0.47	0.52	0.36	0.54	0.46	0.40
Sensor 9	0.41	0.47	0.39	0.38	0.41	0.41	0.35
Sensor 10	0.50	0.32	0.43	0.50	0.56	0.46	0.45
Sensor 11	0.44	0.44	0.47	0.38	0.52	0.45	0.37
Sensor 12	0.39	0.51	0.52	0.42	0.53	0.48	0.41
Sensor 13	0.38	0.49	0.53	0.46	0.48	0.47	0.38
Sensor 14	0.36	0.44	0.45	0.42	0.47	0.43	0.33
Sensor 15	0.34	0.48	0.44	0.43	0.46	0.43	0.46
Sensor 16	0.50	0.32	0.43	0.50	0.56	0.46	0.45
Sensor 17	0.40	0.43	0.37	0.36	0.50	0.41	0.46
Sensor 18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 19	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 20	0.30	0.47	0.48	0.38	0.42	0.41	0.32
Sensor 21	0.40	0.40	0.42	0.37	0.43	0.40	0.39

Table 4.3: nRMSE values for the predicted signal values of each of the sets and the validation run with data from Unit 93. OS 3 and Signals 1, 5, 10, 16, 17, 18 and 19 remain at a constant value throughout the length of the dataset.

reservoir, as the average nRMSE is even lower while costing less computationally. As with the first set of hyperparameters, Annex A.1.1, the leak rate is moderate and the spectral radius scaling factor is low, this combination leads to a somewhat reduced short-term memory with the reservoir not having as much of the previous inputs resonating within. The most important hyperparameters are the input scaling factors, a feature shared with the other three datasets, which assist the reservoir in giving more weight to certain inputs when going through the activation functions. This time the factors are within a reasonable order of magnitude, furthermore there is a clear difference between each one of them, so they are also useful for the reservoir and not just a mean to saturate its internal states. It is of particular interest the scaling factors difference between the operational states and the sensors, giving more than four times the importance to the first against the second, which will be reflected in the individual signal error values.

The individual signal nRMSE values, shown in Figure 4.7, change significantly with respect to the other sets of hyperparameters as a result on the new input scaling factors. For instance the Operational Settings have a lower nRMSE than most of the Sensors, a pattern also found in the other four reservoirs. Sensor 1, 5, 10 and 16 have the highest valued error at just 0.5 with most of the others being within the range between 0.3 and 0.5, therefore the average nRMSE for the predicted signals is lower than the other sets, the key difference this time being the Operational Settings and the lower maximums. Operational Setting 3 and Sensors 18 and 19 are once again at a null error value, a constant element throughout all the sets. The problematic signals are Sensors 1, 5, 10, 16 and 17 which the reservoir never gets to truly capture and four of them happen to be the highest error predictions. The average nRMSE for this set of hyperparameters is 0.36, the lowest of the ones under study, and confirms the necessity for a specific relation between the input scaling factors.

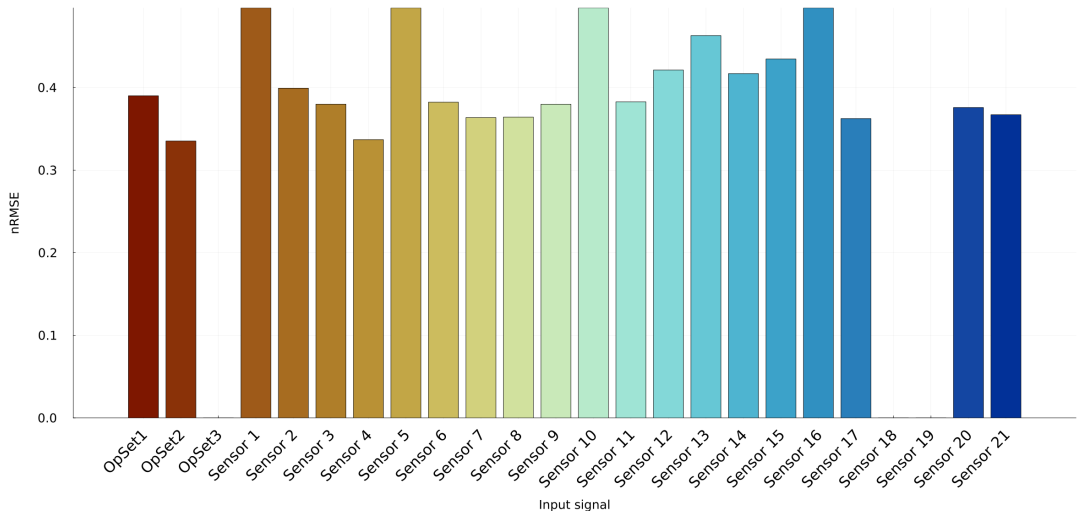


Figure 4.7: nRMSE value of each of the signals for the prediction generated by the reservoir with the fourth set of hyperparameters.

Analysing Operational Setting 1, in Figure 4.8, it is surprising to see the prediction is not accurately following the original. The nRMSE is 0.39 and even though there is no divergence between the two of them, there is only on matching drops or peaks anywhere in the testing phase as the prediction presents high variability. At cycle 75 it gets close to a rise and drop coincidence but apart from that there is nothing else, which can be easily confirmed by the absolute difference variation which is constantly moving around 1 with peaks of up to 6. Operational Setting 2, at Figure 4.9, has similar problems albeit with some characteristics of the target being close to matching the prediction: for instance, the steady signal at 30 cycles, the drop and rise at cycle 68 and the drop at cycle 75. The nRMSE is lower than the first: 0.34, and the generated signal follows the target at the initial cycles up to 18 and during the 40 to 60 cycles range with low variability once again.

Figure 4.10 shows Sensor 2 with a prediction that is once again not fully capturing the variations of the original. The $|\Delta|$ variation clearly depicts an steady increase in the distance between both signals, with the divergence clearly starting soon during the testing phase at 40 cycles. With an nRMSE of 0.4 it would seem there should be some matching peaks and drops such as the ones in other sets, but this prediction is just incapable of capturing the behaviour of the original with just two matching drops at cycles 22 and 36. Sensor 3, in Figure 4.11, was predicted with an nRMSE of 0.38 but during the initial cycles of the testing phase it does not follow the target accurately only coming close to the original at the predicted drop at cycle 19 or the rise at cycle 35 only converging at the original at the latest moments of the prediction. The $|\Delta|$ variation confirms this convergence as at the beginning the difference is steadily rising up to 6 at which point the prediction starts getting closer to the original and the absolute difference decreases.

Sensor 7 in Figure 4.12 presents a divergence between the prediction and the original at the end of the testing phase, from cycle 35 onward from where the variability of the prediction increases wildly. The $|\Delta|$ variation up until 35 cycles is really low, only ever reaching values up to 2.5, which in turn reflects in the prediction as it follows the trends of the original consistently. When the divergence point is reached the absolute difference variation increases up to 7 as the generated signal fails as it does not mimic the downward trend of the original. As for Signal 14, Figure 4.13, the prediction remains close to the original without quite matching its trends due to its variability. The $|\Delta|$ variation is the lowest in all sets as it is found mostly under 2 through the entire testing phase. The prediction does not fully match the drops and rises of the original, but keeps close reflecting the changes in trend with substantial variability while successfully following the downward motion.

The internal states, displayed in Figure 4.14 and the histogram at Figure 4.15, shows a reservoir on the verge of saturation at certain cycles but with enough separability to be valid for training. Early during training, cycles 20 and 40, the internal states are gathering at fringe values which are a clear indication on being close to saturation. At 60 cycles this trend is quickly reversed, achieving and internal states which resembles a normal distribution. After which at 80 cycles close to half of the neurons are at fringe values once again, which could be caused not by the input scaling factors but the size of the reservoir being small for the input or a transient effect as from 100 cycles on there is a shift in the separability of the inputs. Each one of them is clearly distinct as they cover clearly different ranges of the possible values of the activation function.

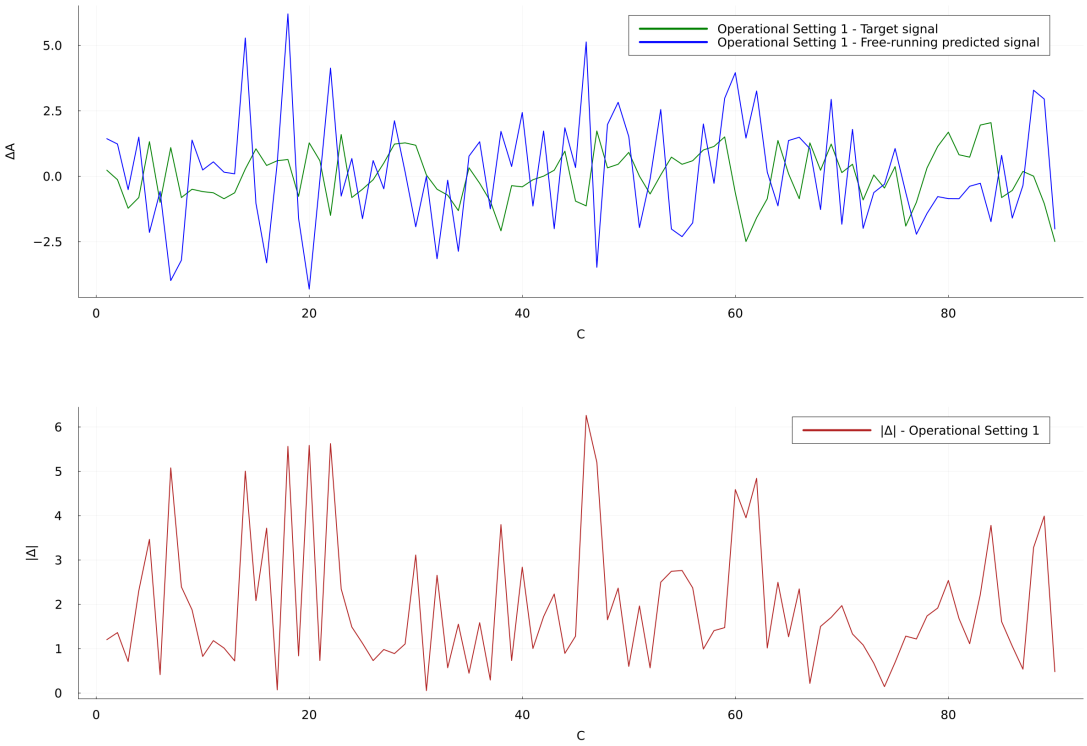


Figure 4.8: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

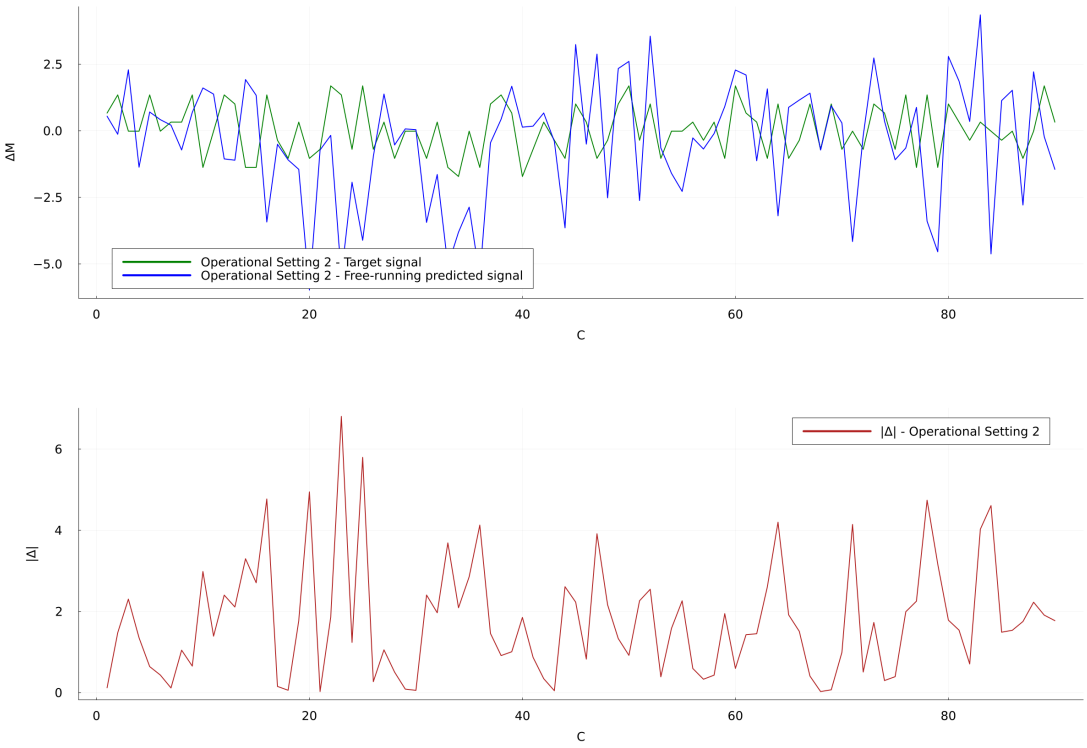


Figure 4.9: Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

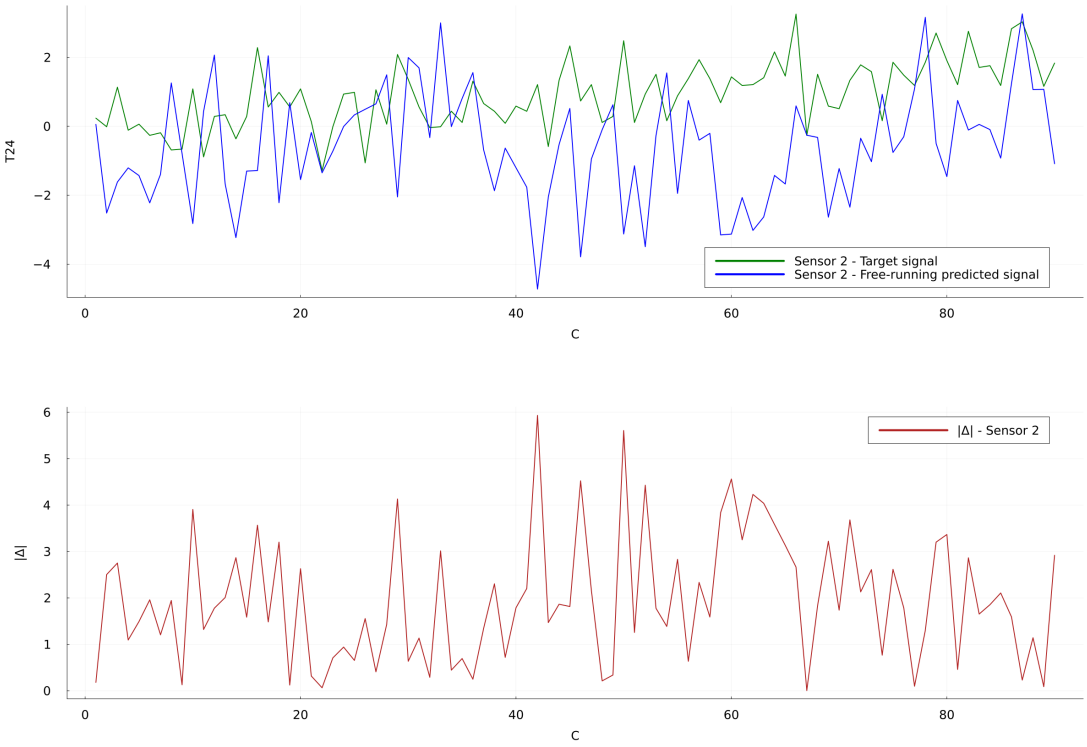


Figure 4.10: Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

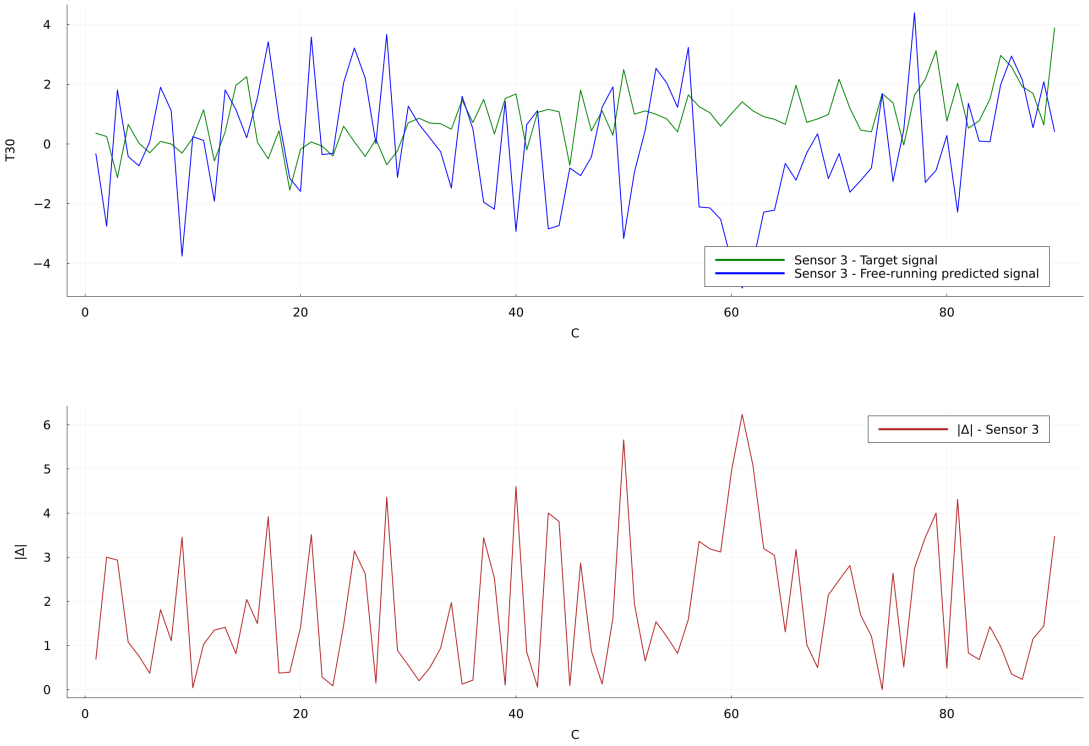


Figure 4.11: Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

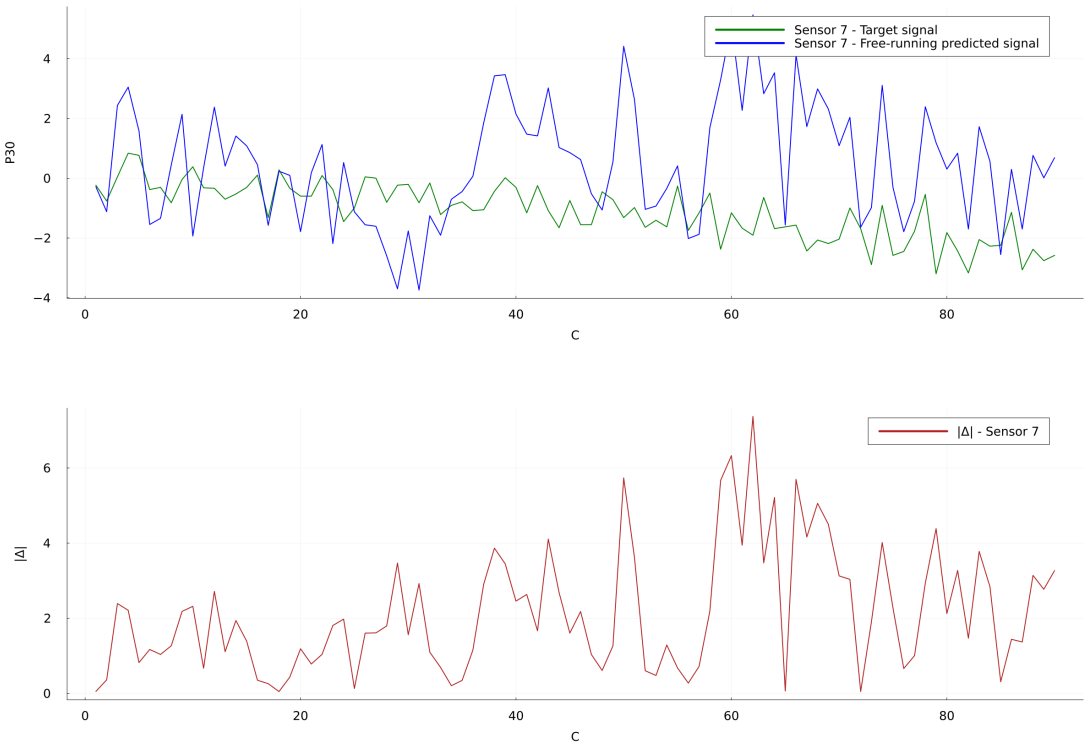


Figure 4.12: Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

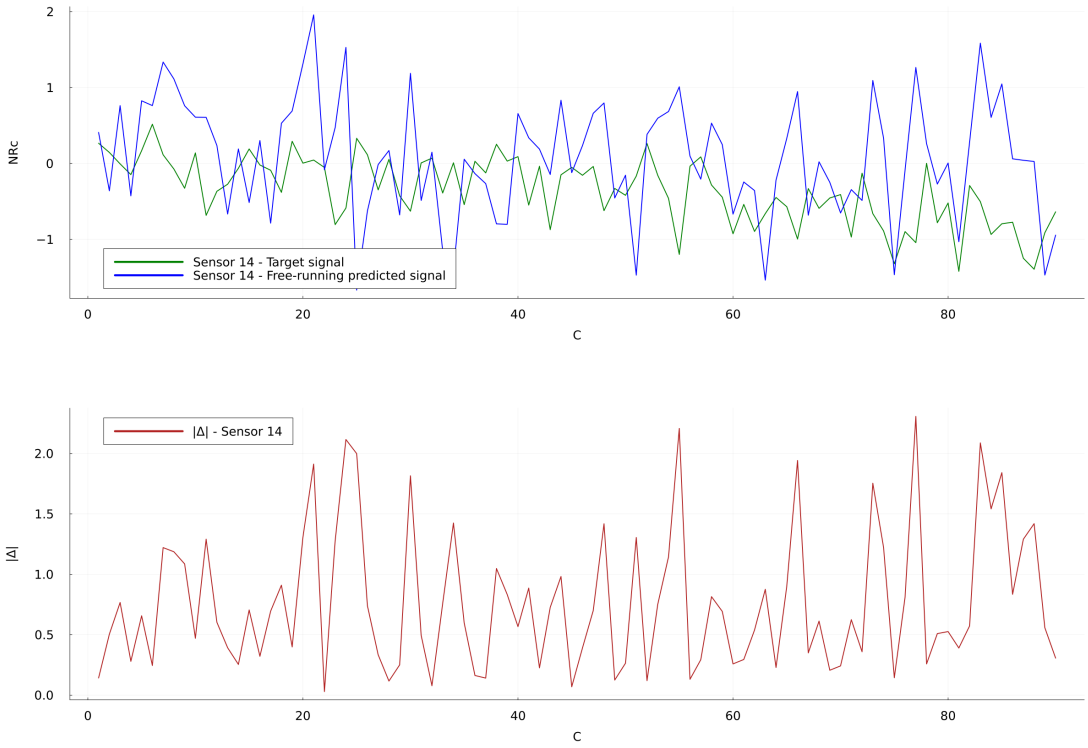


Figure 4.13: Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyperparameters

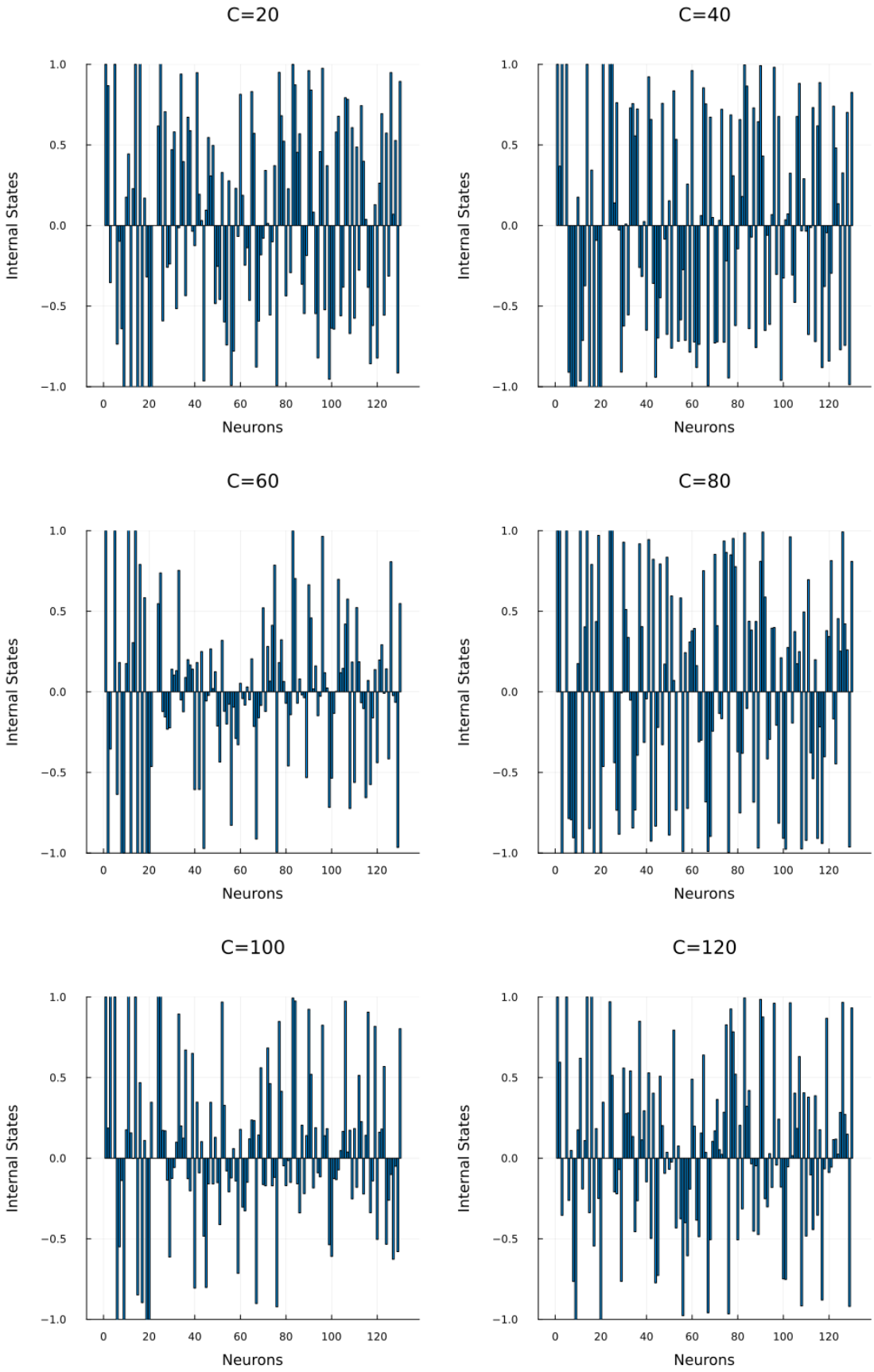


Figure 4.14: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters

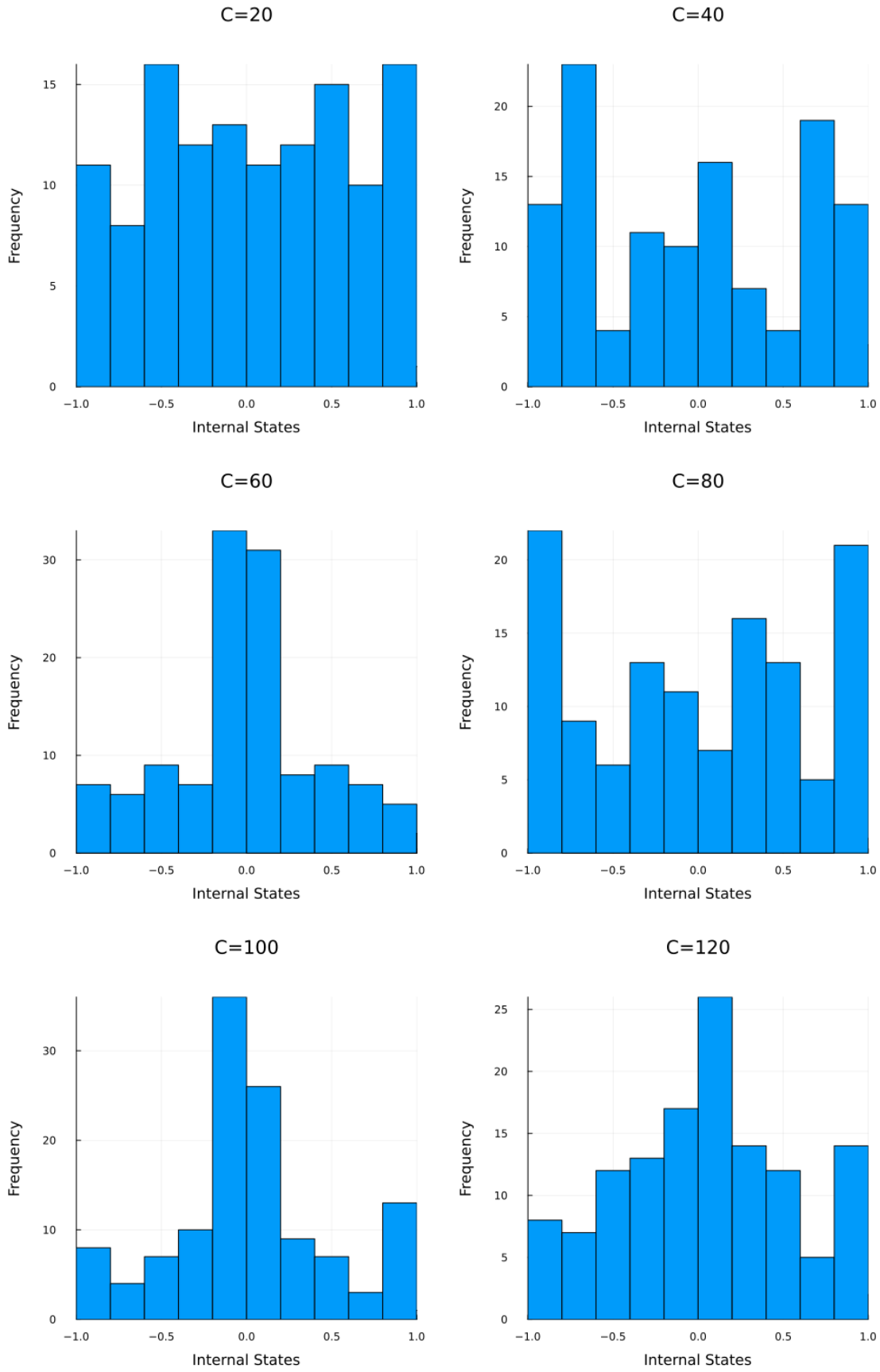


Figure 4.15: Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters

4.2.2 Validation using the fourth set of hyperparameters with Unit 93

The most evident showcase of the ensemble is finding the hyperparameters for a specific input signal and then applying them to a similar one without the need for optimization. Using the reservoir optimized for a specific input in another requires for the new one to be of a similar nature, involve the same internal dynamics and require the same number of inputs and outputs. This has been put in practice by using the reservoir from the fourth set of hyperparameters to the data from Unit 93, another subset of the CMAPSS dataset. As Figure 4.16 shows the nRMSE for this new set of signals is even better than for the original, with an average of 0.35. Furthermore, certain signal predictions perform better than with the original data, such as Sensor 2, 9, 14 and 20 as can be seen in Table 2.21. The rest of the signals are within reasonable error margins with respect to the other examples.

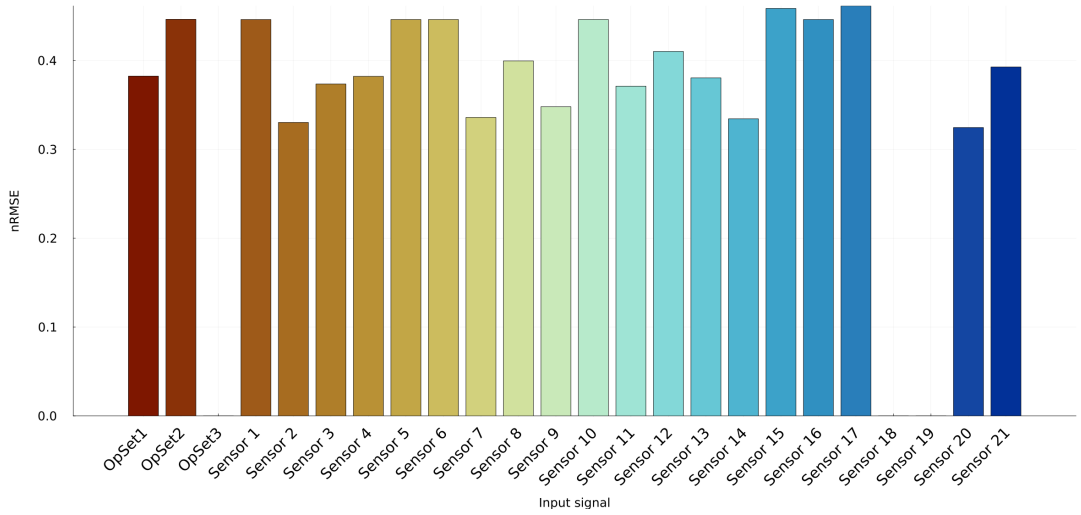


Figure 4.16: RMSE value of each of the signals for the prediction generated by the reservoir with the fourth set of hyperparameters with the data from Unit 93.

Figure 4.17 shows the prediction of Operational Setting 1, with an nRMSE of 0.38 it is a great example on how the hyperparameters for another set of signals can be applied successfully to a new one. The absolute difference , $|\Delta|$, stays under 2 for most of the testing phase except for the end, in which at 40 and 60 cycles the prediction overshoots and undershoots respectively from the target. For the most part the generated signal captures the trends of the original with no divergence, fluctuating around the target´s average and matching certain features such as the local maxima at 37 cycles and 43 cycles. A reservoir optimized for a specific engine can be retooled to another and serve to estimate trend changes in the Operational Settings or Sensors without having to optimize and retrain it.

The internal states of the reservoir during training, shown at Figure 4.18 and as a histogram at Figure 4.19, are less saturated for the data from Unit 93 than it was for Unit 49. The input scaling factors remain unchanged, so it may hint at the idea that the problem for this specific set of hyperparameters is the size of the reservoir, as this particular Unit only has data for 250 cycles while the original was more than 300 cycles long. Very few neurons at different stages of training actually present extreme activation function values, increasing the separability of the inputs even with such a reduced reservoir size. Therefore it is possible to optimize the reservoir for a specific signal and use its configuration parameters for predicting similar ones. It would be also interesting to optimize the ensemble on multiple Units and use the hyperparameters for all of them, having the same reservoir for a whole fleet of turbofan engines as their internal dynamics are based on the same design principles and only very precise details vary from one to another.

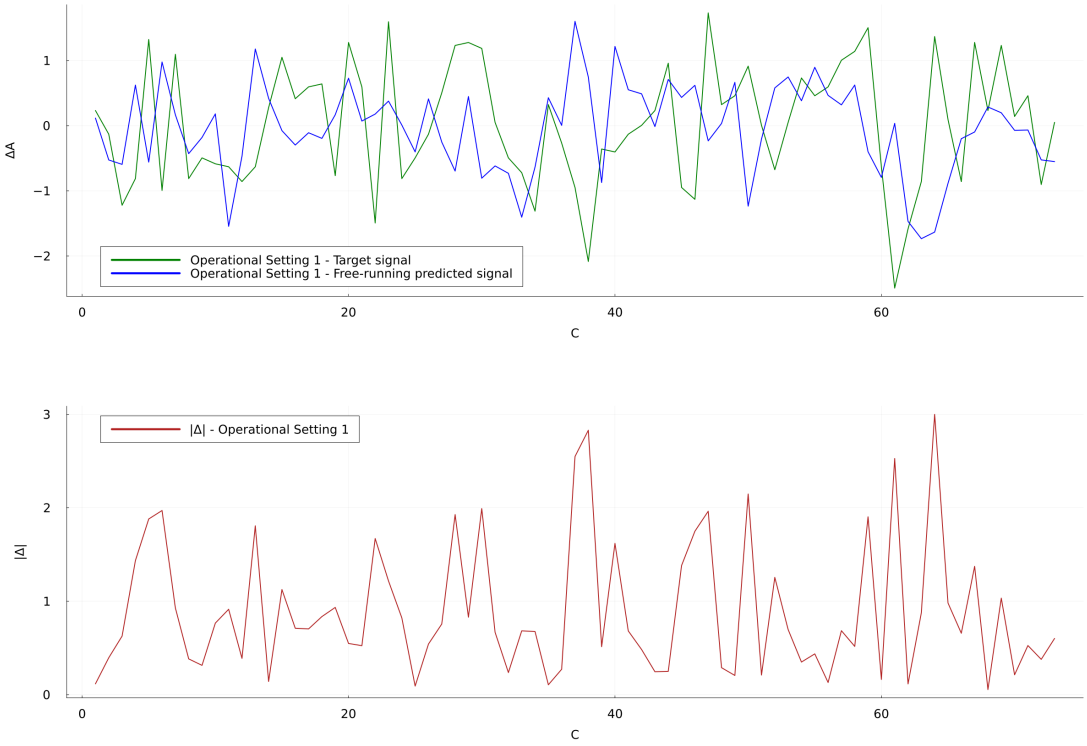


Figure 4.17: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fourth set of hyper-parameters with the data from Unit 93

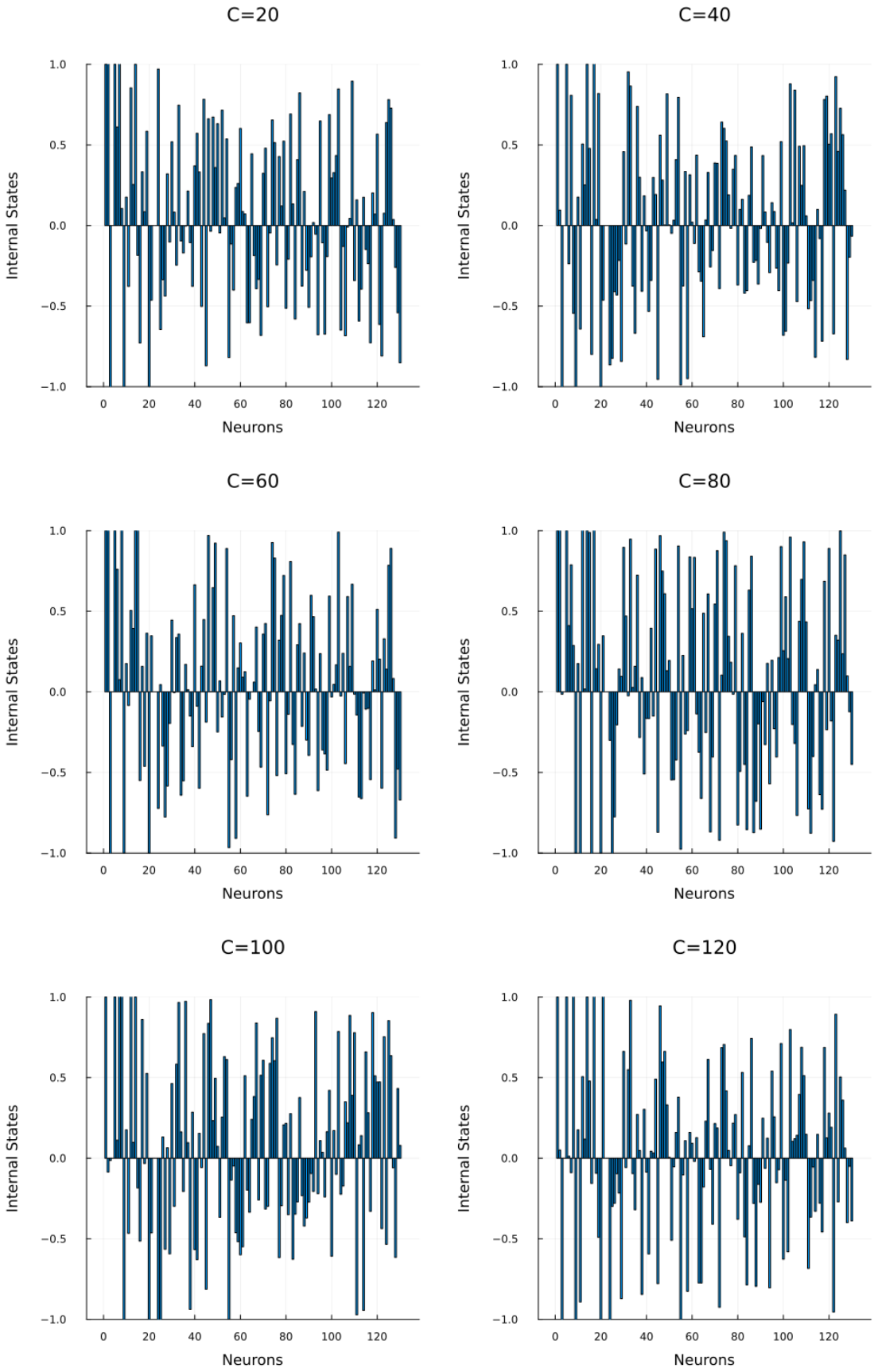


Figure 4.18: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters with the data from Unit 93

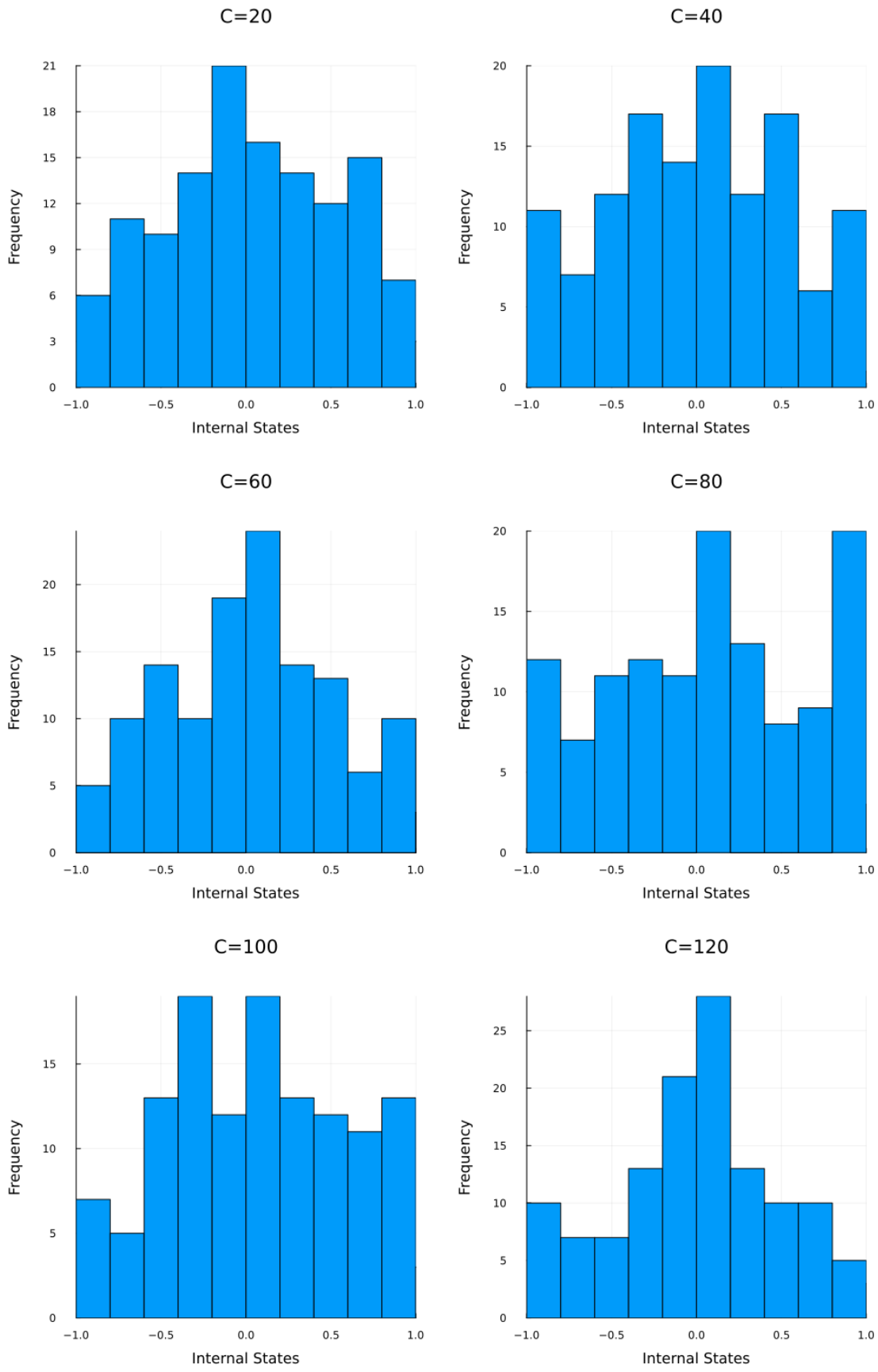


Figure 4.19: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fourth set of hyperparameters with the data from Unit 93

4.3 Punta Cana dataset

The Punta Cana (PC) dataset is a perfect showcase to study how having enough data may not guarantee an stable prediction as the underlying relations between input and output cannot be simply assimilated. The common problem found with the following predictions is the inability of the reservoir to capture the dynamics of the system from the given inputs, more sources of information are required for it to increase the prediction performance. The experimental results prove that predictions can be made on reduced temporal ranges, even though the hyperparemeter optimization is rather arduous. Table 4.4 shows the hyperparameters obtained through the ensemble and the nRMSE used to grade it as the best solution, comparing the predictions global performance against the first 500 time steps. What follows is a preliminary study on how to solve the problems that employing a reservoir for a complicated input like this poses.

Set	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β	k_{CMA}	**nRMSE**
1	987	0.3754	40.4752	75.0355	37.3118	0.1499	0.80303	49.4421	100	0.51
2	889	0.4610	94.1688	88.6220	33.9335	0.2618	0.8835	97.8458	100	0.50
3	415	0.3959	0.9565	4.6038	3.4491	0.2411	0.8166	0.9119	100	0.49
4	3607	0.7649	0.5962	9.3909	7.2666	0.2932	0.4154	7.4841	100	0.45

Table 4.4: Sets of hyperparameters for the reservoir on the PC dataset with the nRMSE of the predictions depending on the filter applied with k_{CMA} for six different ensemble runs

How much filtering was applied to the input before being introduced to the reservoir is also reflected on the results as it affects the prediction. The filter is a Central Moving Average (CMA), which is the unweighted mean of $k/2$ points forward and $k/2$ points backward from a given value (Eq. 4.1), which requires for k to be an even number. The power signal was filtered after normalization and then the δ values on Figure 3.10 calculated so these values refer to the signal given to the reservoir. Two different k values have been used, both are shown in Figure 3.11, the first being $k = 1$ so the signal remains unfiltered and the second $k = 100$, resulting in a completely filtered signal.

$$x_n^{k_{CMA}} = \frac{1}{k+1} \sum_{i=n-k/2}^{n+k/2} x_i \qquad k \in \{2j : j \in \mathbb{N}\} \tag{4.1}$$

On Table 4.4 the optimized hyperparameters for four different ensemble runs are shown. The focus will lay on the quality of the prediction for the first 5 days, how it compares to the full 2 months prediction, how it compares to a prediction without smoothing and the behaviour of the reservoir internal states during training. The optimized sets show some common characteristics among each other, the connectivity for instance can be seen to tend to the upper bound except for a single value that falls smack down the middle. The spectral radius scaling factor is also similar between sets, also tending towards the higher bound with a single outlier at the middle of the range.

Size is a different matter with this optimization, as the neurons within the reservoir go from just under 500 to more than 3500 in total. The results do not place any of the sets at the bounds established for the GWO, and allow for a study on the consequences of reservoir size for such a massive dataset. The leak rate is evenly distributed too between the upper and lower bounds, this is advantageous too, when paired with a similar spectral radius scaling factor, as it will give an insight on how the short-term memory is affected.

Two groups can be found when looking at the input scaling factors at Table 4.4: ones that are of the same order of magnitude and are several times greater than the activation function fringe values; and the others, that have a distinct magnitude between the unitary and the input scaling factor, which can only be up to ten times the fringe values. As seen previously on the MG and CMAPSS datasets the effect of these factors is fundamental on the appearance of saturation of the reservoir and will be further explored for these sets.

The reservoir input is made up of five distinct signals: normalized power consumption, δ_1, δ_{10}, δ_{100} and normalized radiation. The output on the other hand is just going to be four signals: normalized power consumption, δ_1, δ_{10} and δ_{100}. The normalized radiation is not predicted and during the generative mode it is given as a known value depending on the month and time of day. Table 4.5 are the RMSE values for the predicted δ signals, and have no actual purpose but to help finding the faults at each of the sets. Even if at a glance the error results look hopeful, the truth is these signals are incredibly hard to predict as will be seen for each of the sets.

Set	$nRMSE_{\delta 1}$	$nRMSE_{\delta 10}$	$nRMSE_{\delta 100}$
1	0.21	0.22	0.35
2	0.21	0.26	0.32
3	0.17	0.14	0.27
4	0.19	0.17	0.20

Table 4.5: Adimensional nRMSE on the prediction of the δ_1, δ_{10} and δ_{100} for each of the hyperparameter sets

As in the CMAPSS dataset results, the most promising, both in nRMSE results and reservoir saturation, will be explored in this section. The third set is the selected candidate, while an analysis of the remaining three can be found in Annex A.2.

4.3.1 Third set of hyperparameters for the PC dataset

While the third set of hyperparameters may generate an underperforming prediction it does so with the smallest reservoir, the second lowest RMSE (0.49) and a lack of saturation in the internal states during training. Perhaps this reservoir biggest difference with the other sets is the size, as it is an order or magnitude smaller than the biggest and still have the third lowest error. The leak rate is the second lowest from among the sets, close to the ones at sets 1 and 2, and the spectral radius scaling factor is in the vicinity of 0.8. Both these parameters will grant, once more, a greater short-term memory close to the one seen in the two previous sets.

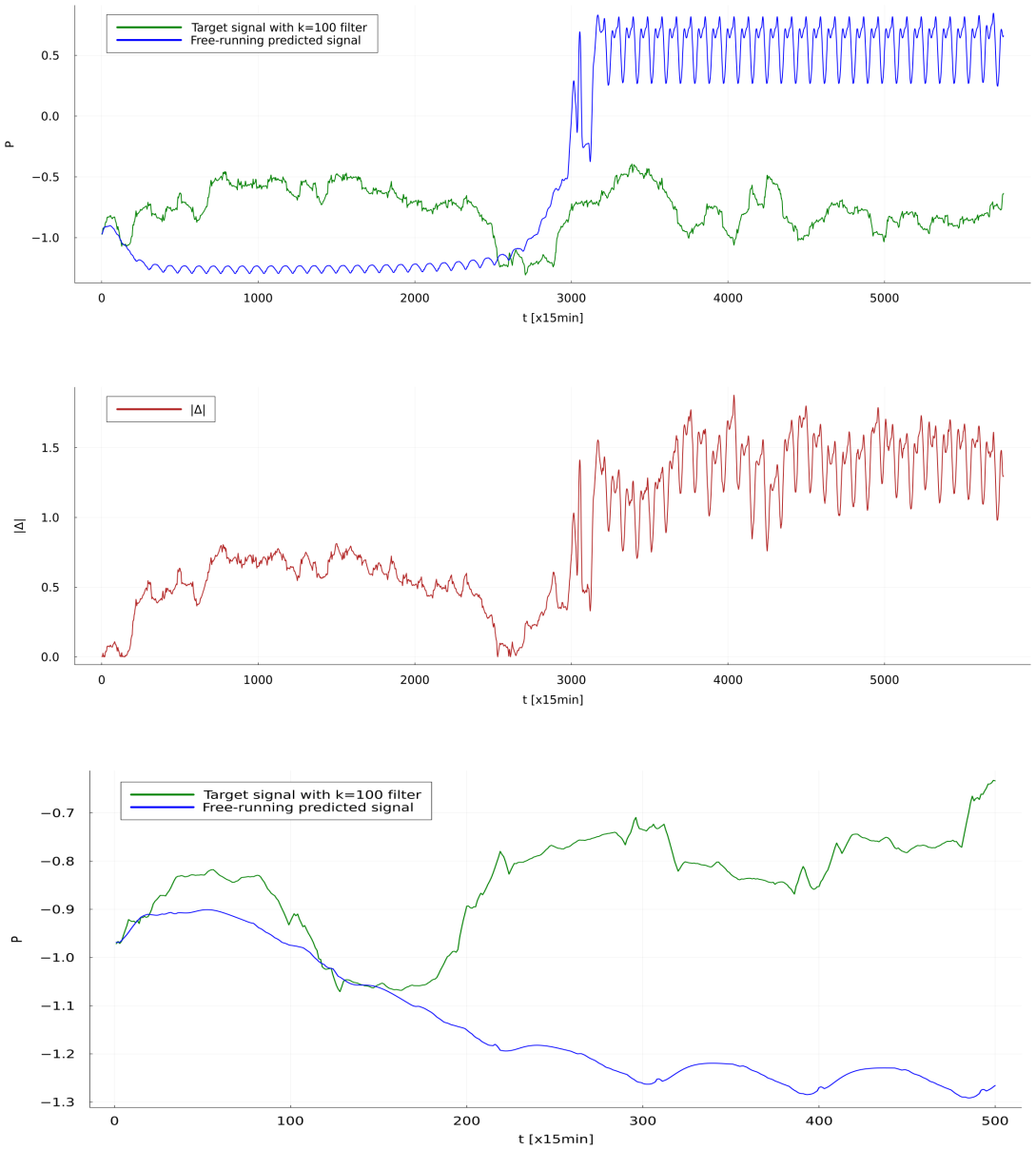


Figure 4.20: PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the third set of hyperparameters

Probably the second biggest difference from the other sets are the input scaling factors, as none of the three are higher than 5 an the highest factor is for the power signal. As seen on the MG and CMAPSS dataset, this proportion between the factors avoids saturation, as even by giving certain input predominance among the rest it does not overwhelm the neurons. It is of note that

the unitary input is the least important of the inputs, obvious taking in consideration it does nothing for the prediction itself. The auxiliary signals, δ, scaling factor comes second as it only supports the reservoir in predicting the coming dynamics. The power input has the most weight in the reservoir, but without going over an order of magnitude above the activation function fringe values, simply nudging it slightly higher without squashing out the rest of inputs.

Figure 4.20 shows the predicted signal both during the training phase and during the first 500 time steps, the abrupt change is once more present even starting earlier than in the other sets. The prediction attempts to follow the original but immediately fails to do so, not mirroring the fluctuations and quickly diverging until ultimately the prediction is nothing but a periodic fluctuation undershooting the target. The absolute difference variation, $|\Delta|$, steadily increases for a while until the instability shifts, once more, at the 3000 time-steps mark after which the difference skyrockets. The prediction differs abruptly and starts the fluctuation overshooting the target. The prediction itself is not of interest in this set, but it is how the abrupt changes come to be and the effect on the reservoir.

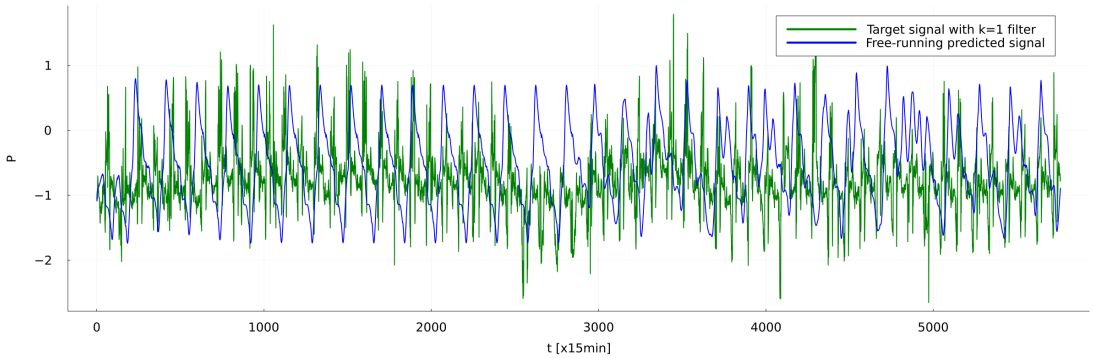


Figure 4.21: PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the third set of hyperparameters

Once again the unfiltered signal at Figure 4.21 is unable to effectively predict the the original, even with the absence of abrupt changes the input signal is too noisy for the reservoir. The period does get mirrored, and for this case the prediction can at times follow the trends and certain maxima and minima. The nRMSE for this prediction is 0.26, significantly lower than for the filtered prediction even if both are not apt for power consumption prediction.

An interesting occurrence can be observed in Figure 4.22, as the prediction for the δ auxiliary signals clearly shows the moment the abrupt change at time step 3000 appears on the reservoir, affecting the power signal as well. The most severe disturbance appears on δ_{10}, with a peak reaching up to 2.5 when the prediction and the original were fluctuating around 0. Even if δ_{10} suffered the most from the fluctuation, all evidence speaks for the initial instability to originate at δ_1 as it is the first to diverge suddenly after successfully predicting the peak at time step 3000. Previous to the abrupt change the δ predictions followed the average value of the original as depicted in Figure 4.23.

The internal states are what matters the most on this set, as they are prove than the input scaling factors are responsible for saturation. Take for instance the states at Figure 4.24, in which there is clear difference between the six snapshots during training presented. The first two time steps, 1000 and 2000, present more separability as even if a big number of neurons are at extreme values the bulk remains on the intermediate range. It is after time step 3000 that the abrupt changes kick in and the reservoir has an increased saturation. Histograms at Figure 4.25 are prove of this evolution, as initially during training the frequency for intermediate values is high and just at 3000 time steps there is a significant behaviour shift in the reservoir, with the internal states moving

to the fringe values at the end of the training phase. The reason behind this saturation, even with the reduced input scaling factors, is the size of the reservoir. After certain time steps the complexity of the dynamics is too high for such a reduced number of neurons, forcing saturation and driving the predictions to those extreme fluctuations. Increasing the size of the reservoir to 1000 and keeping this same input scaling factors led to a similar prediction but without saturation and sudden instabilities.

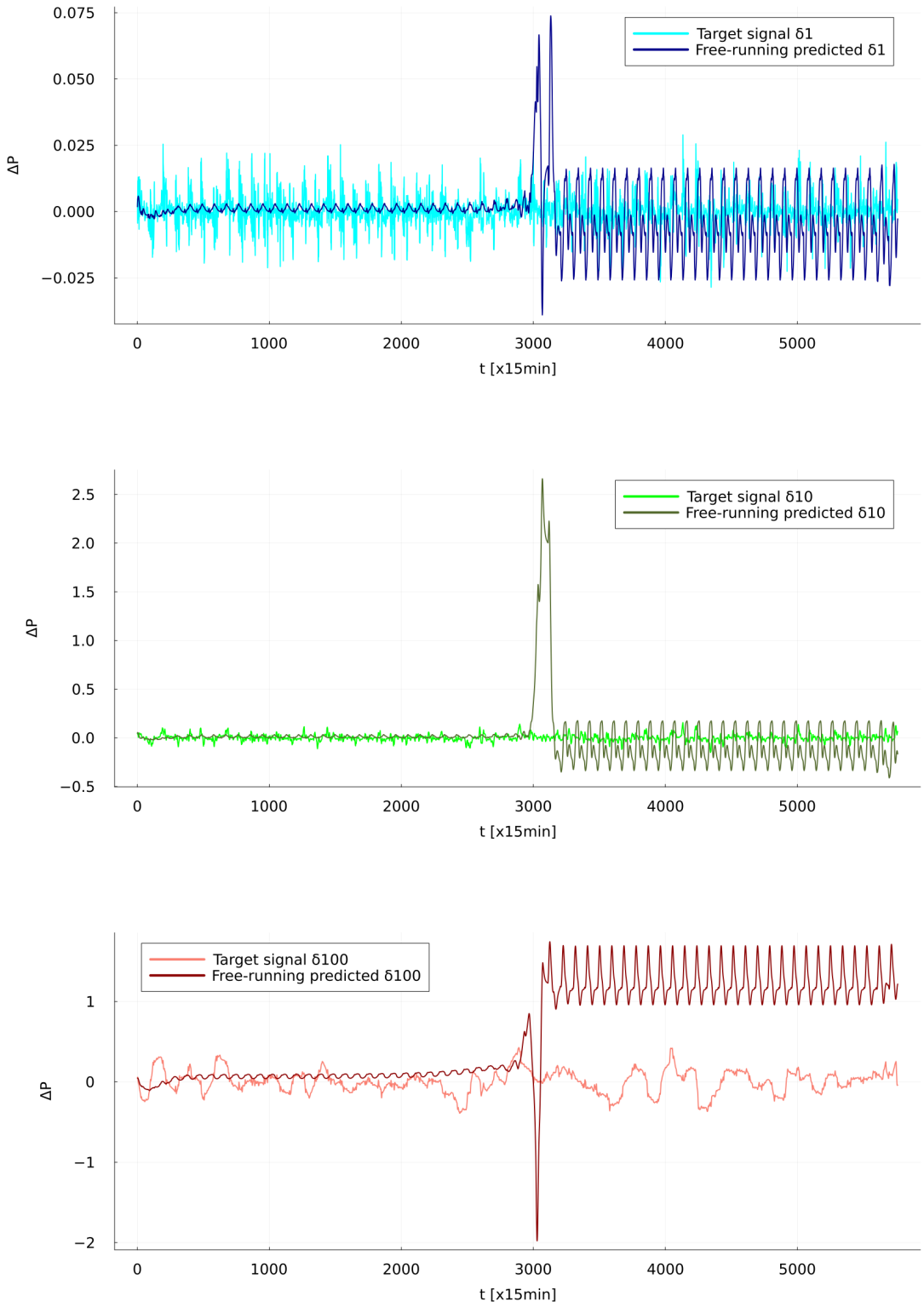


Figure 4.22: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the third set of hyperparameters



Figure 4.23: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the third set of hyperparameters

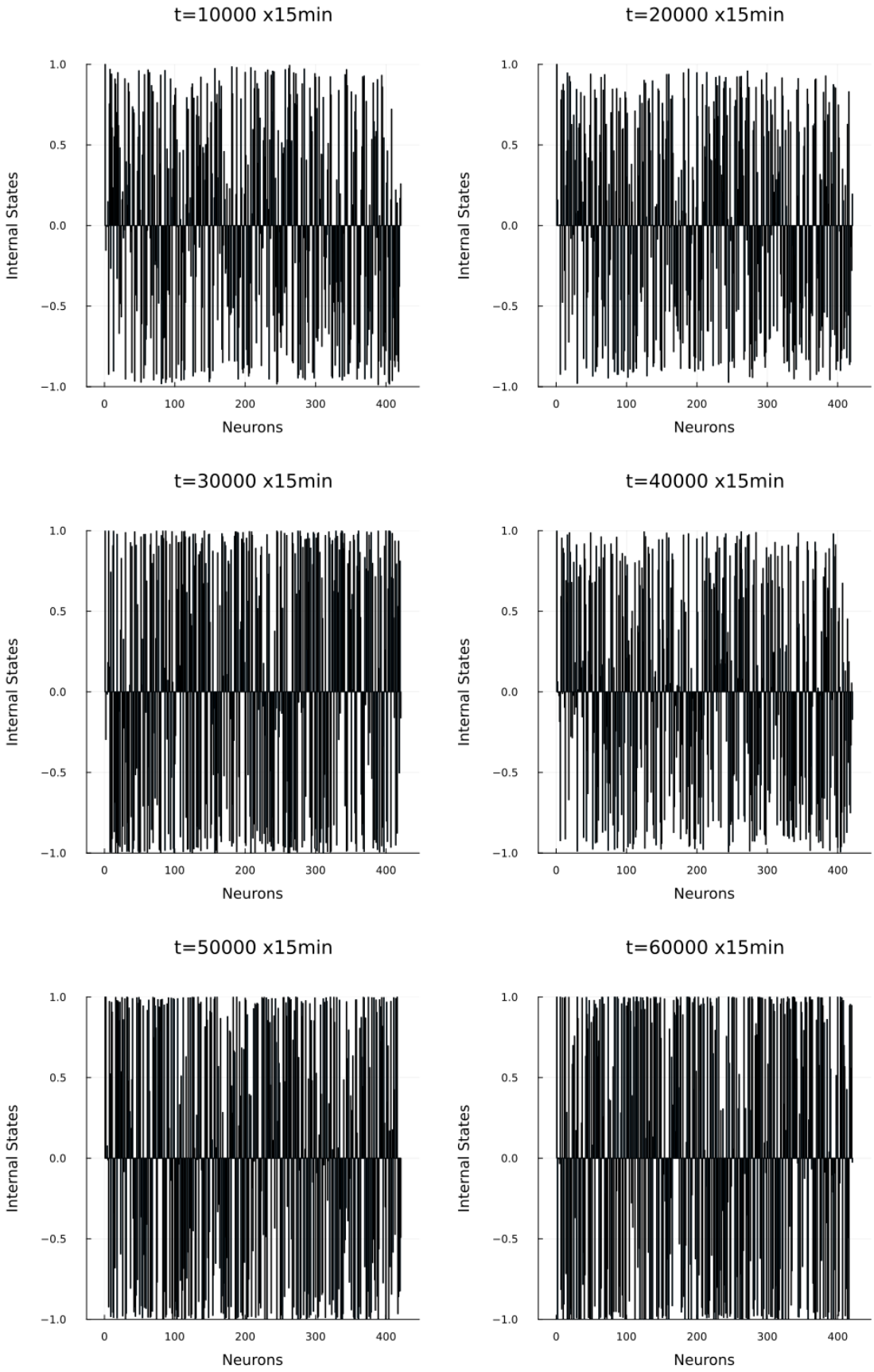


Figure 4.24: Reservoir internal states at various moments during training for the PC time series for the third set of hyperparameters

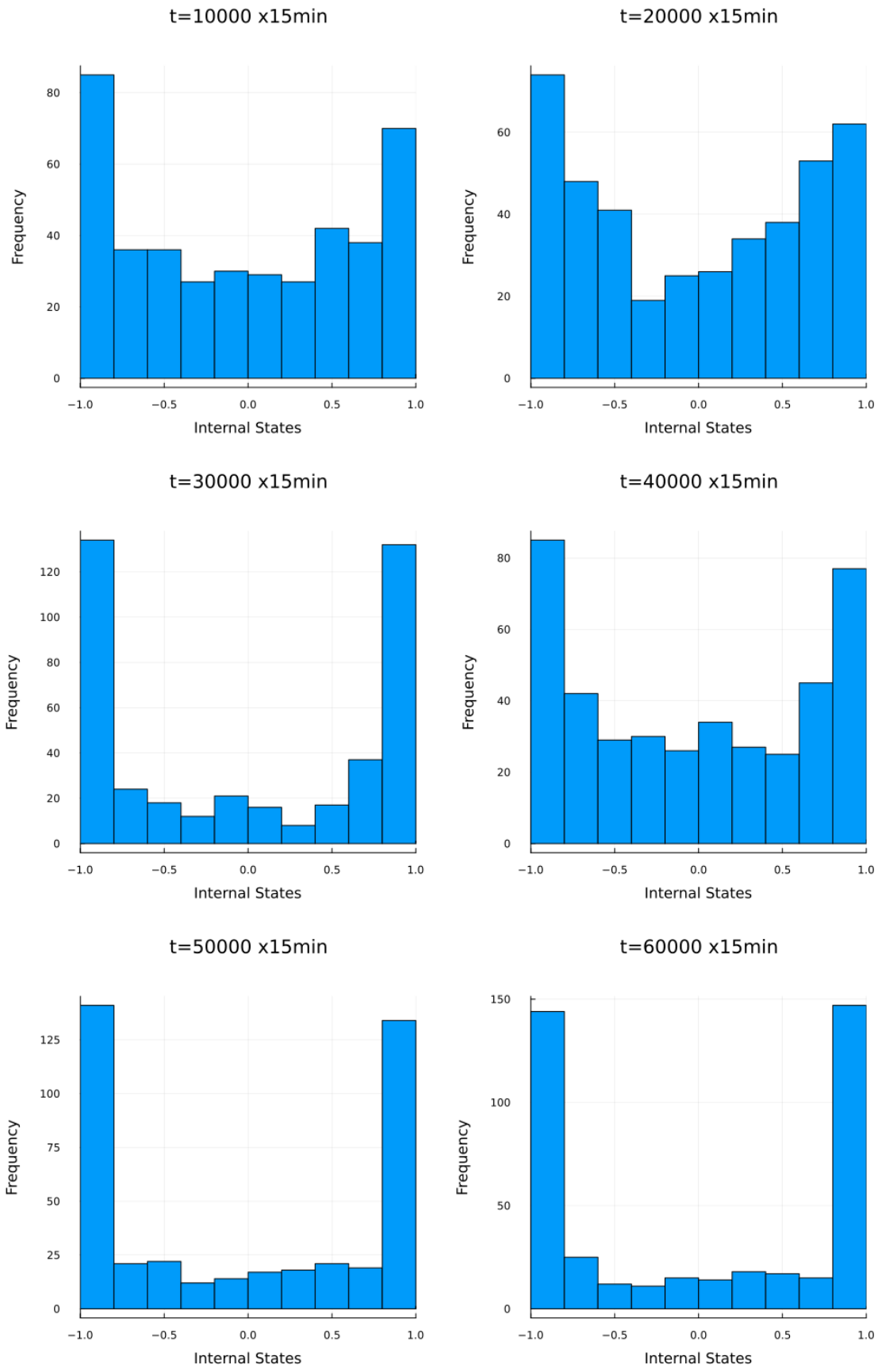


Figure 4.25: Distribution of the reservoir internal states at various moments during training for the PC time series for the third set of hyperparameters

5 | Conclusions

Predictive maintenance is a budding field in the aerospace industry with certain caveats that have made implementation sluggish, specially considering the need for data availability and sensor technologies which lack wide commercial implementation industry-wide. Specialist trained both in aerospace and data science are required to develop and evaluate data models based on historical data, creating models that assist in improving maintenance scheduling effectively while reducing overall cost. Reservoir computing is a powerful tool that may assist this endeavor, emulating future behaviour of certain components and thus predict the remaining life and how the failure will occur. Determining the how, when and what in a operational complicated system requires fine knowledge even when applying Machine Learning, as generating a prediction does not automatically grant a deterministic solution and may need interpretation for any useful output to be applied. Strides for a better comprehension on how to implement Reservoir Computing have been taken before: using FADEC data to predict turbofan engine failure [13] or detect fluctuations in avionic components [12]. What remains unanswered is developing a tool that allows for a comprehensive evaluation of the entire process, avoid the black-box effect most Machine Learning solutions exhibit so the predictions can be understood, evaluated and accepted without fear of failure that may well pose a safety risk to people, workers and companies. Transparency in the field of Artificial Intelligence is a requirement for the general well-being, assuring users and authorities that these systems are reliable tools rather than cheap shortcuts.

Therefore the purpose of this thesis comes into play, the ESN-GWO System aims at analyzing operational engineering data so predictions are generated that assist in determining if a system requires maintenance at just the right moment without loosing visibility of what is going on within the reservoir. The first step is to generate signals from a given output that serves to establish when something may fail and, from there on, develop criteria to act and verify the data is accurate and the solution may serve to reduce operating costs and increase safety. The results exposed in the thesis are a preliminary study on how to undertake this task, taking datasets and optimizing a reservoir so the predictions are fine tuned to its specific dynamics, characteristics and potential problems. Understanding how a reservoir categorizes, separates and generates information is fundamental in the sake of transparency, as a prediction can not be trusted unless the involved mechanisms are understood and certain general rules are set up to guarantee its accuracy.

The ESN-GWO System is a reservoir whose hyperparameters, which establish its configuration as well as the input and output layers, are optimized by means of a Grey Wolf Optimizer so as to find the best suited prediction for any kind of time series coming from a complicated system. By means of the optimizer the hyperparameters that define the reservoir are tested, evaluating which is the best set for a specific input signal by ranking them according to the nRMSE of the prediction. This allows for the exploration of the best possible candidate and reduce the barrier of entry to reservoir computing, as producing an accurate prediction is greatly dependant on the hyperparameters which most of the time requires a great deal of fine tuning and previous knowledge.

Three datasets have been employed to put the ensemble to test and in so help determine what

common characteristics does a reservoir require to generate an accurate prediction. The first dataset is the Mackey-Glass time series, a nonlinear time delay differential equation which chaotic dynamics and serves as a preliminary test bed before proceeding with the other datasets. The second dataset is the one with more interest in the scope of this thesis as the CMAPSS dataset contains operational data from several simulated turbofan engines, allowing for the prediction of the sensor signals from various components in the system. By observing the fluctuations from the engine anomalies can be detected at each of the signals and determine what is the component most likely to fail and when will it do so, the first step being the design of a reservoir capable of predicting the dynamics of any given turbofan during service. As for the last dataset, the Punta Cana resort facility, the objective was to predict its power consumption with the minimal available data. Given a valid prediction it can be analyzed to establish if for instance maintenance on the installation is needed or power generation reduced when consumption is expected to fall.

The predicted signals from the datasets show faults that prevent them from accurately describing the target even though the ensemble found the best possible reservoir configuration given certain optimization bounds. Optimization has proven unsuccessful, as results show, in avoiding saturation in the internal states during training. When the internal states are saturated the separability condition for inputs is lost and the echo state property does not come into play anymore, putting the output layer regression at severe disadvantage.

Even with these drawbacks the simulations can be considered a success as an initial step towards a functional system. All predictions have generated valid results that even when missing the target provide valuable information on how the reservoir behaves. Two predictions have proven great potential, leading the way for finding a functional reservoir configuration. One is the fourth set from the CMAPSS dataset, which evidence strongly suggests only requires fine tuning the input scaling factors to generate valid predictions. The other is the third set from the PC dataset as the prediction has improvement potential with its unsaturated and balanced internal states. Furthermore, the challenge such highly fluctuating target signal poses as is the PC dataset makes generating a potential solution a greater success.

In order to avoid saturation and generate accurate predictions certain configurations have been observed to be beneficial, allowing the internal states to retain separability while training and keeping the reservoir computationally cheap to run. These new rules would allow for a more precise optimization as these observations apply to the three inputs under study and can be extrapolated to work for any other input. Therefore, the findings of this thesis could be surmised as follows:

1. The size of the reservoir, N, should be big enough to capture the input signal dynamics while avoiding a massive amount of neurons that would instead cause overfitting and, in some cases, saturation. The equilibrium for a reservoir capable to classify and assimilate complex dynamics while being light enough to run on conventional software is thin. A good starting point would be to place the upper bound of the reservoir at 50% the length of the input time series and the lower bound at 20%. The best observed results on the three datasets studied in this thesis have been found in that size range.

2. The leak rate, α, and the spectral radius scaling factor, K_ρ, are both fundamental in establishing the short-term memory the reservoir has available. The most preferable combination would be to have a reservoir with slow dynamics that maintains previous inputs resonating within its internal states. The recommended range for the leak rate is to keep it between 0.1 and 0.4, to guarantee the dynamics are appropriate for the given input and necessary information is kept inside the reservoir. As for the spectral radius scaling factor, and following the observed prediction behaviours, is better to keep it between 0.9 and 0.5 so the echo state property is guaranteed as the spectral radius is under one and the reservoir may adapt to the specific input.

3. The input scaling factors, a_{Win}, are perhaps the hyperparameters that affect the most whether the reservoir internal states are saturated or not. As Lukoševičius [29] claims linear tasks benefit from small scaling values as the activation functions operate in the 0 range where it is mostly linear, while nonlinear tasks are more suited to bigger scaling values that turn the activation function into a binary switch. The problem comes when the factors are over an order of magnitude above the characteristic values of the activation function, in consequence most neurons then to output a value equal to 1 and the separability condition between inputs is lost. To avoid facing this problem while allowing for linear or nonlinear behaviour, which depends exclusively on the input, it is recommended to establish the range $(0, 1]$ for the unitary scaling factor, $(0, 10]$ for the main input scaling factor and $(0, 5]$ for any other auxiliary inputs which are not the primary objective for the prediction but give extra information.

4. The connectivity, c, represents the sparseness and how many neurons is a single one connected to as a fraction of the reservoir size. When increasing the size of the reservoir the number of connections between neurons increases as well, with a given input these contributes to the activation function reaching saturation sooner even when controlling the input scaling factors, as the neurons are bombarded with inputs not properly scaled to the volume of traffic in the reservoir. The size threshold for this to happen depends on the input signal, but once detected it is possible to either reduce the input scaling factors or the connectivity, which is the recommended option. As an starting point for the optimization is better to guarantee the sparsity of the reservoir and use the range $(0, 0.3]$ as have been used for this thesis predictions.

5. The Tikhonov regularization coefficient, β, is included in the hyperparameter optimization to ensure the stability of the predictions and prevent overfitting. On this thesis a precise strategy has not been found to improve the selection on the regularization coefficient, but an study of the solutions suggests than establishing the range $(0, 10)$ is better suited for the ridge regression of the outer layer than an upper bound of 10.

These new ranges for the system could potentially allow for more efficient optimizations, finding faster the appropriate reservoir for a given input while avoiding internal states saturation and a subpar training of the output layer. The problem presented on this thesis is not a trivial one and the proposed system grants the opportunity to begin deeper studies on the matter. Following these fundamentals it is possible to find configurations that allow short-term predictions and therefore be viable for fault detection and predictive maintenance. The best results on this thesis have always been found at the beginning of the times series, confirming there is a range at which accurate predictions can be generated.

The reservoir used for the simulations is fundamentally simple. It is the foundation stone on which to build the rest of the system and still requires further work. Reservoir modularity [48] or critical conditions [49] are advanced aspects that have not been included and greatly favor the prediction of complex time series such as the ones used here.

The ESN-GWO Ensemble has potential to speed up reservoir computation in engineering by reducing the time spent in tailoring its configuration and only requiring fine tuning. Once the time series from a complicated system is predicted, a deep analysis on potential faults and risks can be done, allowing for more precise maintenance and safer operations.

6 | Future Steps

Improvements are required for the ESN-GWO System to be an effective tool for predictive analysis. The following changes are reflections from the author on how the work done in this thesis can evolve and in the future leave the preliminary concept it is currently at and become a reality. Some of the potential steps to work on are the following:

- Develop and formalize the ground rules for early failure detection and where in the system they may occur. Once accurate predictions are available it is necessary to develop the engineering principles from where to act, establishing analysis criteria that may guarantee the predictions generated with the system are actual potential faults that need to be fixed.
- Improve the selection criteria for the Grey Wolf Optimizer, moving away from just using the nRMSE to rank the potential hyperparameter sets and selecting more accurate predictions. Certain methods exist to evaluate the similarity between two signals which are better suited to the task at hand than using an error value. For instance, methods like Dynamic Time Warping [50] or cross-correlation [51] are potential candidates for prediction ranking during optimization.
- Modify the stopping criteria for the Grey Wolf Optimizer. Instead of establishing a specific number of iterations until the optimal hyperparameters is found, the optimization could iterate until a certain nRMSE tolerance is reached on a prediction.
- To allow for an improved hyperparameter selection criteria with several signal matching methods it is necessary to select a new optimization algorithm up to the task. The Multi-objective Grey Wolf Optimizer [52] is a perfect candidate as it is based on the current optimizer used on the ensemble and would only require further improvements and adding bound enforcement to be compatible with the reservoir.
- Implement additional optimizers, such as other previously talked about meta-heuristic algorithms based on nature: Particle Swarm Optimization [34], Firefly Optimization [35] or Jellyfish Search [36].
- Change the optimization architecture so it is fully developed on *CUDA* to further improve the processing speed of the ESN-GWO System.
- Analyze the role of modularity [48], to increase input memory, and network criticality [49], to increase the accuracy by placing the reservoir at the edge of stability, in search of a Neural Network with increased performance.

Bibliography

[1] Enrico Zio. "Prognostics and Health Management (PHM): Where are we and where do we (need to) go in theory and practice".
In: *Reliability Engineering System Safety* 218 (2022), p. 108119. ISSN: 0951-8320.
DOI: https://doi.org/10.1016/j.ress.2021.108119.
URL: https://www.sciencedirect.com/science/article/pii/S0951832021006153.

[2] Steven L. Brunton et al.
"Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning".
In: *AIAA Journal* 59.8 (2021), pp. 2820–2847. DOI: 10.2514/1.J060131.

[3] Jichao Li, Xiaosong Du, and Joaquim R.R.A. Martins.
"Machine learning in aerodynamic shape optimization".
In: *Progress in Aerospace Sciences* 134 (2022), p. 100849. ISSN: 0376-0421.
DOI: https://doi.org/10.1016/j.paerosci.2022.100849.
URL: https://www.sciencedirect.com/science/article/pii/S0376042122000410.

[4] Salvatore Cuomo et al. "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next".
In: *Journal of Scientific Computing 2022 92:3* 92.3 (July 2022), pp. 1–62. ISSN: 1573-7691.
DOI: 10.1007/S10915-022-01939-Z.
URL: https://link.springer.com/article/10.1007/s10915-022-01939-z.

[5] Michael T. Tong.
"Using Machine Learning to Predict Core Sizes of High-Efficiency Turbofan Engines".
In: *Journal of Engineering for Gas Turbines and Power* 141.11 (Oct. 2019), p. 111023.
ISSN: 0742-4795. DOI: 10.1115/1.4044770.
eprint: https://asmedigitalcollection.asme.org/gasturbinespower/article-pdf/141/11/111023/6434055/gtp_141_11_111023.pdf.
URL: https://doi.org/10.1115/1.4044770.

[6] Thyago P. Carvalho et al. "A systematic literature review of machine learning methods applied to predictive maintenance".
In: *Computers Industrial Engineering* 137 (2019), p. 106024. ISSN: 0360-8352.
DOI: https://doi.org/10.1016/j.cie.2019.106024.
URL: https://www.sciencedirect.com/science/article/pii/S0360835219304838.

[7] Gian Antonio Susto et al.
"Machine Learning for Predictive Maintenance: A Multiple Classifier Approach".
In: *IEEE Transactions on Industrial Informatics* 11.3 (2015), pp. 812–820.
DOI: 10.1109/TII.2014.2349359.

[8] H.M. Hashemian and Wendell C. Bean.
"State-of-the-art predictive maintenance techniques".
In: *IEEE Transactions on Instrumentation and Measurement* 60.10 (2011), pp. 3480–3492.
DOI: 10.1109/TIM.2009.2036347.
URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-80052823830&doi=10.1109%2fTIM.2009.2036347&partnerID=40&md5=266863a3bbad32f5df0a39de7a76288a.

[9] Link C. Jaw and Sanjay Garg. "Propulsion Control Technology Development in the United States: A Historical Perspective". In: *NASA STI Program* (2005). URL: https://ntrs.nasa.gov/citations/20050238468.

[10] Alfredo Güemes et al. "Structural Health Monitoring for Advanced Composite Structures: A Review". In: *Journal of Composites Science* 4.1 (2020). ISSN: 2504-477X. DOI: 10.3390/jcs4010013. URL: https://www.mdpi.com/2504-477X/4/1/13.

[11] *FADEC 3 - 3rd generation Full Authority Digital Engine Control | Safran.* URL: https://www.safran-group.com/products-services/fadec-3-3rd-generation-full-authority-digital-engine-control.

[12] Chi Gao et al. "Medium and Long-Term Fault Prediction of Avionics Based on Echo State Network". In: vol. 2022. NLD: IOS Press, Jan. 2022. DOI: 10.1155/2022/5343909. URL: https://doi.org/10.1155/2022/5343909.

[13] M. Rigamonti, Piero Baraldi, and Enrico Zio. "Echo State Network for the Remaining Useful Life Prediction of a Turbofan Engine". In: (July 2018).

[14] J.L. Montañés. *Actuaciones, Diseño y Ensayo de Aerorreactores.* Publicaciones ETSIAE, 2020.

[15] Xinglong Zhang, Lingwei Li, and Tianhong Zhang. "Research on Real-Time Model of Turboshaft Engine with Surge Process". In: *Applied Sciences* 12.2 (2022). ISSN: 2076-3417. DOI: 10.3390/app12020744. URL: https://www.mdpi.com/2076-3417/12/2/744.

[16] *DLR - Institute of Aeroelasticity - Surge - When the air flows backwards through the engine.* URL: https://www.dlr.de/ae/en/desktopdefault.aspx/tabid-18841/30181_read-81026/.

[17] Guanya Shi et al. "Neural Lander: Stable Drone Landing Control Using Learned Dynamics". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9784–9790. DOI: 10.1109/ICRA.2019.8794351.

[18] Erwhin Irmawan, Agus Harjoko, and Andi Dharmawan. "Model, Control, and Realistic Visual 3D Simulation of VTOL Fixed-Wing Transition Flight Considering Ground Effect". In: *Drones* 7.5 (2023). ISSN: 2504-446X. DOI: 10.3390/drones7050330. URL: https://www.mdpi.com/2504-446X/7/5/330.

[19] M.R. Haque et al. "Autonomous quadcopter for product home delivery". In: *2014 International Conference on Electrical Engineering and Information Communication Technology*. 2014, pp. 1–5. DOI: 10.1109/ICEEICT.2014.6919154.

[20] B. Lantz. *Machine Learning with R.* Community experience distilled. Packt Publishing, 2013. ISBN: 9781782162155.

[21] Suzana Herculano-Houzel. "The Human Brain in Numbers: A Linearly Scaled-up Primate Brain". In: *Frontiers in Human Neuroscience* 3.NOV (Nov. 2009). ISSN: 16625161. DOI: 10.3389/NEURO.09.031.2009. URL: /pmc/articles/PMC2776484/%20/pmc/articles/PMC2776484/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2776484/.

[22] Zoran Konkoli. "Reservoir Computing". In: *Encyclopedia of Complexity and Systems Science* (2017), pp. 1–12. DOI: 10.1007/978-3-642-27737-5{_}683-1. URL: https://link.springer.com/referenceworkentry/10.1007/978-3-642-27737-5_683-1.

[23] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an Erratum note 1". In: (2010).

[24] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-time computing without stable states: a new framework for neural computation based on perturbations".
In: *Neural computation* 14.11 (Nov. 2002), pp. 2531–2560. ISSN: 0899-7667.
DOI: 10.1162/089976602760407955.
URL: https://pubmed.ncbi.nlm.nih.gov/12433288/.

[25] Hongwei Cai et al. "Brain organoid reservoir computing for artificial intelligence".
In: *Nature Electronics 2023 6:12* 6.12 (Dec. 2023), pp. 1032–1039. ISSN: 2520-1131.
DOI: 10.1038/s41928-023-01069-w.
URL: https://www.nature.com/articles/s41928-023-01069-w.

[26] Charles H. Bennett. "Universal computation and physical dynamics".
In: *Physica D: Nonlinear Phenomena* 86.1-2 (Sept. 1995), pp. 268–273. ISSN: 0167-2789.
DOI: 10.1016/0167-2789(95)00107-F.

[27] Herbert Jaeger. "Echo state network". In: *Scholarpedia* 2.9 (2007), p. 2330. ISSN: 1941-6016.
DOI: 10.4249/SCHOLARPEDIA.2330.

[28] *A brief introduction to Reservoir Computing | Francesco Martinuzzi.*
URL: https://martinuzzifrancesco.github.io/posts/a-brief-introduction-to-reservoir-computing/#3.

[29] Mantas Lukoševičius. "A practical guide to applying echo state networks".
In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7700 LECTURE NO (2012), pp. 659–686.
ISSN: 16113349. DOI: 10.1007/978-3-642-35289-8{_}36/COVER.
URL: https://link.springer.com/chapter/10.1007/978-3-642-35289-8_36.

[30] Sachin Desale et al.
"Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey".
In: *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS VOLUME 2, ISSUE 5*. Citeseer, 2015. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=039da802e218c89a3c49e235af8e555b2fdfa063.

[31] Frank H.F. Leung et al. "Tuning of the structure and parameters of a neural network using an improved genetic algorithm".
In: *IEEE Transactions on Neural Networks* 14.1 (Jan. 2003), pp. 79–88. ISSN: 10459227.
DOI: 10.1109/TNN.2002.804317.

[32] L. M.Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy.
"Simulated Annealing Algorithm for Deep Learning".
In: *Procedia Computer Science* 72 (Jan. 2015), pp. 137–144. ISSN: 1877-0509.
DOI: 10.1016/J.PROCS.2015.12.114.

[33] Ekrem Duman, Mitat Uysal, and Ali Fuat Alkaya. "Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem".
In: *Information Sciences* 217 (Dec. 2012), pp. 65–77. ISSN: 0020-0255.
DOI: 10.1016/J.INS.2012.06.032.

[34] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks* 4 (), pp. 1942–1948.
DOI: 10.1109/ICNN.1995.488968.
URL: http://ieeexplore.ieee.org/document/488968/.

[35] Xin She Yang. "Firefly algorithms for multimodal optimization".
In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5792 LNCS (2009), pp. 169–178.
ISSN: 03029743. DOI: 10.1007/978-3-642-04944-6{_}14/COVER.
URL: https://link.springer.com/chapter/10.1007/978-3-642-04944-6_14.

[36] Jui Sheng Chou and Dinh Nhat Truong.
"A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean".

In: *Applied Mathematics and Computation* 389 (Jan. 2021), p. 125535. ISSN: 0096-3003. DOI: 10.1016/J.AMC.2020.125535.

[37] Aytaç Altan, Seçkin Karasu, and Enrico Zio.
"A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer".
In: *Applied Soft Computing* 100 (Mar. 2021), p. 106996. ISSN: 1568-4946.
DOI: 10.1016/J.ASOC.2020.106996.

[38] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey Wolf Optimizer".
In: *Advances in Engineering Software* 69 (Mar. 2014), pp. 46–61. ISSN: 0965-9978.
DOI: 10.1016/J.ADVENGSOFT.2013.12.007.

[39] Juan Manuel Rodríguez Riesgo and Juan Luis Cabrera Fernández.
"Reservoir Neural Network Computing for Time Series Forecasting in Aerospace: Potential Applications to Predictive Maintenance". In: *Engineering Proceedings* 68.1 (2024).
ISSN: 2673-4591. DOI: 10.3390/engproc2024068017.
URL: https://www.mdpi.com/2673-4591/68/1/17.

[40] Michael C. Mackey and Leon Glass.
"Oscillation and Chaos in Physiological Control Systems".
In: *Science* 197.4300 (1977), pp. 287–289. ISSN: 00368075. DOI: 10.1126/SCIENCE.267326.
URL: https://www.science.org/doi/10.1126/science.267326.

[41] Abhinav Saxena et al.
"Damage propagation modeling for aircraft engine run-to-failure simulation". In: *2008 International Conference on Prognostics and Health Management, PHM 2008* (2008).
DOI: 10.1109/PHM.2008.4711414.

[42] *Prognostics Center of Excellence Data Set Repository - NASA.*
URL: https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/.

[43] Mingjing Xu et al. "Fault prognostics by an ensemble of Echo State Networks in presence of event based measurements".
In: *Engineering Applications of Artificial Intelligence* 87 (Jan. 2020), p. 103346.
ISSN: 0952-1976. DOI: 10.1016/J.ENGAPPAI.2019.103346.

[44] *NSRDB.* URL: https://nsrdb.nrel.gov/data-viewer.

[45] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing".
In: *https://doi.org/10.1137/141000671* 59.1 (Feb. 2017), pp. 65–98. ISSN: 00361445.
DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671.

[46] Tim Besard, Christophe Foket, and Bjorn De Sutter.
"Effective Extensible Programming: Unleashing Julia on GPUs".
In: *IEEE Transactions on Parallel and Distributed Systems* (2018). ISSN: 1045-9219.
DOI: 10.1109/TPDS.2018.2872064. arXiv: 1712.03112 [cs.PL].

[47] Tim Besard et al.
"Rapid software prototyping for heterogeneous and distributed platforms".
In: *Advances in Engineering Software* 132 (2019), pp. 29–46.

[48] Nathaniel Rodriguez, Eduardo Izquierdo, and Yong Yeol Ahn.
"Optimal modularity and memory capacity of neural reservoirs".
In: *Network Neuroscience* 3.2 (Apr. 2019), pp. 551–566. ISSN: 24721751.
DOI: 10.1162/NETN{_}A{_}00082. URL: https://dx.doi.org/10.1162/netn_a_00082.

[49] Swarnendu Mandal and Manish Dev Shrimali.
"Achieving criticality for reservoir computing using environment-induced explosive death".
In: *Chaos* 31.3 (Mar. 2021). ISSN: 10897682. DOI: 10.1063/5.0038881/342107.
URL: /aip/cha/article/31/3/031101/342107/Achieving-criticality-for-reservoir-computing.

[50] Omer Gold and Micha Sharir. "Dynamic Time Warping and Geometric Edit Distance". In: *ACM Transactions on Algorithms (TALG)* 14.4 (Aug. 2018). ISSN: 15496333. DOI: 10.1145/3230734. URL: https://dl.acm.org/doi/10.1145/3230734.

[51] Pejman Tahmasebi, Ardeshir Hezarkhani, and Muhammad Sahimi. "Multiple-point geostatistical modeling based on the cross-correlation functions". In: *Computational Geosciences* 16.3 (June 2012), pp. 779–797. ISSN: 14200597. DOI: 10.1007/S10596-012-9287-1/METRICS. URL: https://link.springer.com/article/10.1007/s10596-012-9287-1.

[52] Seyedali Mirjalili et al. "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization". In: *Expert Systems with Applications* 47 (Apr. 2016), pp. 106–119. ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2015.10.039.

A | Additional experimental results

Predictions and analysis were carried out for the remaining hyperparameter sets of both the CMAPSS dataset and the PC dataset that were not shown in the main body of the thesis. The content and methodology of each of the analysis is identical to the ones at Subsections 4.3.1 and 4.2.1.

A.1 CMAPSS dataset

A.1.1 First set of hyperparameters for the CMAPSS dataset

The first set of hyperparameters to be discussed is the one with the least amount of neurons and the lowest average error, a position shared with the fourth set in Subsection 4.2.1. Short-term memory is an apparent issue with this configuration, as the leak rate is the highest and the spectral radius scaling factor is the second highest, which in turn leaves the reservoir in an in-between of remembering previous inputs as they both pull in opposite directions for the memory capacity. All input scaling factors are found in the same range, which is an order of magnitude bigger than the fringe activation functions value, and most likely will contribute to its saturation, as it does not contribute at all in assisting the reservoir in giving more weight to the appropriate inputs.

Figure A.1 shows the nRMSE value of each of the signals for the prediction generated with the first set of hyperparameters, having the average for the set at Table 4.2. The reasoning behind employing the average error for selecting the best configuration is to guarantee a global minimization, including those that are supposed to be null to guarantee that the signals that are constant are also represented fairly. In Figures 3.3 to 3.8 there are certain signals that are constant throughout the training and testing phase, such as Operational Setting 3 and Signals 1, 5, 10, 16, 17, 18 and 19. However in Figure A.1 only Operational Setting 3 and Signals 18 and 19 show a null error, which means the reservoir has correctly predicted a constant behaviour through the testing phase, while in Signals 1, 5, 10 and 16 the RMSE value is some of the highest at 0.5 for all of them. Those signals that have a nRMSE that should be zero need to be also minimized so the prediction is accurate.

The rest of the signals present an error value between 0.3 and 0.5 driving down the average error value of the prediction. For instance in Operational Setting 1, shown in Figure A.2, the prediction does not match completely the original but attempts to follow the same pattern through the testing phase, mimicking with some sort of success during the testing phase. As can be seen in the absolute difference variation through time graph, $|\Delta|$, null difference values seems to be common up to the 80 cycles mark, where it shoots up due to the prediction not matching the sudden rise in the original.

Figure A.3 shows the prediction for Operational Setting 2, with similar shortfalls as Operational Setting 1. The initial prediction starts off by undershooting the original to then overshoot throughout what is left of the testing phase without diverging completely. No particular feature from the

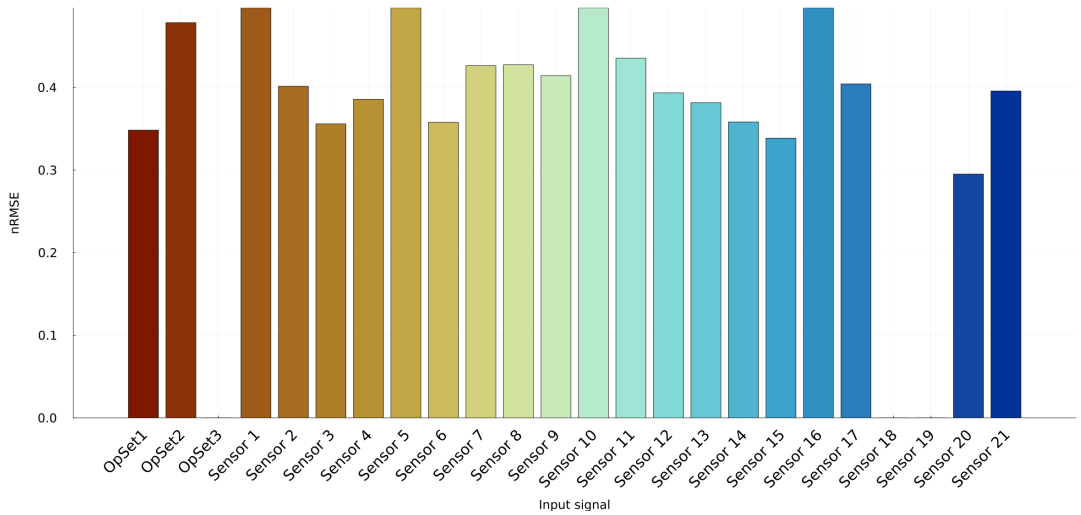


Figure A.1: RMSE value of each of the signals for the prediction generated by the reservoir with the first set of hyperparameters. Average RMSE: 1.2784

target is replicated by the free-running signal, thus contributing to its error. This prediction is the one with second highest error (0.48), as can be confirmed by $|\Delta|$ which reaches up to 3 in the region where the prediction goes opposite compared to the original. On the other hand, Sensor 2 in Figure A.4 has an average nRMSE value (0.4) and keeps through the entire test phase close to the original. The $|\Delta|$ variation remains low up to the 750 cycles, in which then quickly downplays the predicted values with respect to the original. The rest of the signal can be considered a good match, as in some cycles, specially between 15 to 20 and 40 to 50, $|\Delta|$ is minimal and the trends in the sensor can be considered as usable.

Sensor 3, at Figure A.5, is predicted with problems similar to the ones in the Operational Settings as it has an mRMSE value of 0.36 and only matches the variations of the target signal during the initial cycles of the test phase. Up to 30 cycles the prediction follows the original with some success, but then quickly follows a downward tendency whether as in the original the trend is upwards. This leads for the $|\Delta|$ variation to reach maximum values at the end of the phase but as the prediction does not stride far from the original it retains a moderate error. As for Sensor 7, Figure A.6, the increased divergence between the original and the predicted signal is much more noticeable, as the $|\Delta|$ variation initially has maximums of around 1 but as cycles progresses it reaches values surpassing 3. The predicted signal closely follows the target up until around 40 cycles, and then it overshoots and reaches much higher values than the original.

The last example, Sensor 14 in Figure A.7, has the second lowest RMSE from the non-constant signals at 0.36 and it is clear than even though it does not predict specific drops or rises it sticks close to the original throughout the entire testing phase. As for $|\Delta|$ its maximums are kept low through the prediction, and only reflects the divergence from the target at the very end. Even if it is not the example with the closest similarities to the original it does a solid job at reflecting the trend changes that can be expected from this signal.

The internal states during training, shown in Figure A.9 and as an histogram at Figure **??**, certainly show saturation for half of the neurons at any given cycle. It is particularly problematic, as at any given input this reservoir will adopt an internal state with maximum or very close to zeros values which will make training difficult as most inputs will be indistinguishable from each other. For this configuration the main contribution to the saturation is the input scaling factors that increase the global values that are introduced into the reservoir rather than helping focus on a specific subset of the signals.

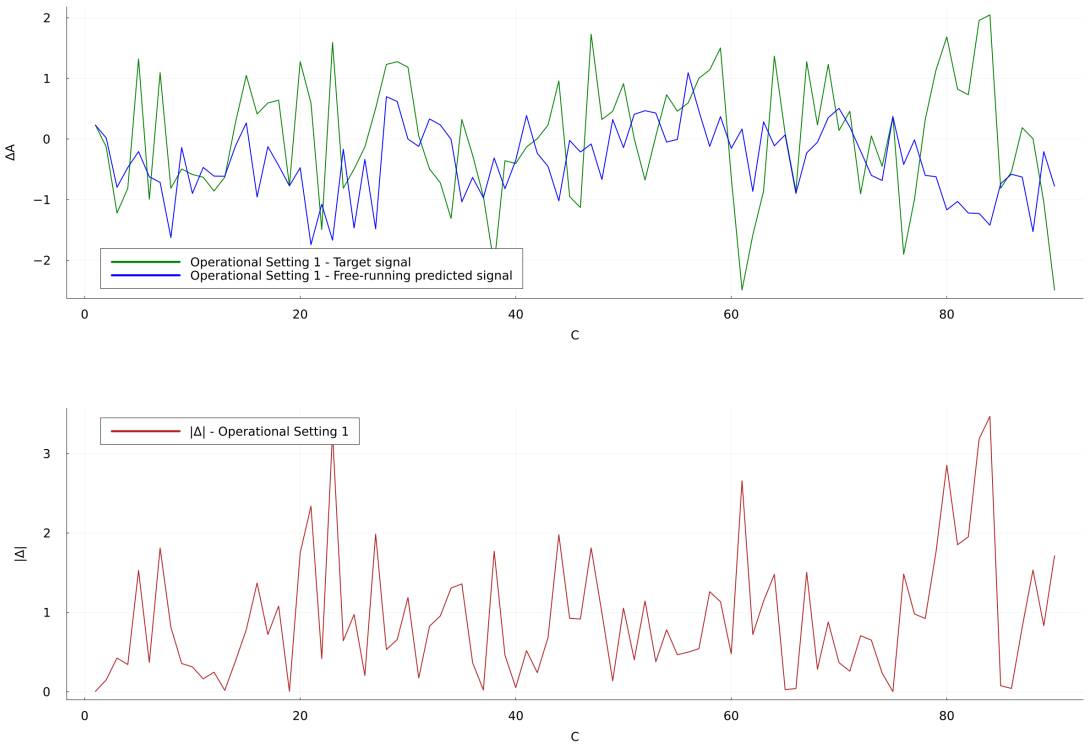


Figure A.2: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyper-parameters

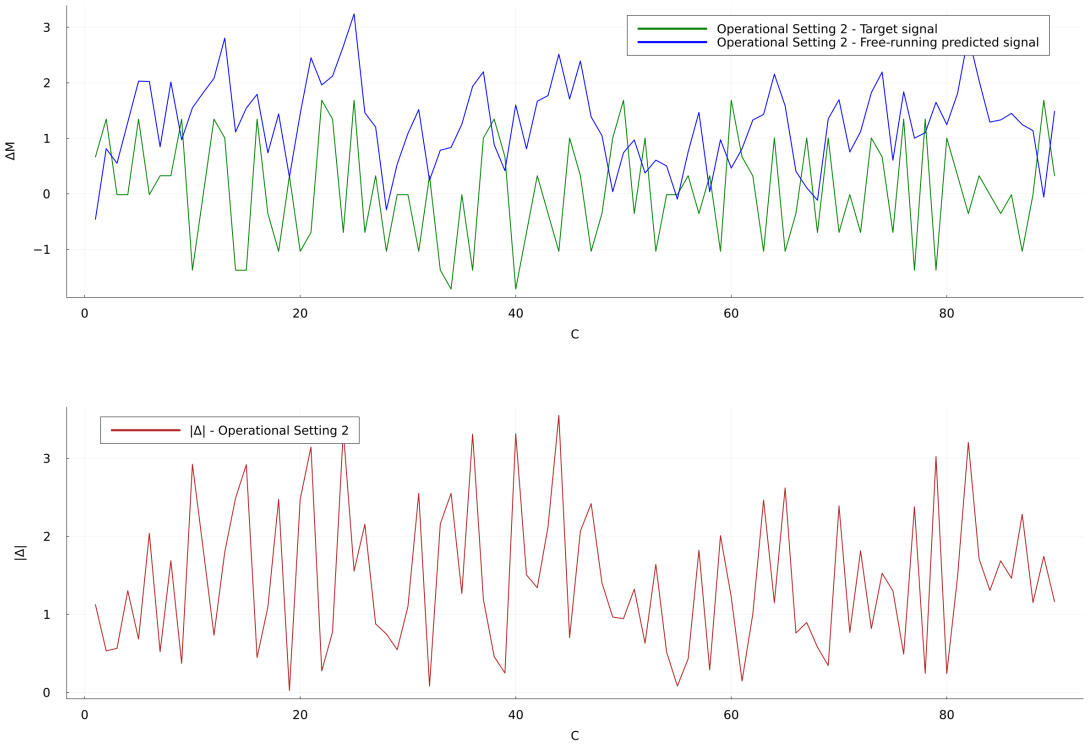


Figure A.3: Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyper-parameters

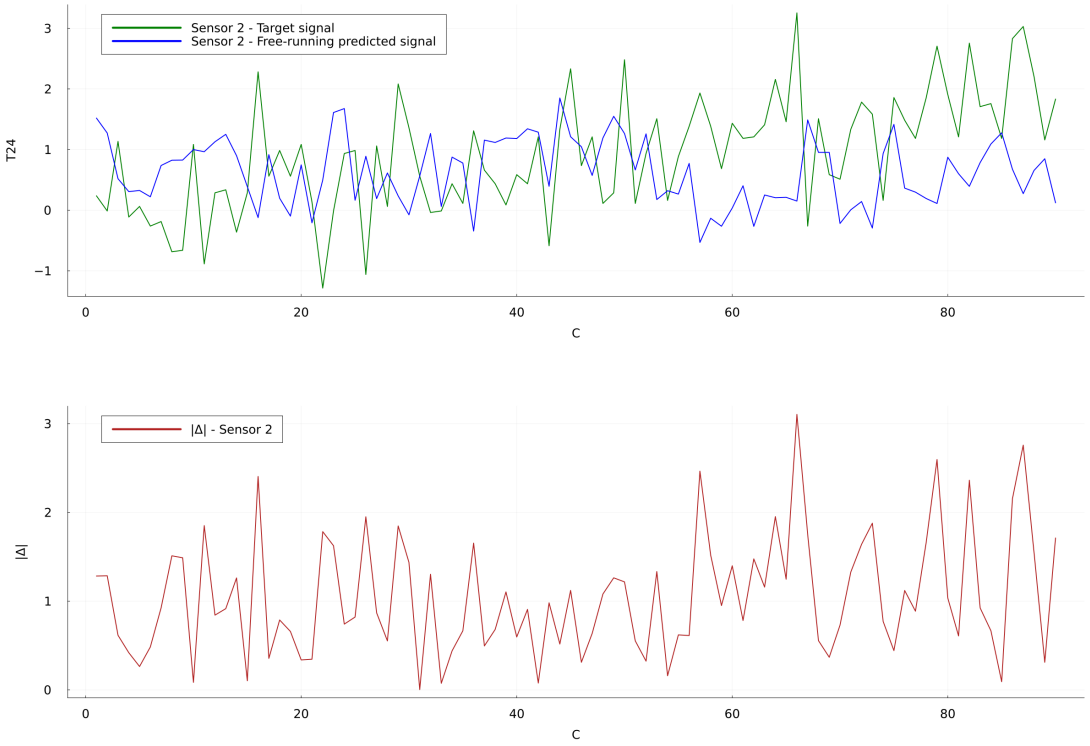


Figure A.4: Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters

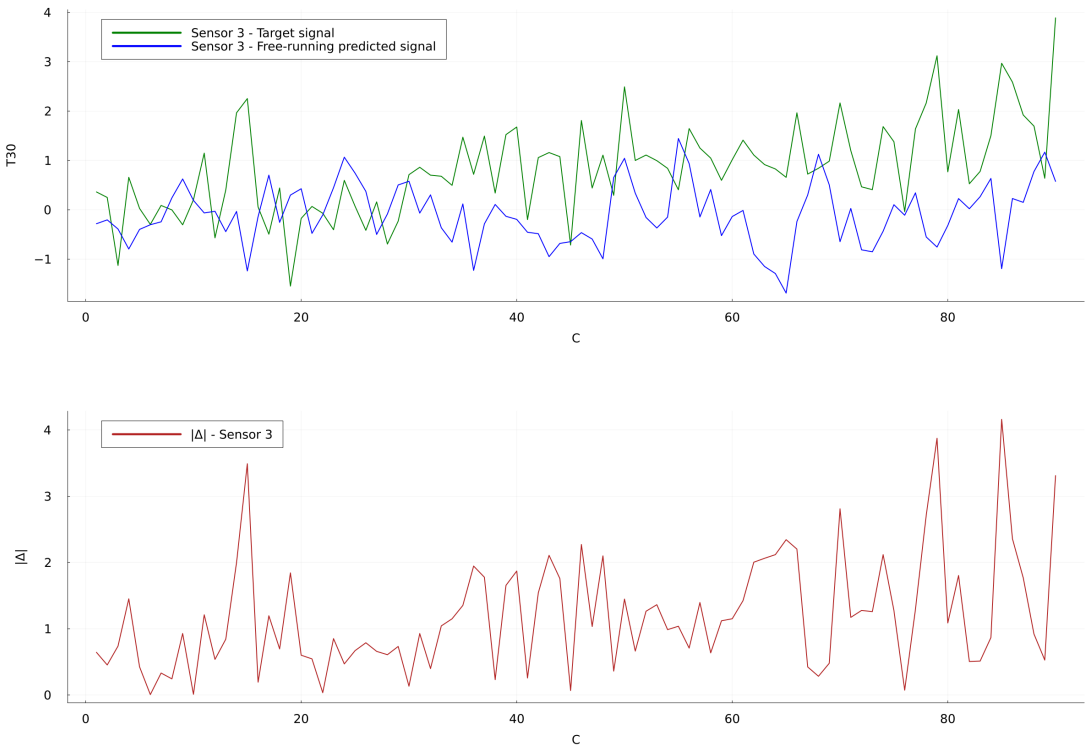


Figure A.5: Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters

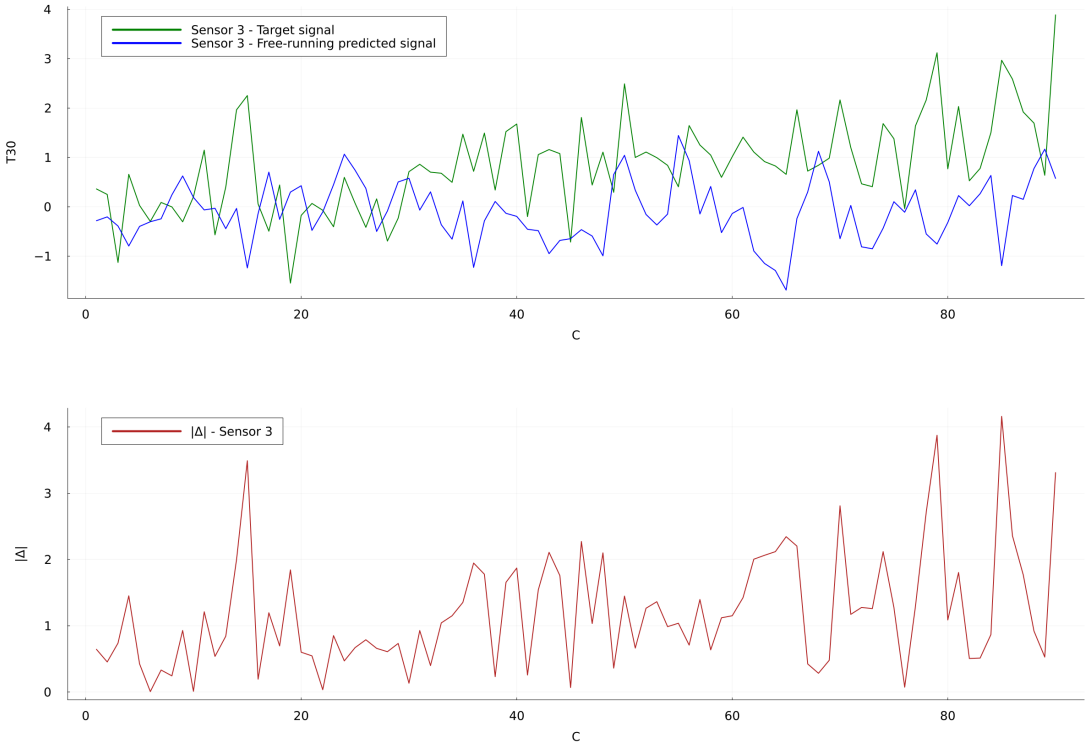


Figure A.6: Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters

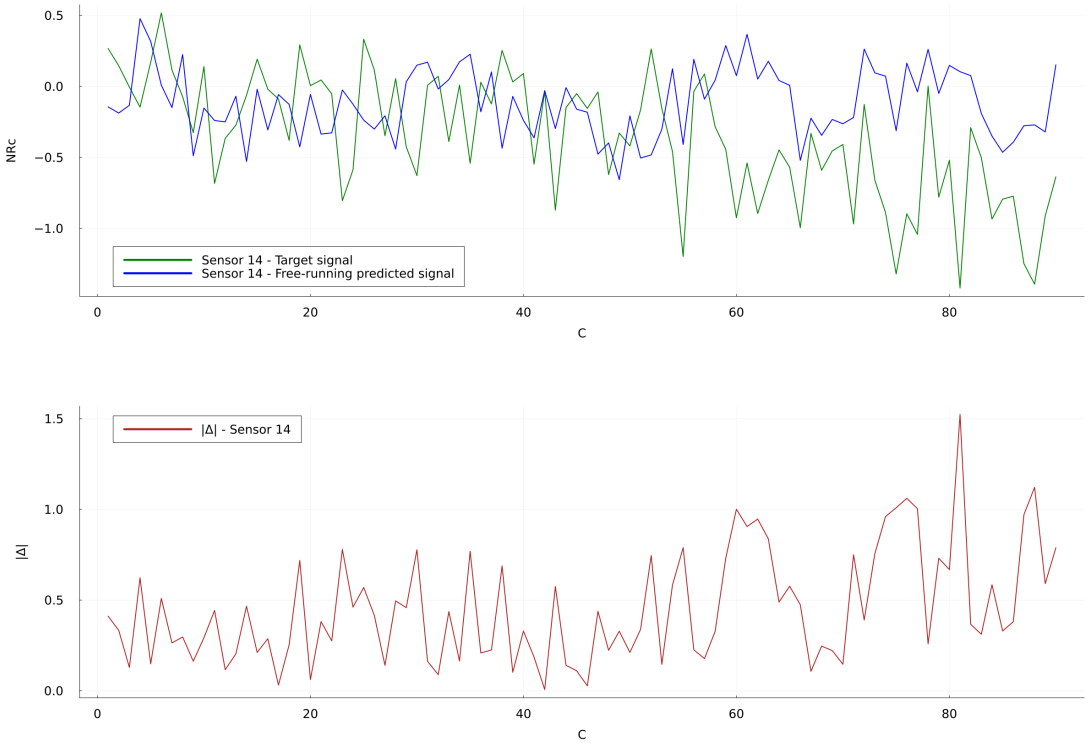


Figure A.7: Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the first set of hyperparameters

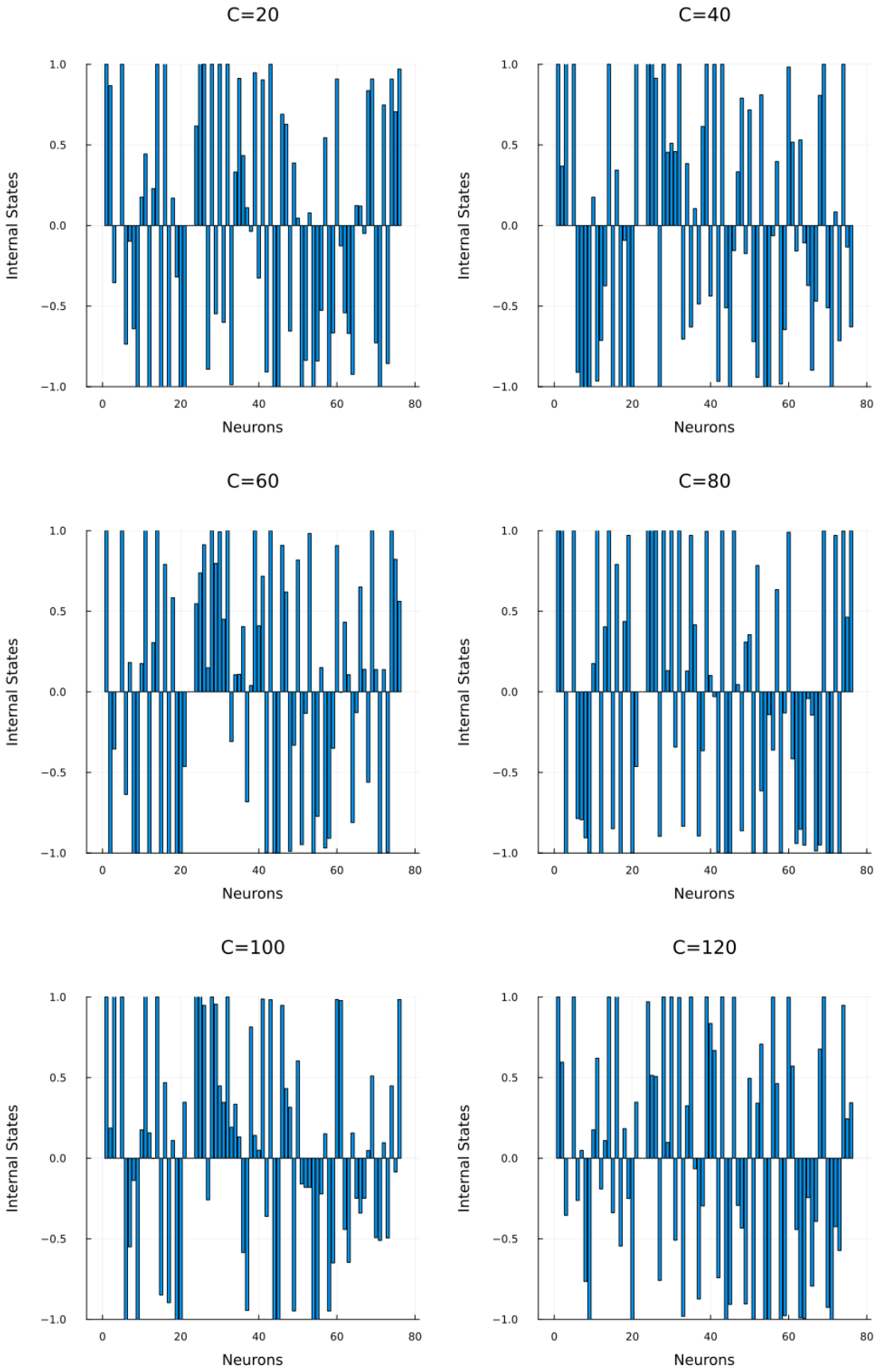


Figure A.8: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the first set of hyperparameters

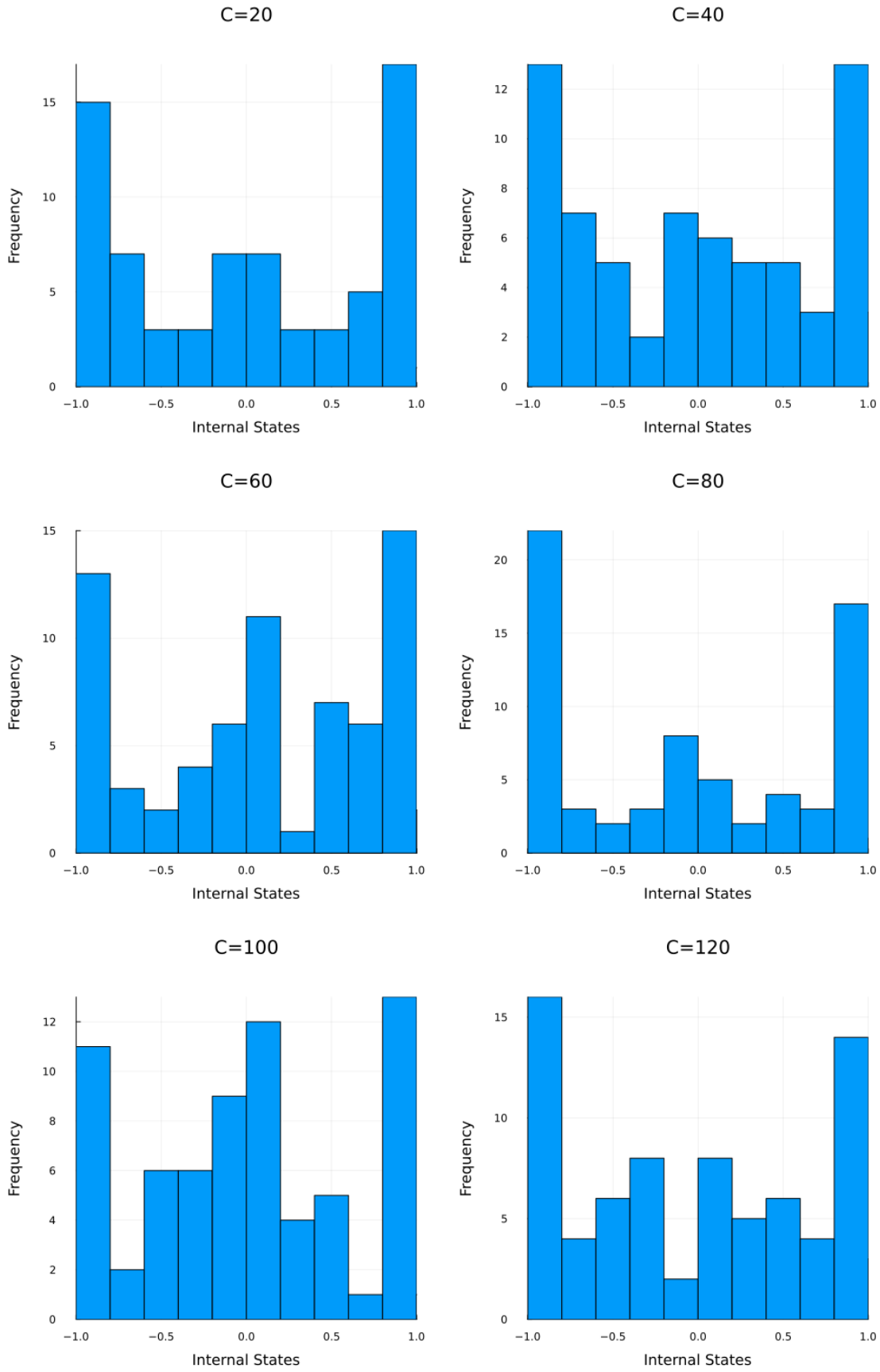


Figure A.9: Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the first set of hyperparameters

A.1.2 Second set of hyperparameters for the CMAPSS dataset

The second set of hyperparameters has the second biggest reservoir size and the lowest regularization coefficient, which is not a defining factor but goes against the trend of higher valued *Tikhonov* coefficients. Once again the neurons are closer to 500, an order of magnitude bigger than the first and also an order more expensive to compute. The leak rate is also among the lowest of the set, along with the also low scaling matrix factor makes for a short-term memory with slow dynamics but that does not hold much information on the previous inputs, behaving in a similar fashion to the memory described for the first set. The main disadvantage once again for this set are the input scaling factors as the three of them are around 90 and will probably cause saturation in the reservoir as in the previous cases.

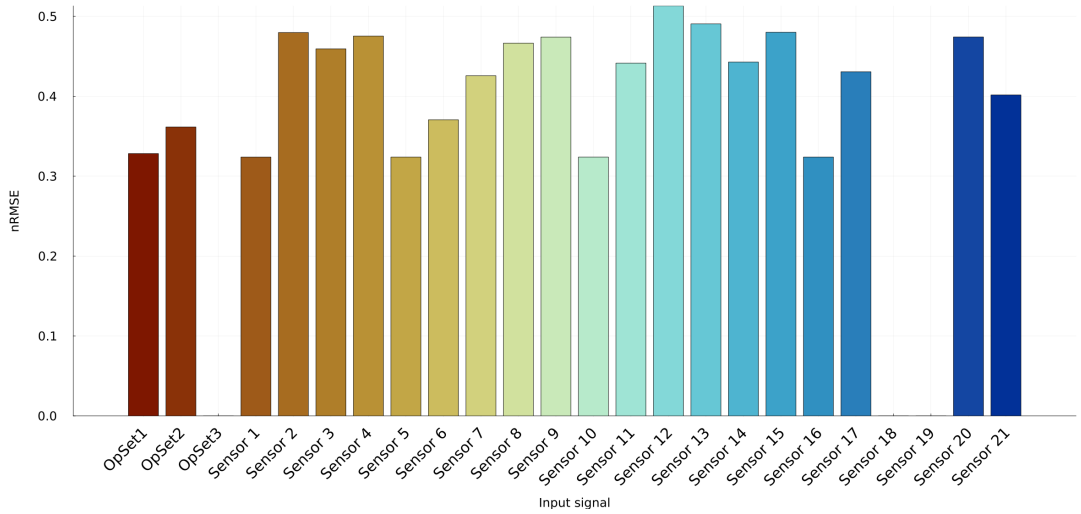


Figure A.10: nRMSE value of each of the signals for the prediction generated by the reservoir with the second set of hyperparameters.

In Figure A.10 the nRMSE is shown for each of the signals, and the predictions for Operational Setting 3 and Sensors 18 and 19 are accurate and generated a constant signal during the training phase. However for the Sensors 1, 5, 10, 16 and 17 the prediction is severely off course, with a moderate error, lower than at Set 1, and not generating constant signals such as the other ones. The average error is 0.37, with most of the signal errors being within the 0.3 to 0.5 range as in the previous set.

The prediction for the Operational Settings, Figure A.11 for the first and Figure A.12 for the second, are underwhelming and even though there are no severe divergences from the original it is still not fully capturing its trends. For Operational Setting 1 the absolute difference variation, $|\Delta|$, shows that only at certain points such as cycles 15 to 25 or cycles 65 to 75 the trends are predicted while the rest of the testing phase the generated signal is nearby the target. As for the second Operational Setting the nRMSE is also lower as with the other two sets, the first with 0.33 and the second with 0.36, and the absolute difference shows that it does predict the original better than the first one. For instance, the initial stages of the phase up to 10 cycles and the drops at around 50 cycles are matched appropriately with the only outlier being the huge predicted drop at 79 which takes the $|\Delta|$ up to 3.

Sensor 2, at Figure A.13, has an nRMSE of 0.48 and an absolute difference variation maximum at 3.5, so the prediction does not mirror the slightly upward trend of the original, however some trends are replicated. Take for instance the range between cycles 0 and 10 where the rises and drops are followed at a phaseout, or the one at 26 where the coincidence for that peaks is almost exact. The $|\Delta|$ variation clearly shows that there is divergence between the two through the phase

and steadily increaeses when the free-running signal undershoots the original. In Figure A.14, for Sensor 3, the situation is similar as the previous Sensor, the nRMSE is 0.46 and the $|\Delta|$ maximum is at 3.5 once again. The predicted signal matches the original at just one place, the rise at 5 cycles. Throughout the rest of the phase the prediction follows the target in it ascending motion with some margin, but with no evident divergence from the two of them up until the end of the testing phase.

With the last two Sensors 7, at Figure A.15 and 14 at Figure A.16, have an nRMSE of 0.43 for the first 0.44 for the second. Sensor 7 actually has an absolute difference maximum of 4.5, which happens due to the divergence starting on cycle 25 which propagates onward. During the rest of the testing phase the prediction has no success following the original in its downward motion and there is no specific point in which the behaviour is completely mirrored. The last of the Sensors is also a remarkable case, as the error is not the highest, and yet there are no recognizable trends on the original the prediction follows with a substantial divergence due to an overshoot. The $|\Delta|$ variation is high at a maximum value of 2, steadily increasing throughout the entire testing phase.

The internal states of the reservoir, depicted in Figure A.17 and as a histogram at Figure A.18, show once again a severely saturated reservoir as was expected by the optimized input scaling factors in Table 4.2. The sampled cycles during training show most of the neurons at the activation function reach extreme values, degrading the capacity for training as the characteristics of the target are lost. Only at the end of training, cycle 100 onward, does the internal states present a little bit more of separability with intermediate values. Once again the problem with having the input scaling factors with several orders of magnitude above unitary is present, which degrades the capacity of the reservoir and makes training less useful.

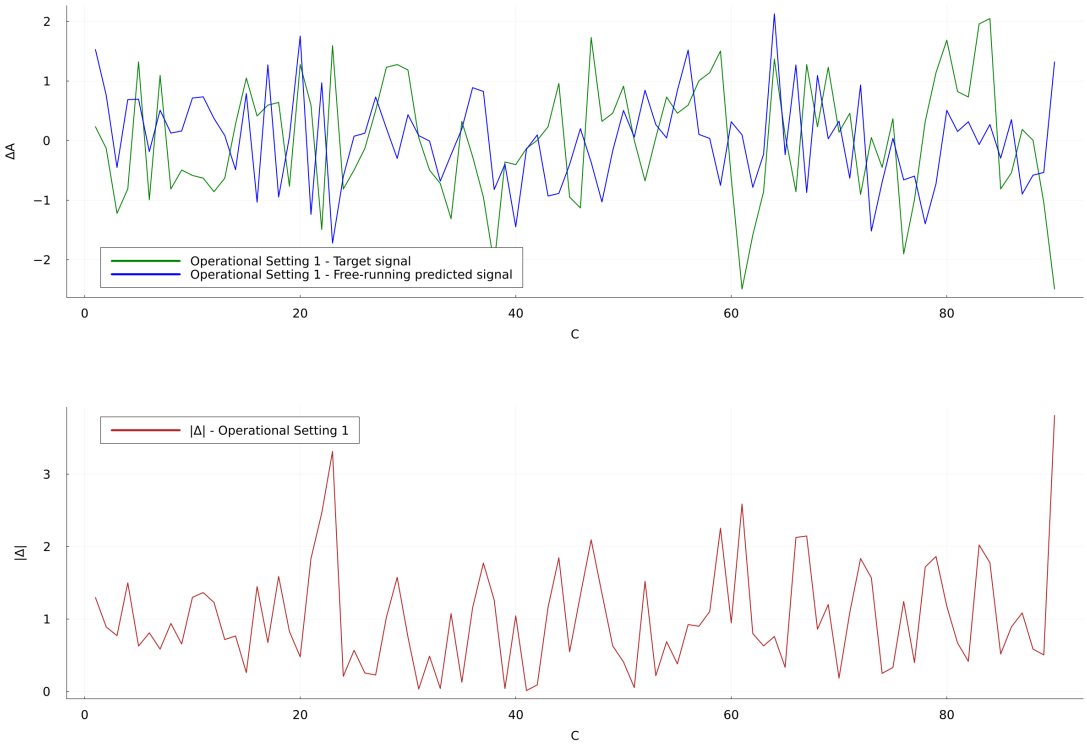


Figure A.11: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

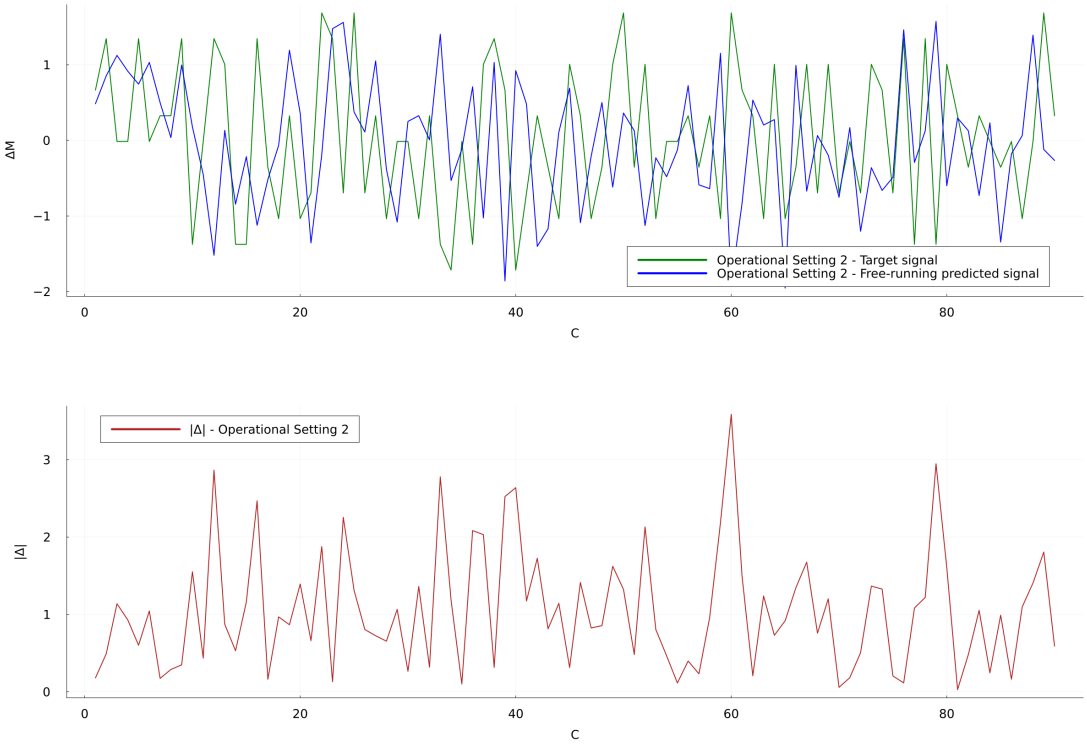


Figure A.12: Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

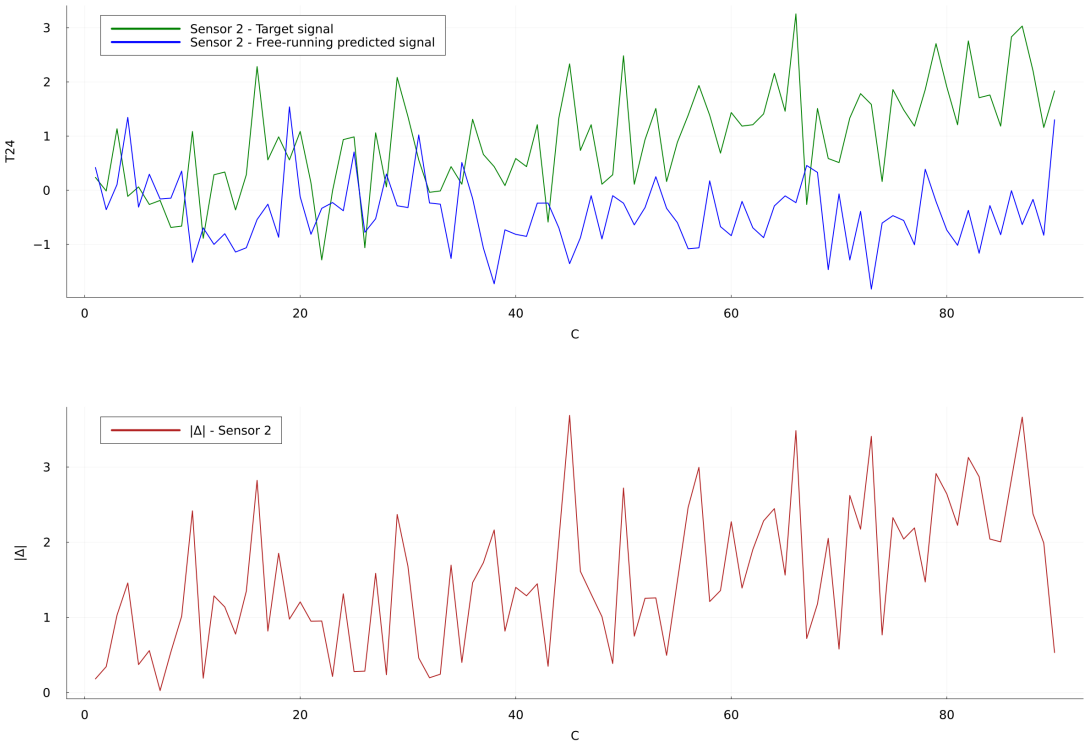


Figure A.13: Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

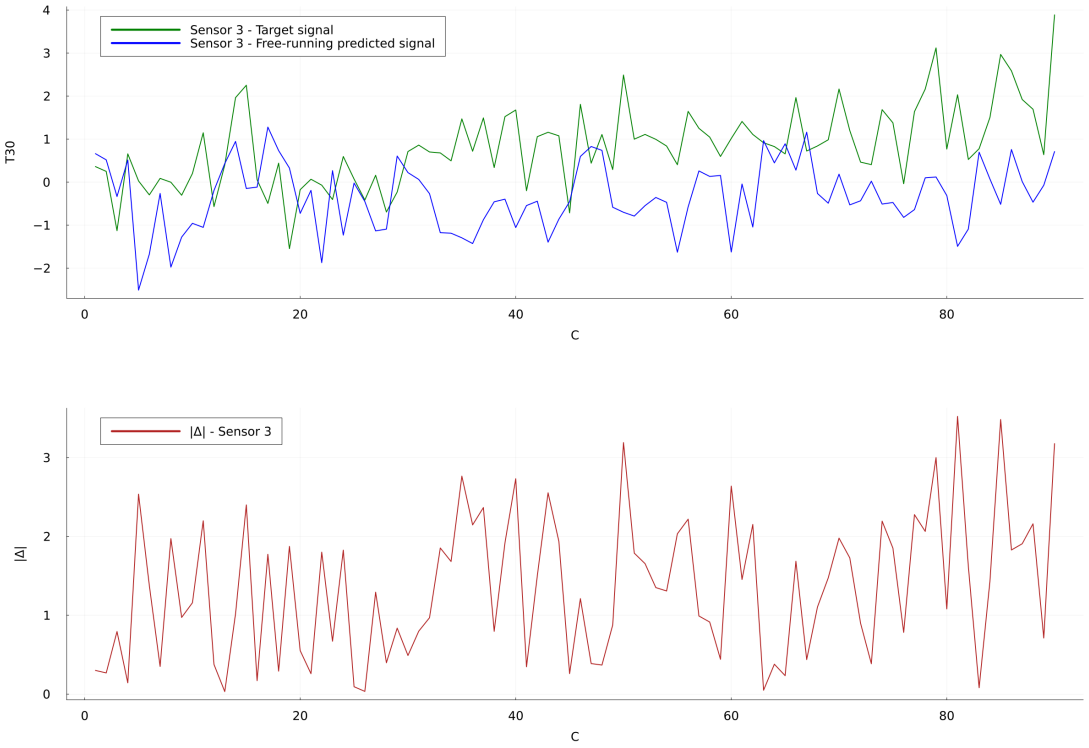


Figure A.14: Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

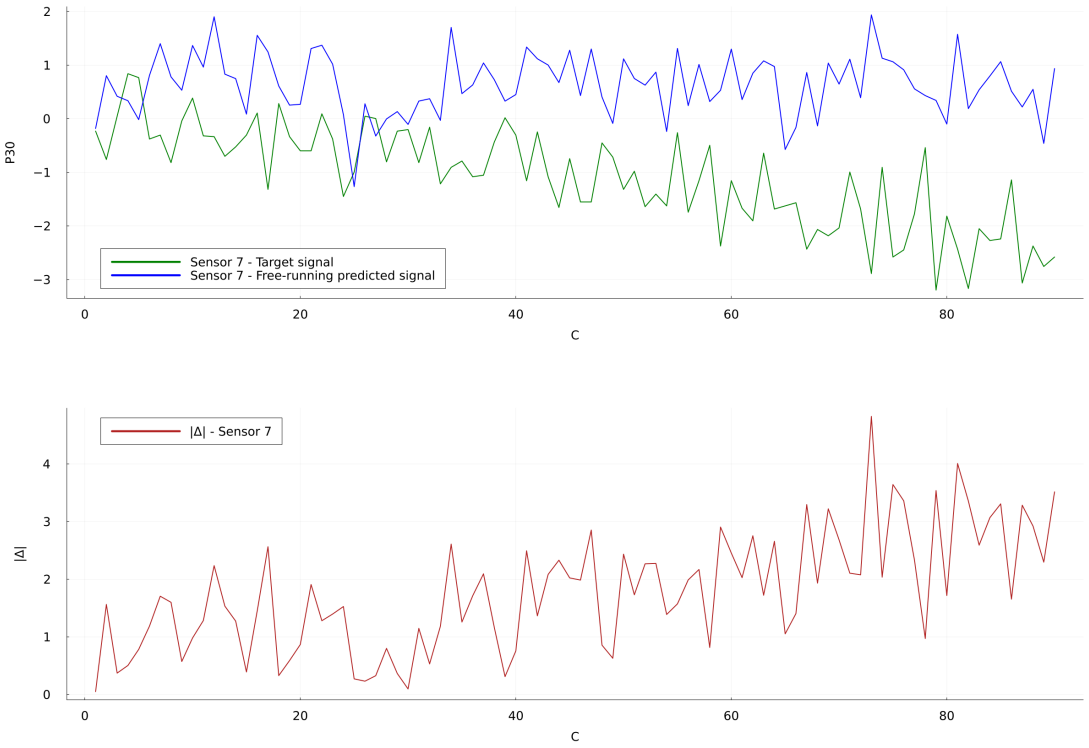


Figure A.15: Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

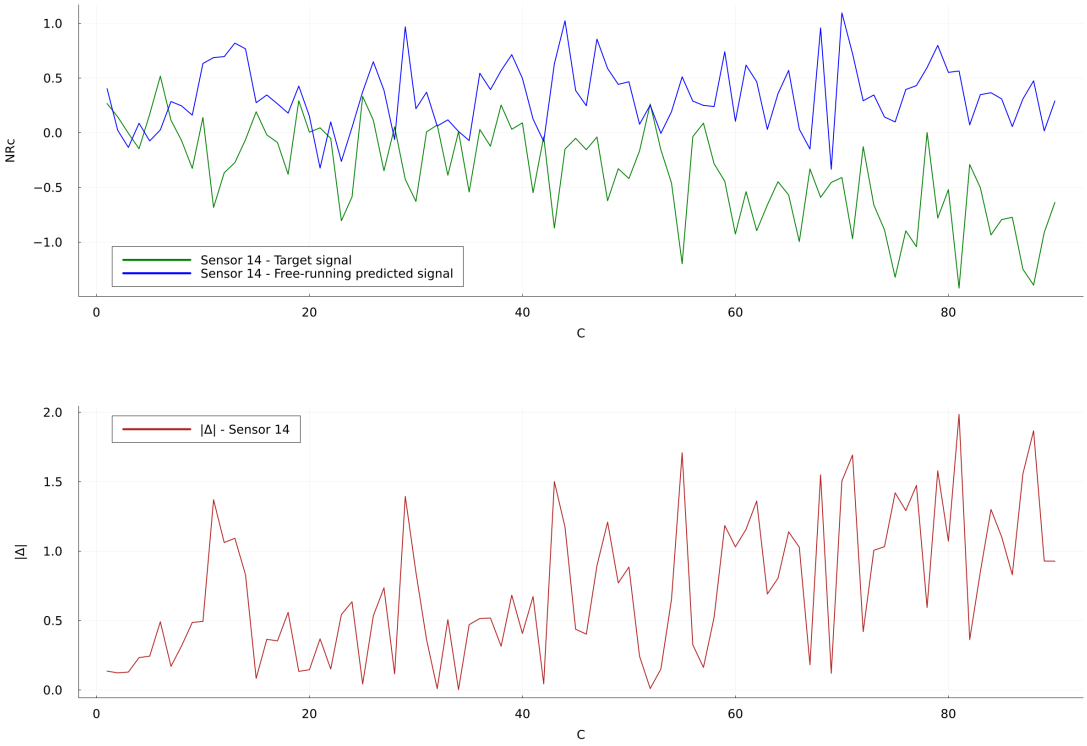


Figure A.16: Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the second set of hyperparameters

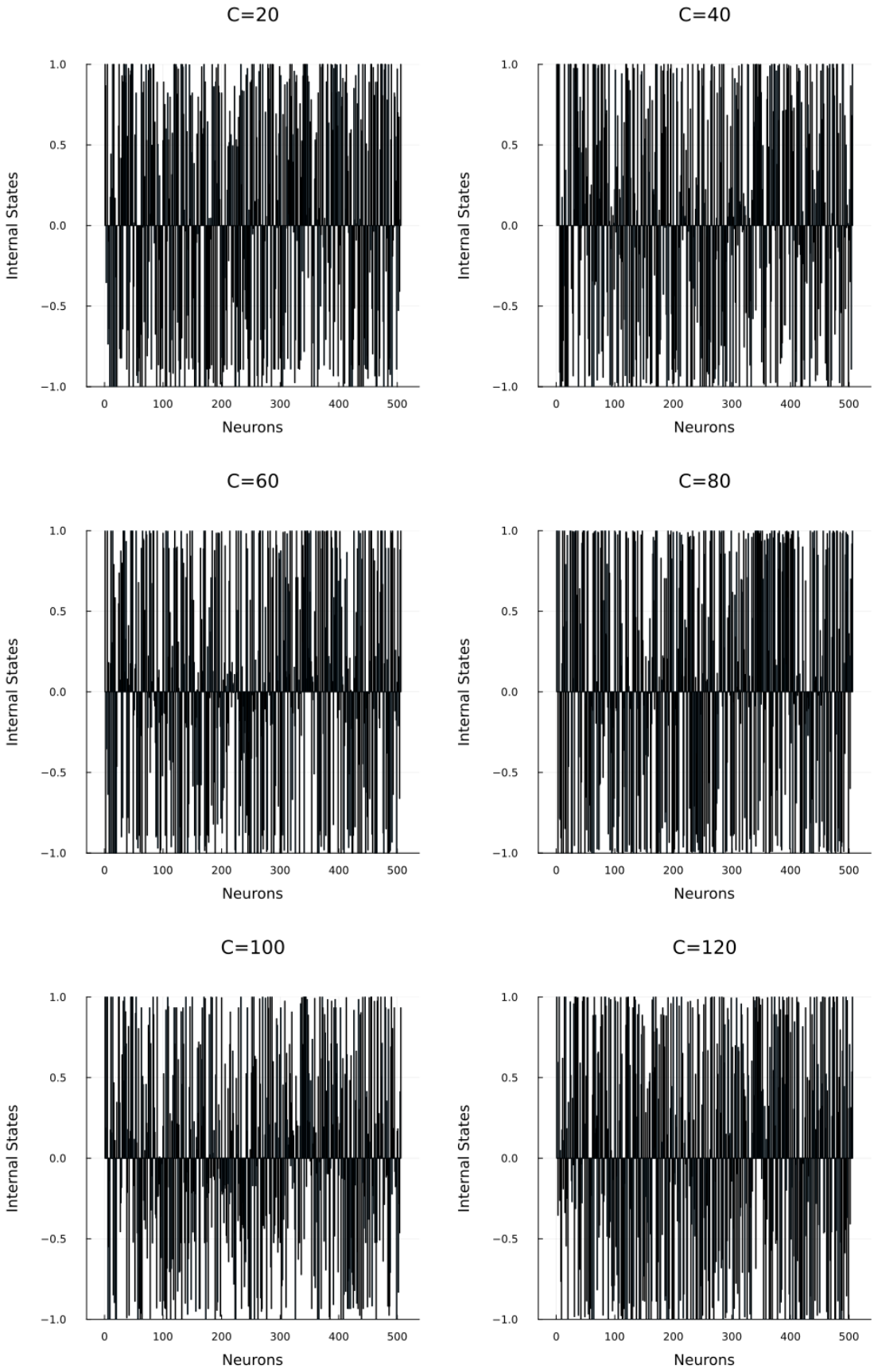


Figure A.17: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the second set of hyperparameters

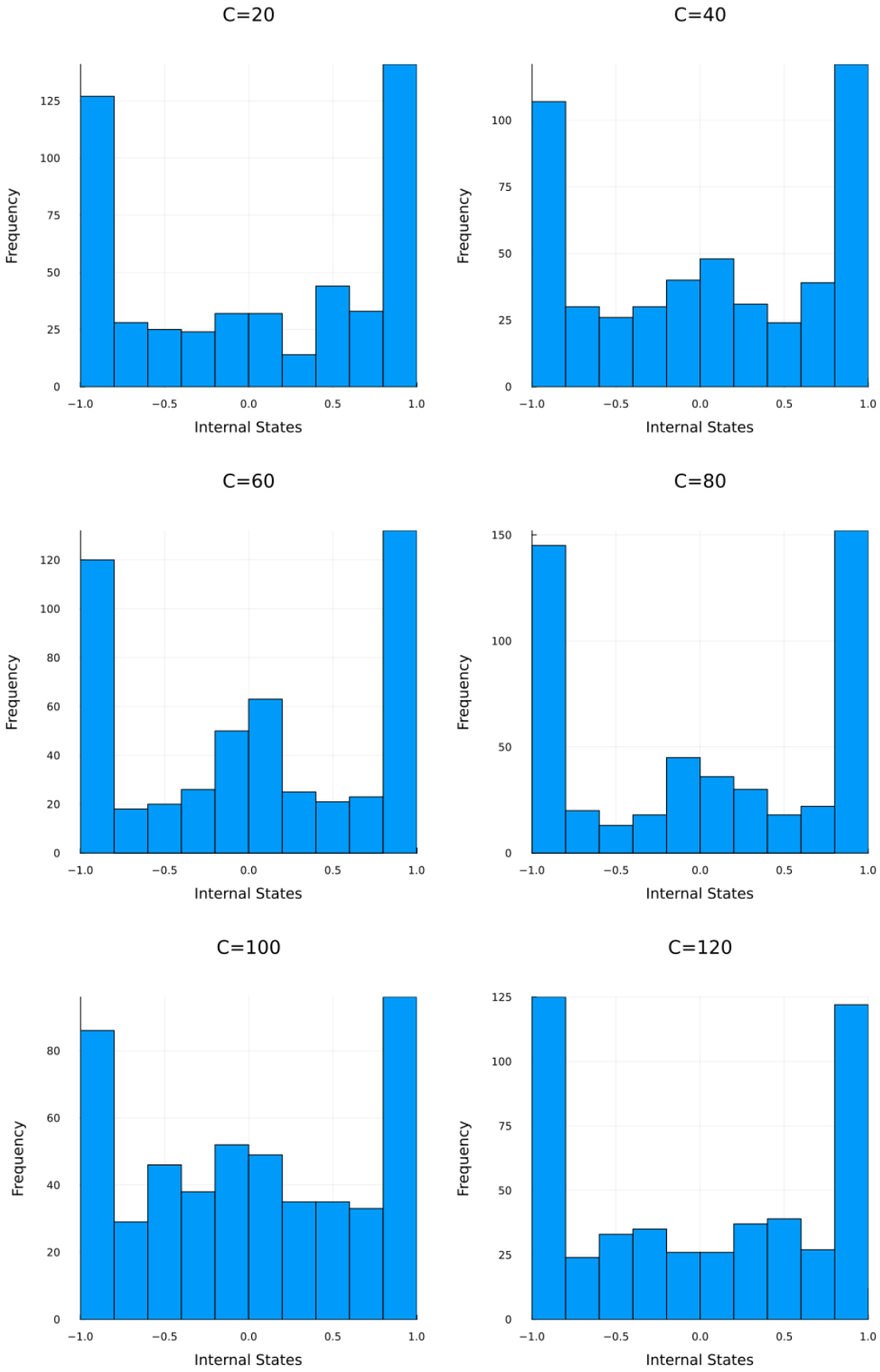


Figure A.18: Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the second set of hyperparameters

A.1.3 Third set of hyperparameters for the CMAPSS dataset

The third set of hyperparameters is the one under study on the publication by Riesgo et al. [39], the analysis hereinafter expands the discussion on the prediction quality and reservoir saturation. This third reservoir is the biggest in size and the generates a prediction with the second highest error, thus indicating that size is not the main hyperparameter acting on the average error. As for the leak rate, its higher value leads the reservoir to forget sooner the features of the dynamics but due to the limited training data available this drawback is mitigated. Input scaling factors on this set are extremely desirable as it gives the Operational Settings and the Sensors equal significance to the activation functions while quieting the unitary input, thus amplifying these signals without leading to saturation as in the first two sets. Memory is also affected by the spectral radius scaling factor, which also enforces the echo state property, and this set´s higher value offsets the leak rate, therefore improving memory slightly.

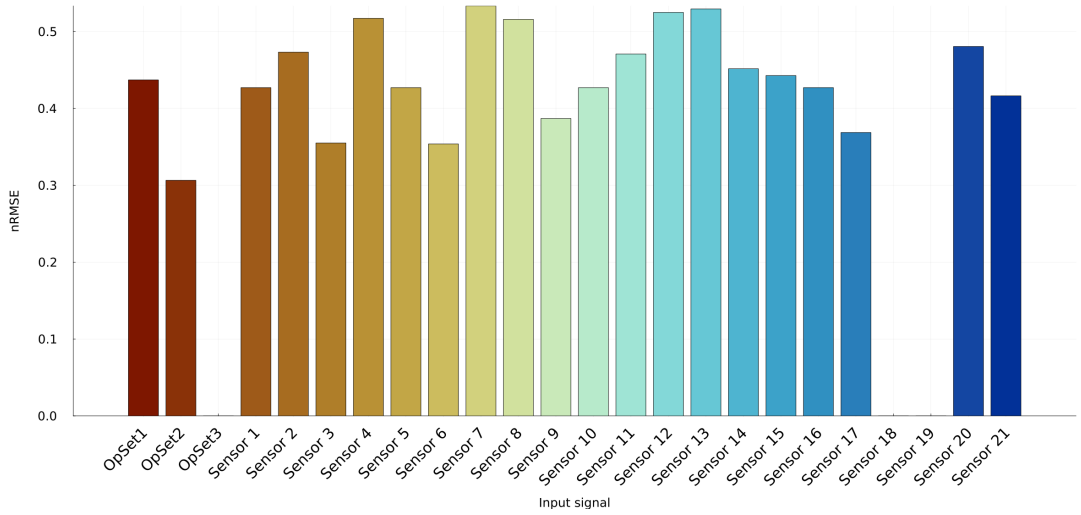


Figure A.19: nRMSE value of each of the signals for the prediction generated by the reservoir with the third set of hyperparameters.

Figure A.19 depicts the nRMSE, as in previous sets the predictions for Operational Setting 3 and Sensors 18 and 19 are accurate and generated a constant signal during the training phase while Sensors 1, 5, 10, 16 and 17 are not captured accurately. The error range is similar to the rest of the sets, all signals falling between the range of 0.3 and 0.5, and an average of 0.39 caused by signals such as Sensors 4, 7, 8, 12 and 13 passing the 0.5 error mark.

On Operational Setting 1, Figure A.20, the prediction achieves a higher nRMSE than the rest of the sets, 0.44, caused by the few matches with the original. The first 10 cycles of target initial trend are followed by the prediction, as well as the drop at 60 cycles. The absolute difference variation, $|\Delta|$, clearly indicates that while the free-running signal is not diverging from the original it is fluctuating widely throughout the testing phase, reaching at certain moments up to 3 in absolute difference. The first 10 cycles of target initial trend are followed by the prediction, as well as the drop at 60 cycles. Operational Setting 2 at A.21 has just 0.31 nRMSE, the lowest error for the five sets and the second best error value for any of the generated signals. The prediction itself shows the reservoir capacity to mimic the original, even with reduced training data. The $|\Delta|$ graph clearly show how the initial 20 cycles the target is followed closely while staying within the bounds of the original series, having similar features at the 50 cycles mark and between 60 and 80 cycles.

The prediction of Sensor 2, at Figure A.22, has an nRMSE of 0.47 clearly caused by the divergence of the prediction from the target at the early stages of the testing phase. During the 20 first cycles the free-running signal swings in the vicinity of the original, matching the drop at the end of the

range. Afterwards, as the $|\Delta|$ variation shows, the prediction undershoots the target until reaching its maximum at the end of the testing phase with 4 of absolute difference. As for Sensor 3, Figure A.23, the nRMSE is 0.36 for a prediction with the first 30 cycles matching the target, albeit with some excessive fluctuations, and a massive divergence at 60 cycles. The $|\Delta|$ variation up to 30 cycles is kept under 3, with even a matching drop at 20 cycles, but doubles due to the divergence.

Sensor 7, at Figure A.24, generates a prediction with an nRMSE of 0.53 due to, once again, divergence from the target. After the first 10 cycles, where no trends are matched with the original, the free-running signal overshoots steadily until the end of the testing phase. The absolute difference variation, $|\Delta|$, confirms this trend as it rises till 4 at the end of the phase. The prediction of Sensor 14 at Figure A.25, with an nRMSE of 0.45, shares some similarities with Sensor 7. It also overshoots the target, however it begins at 40 cycles and the divergence is not as severe. The $|\Delta|$ variation increases through the testing phase until reaching 2, but not as aggressively as Sensor 7 therefore the lesser error value.

The internal states for the third set, Figure A.26 and as a histogram at Figure A.27, show a reservoir on the verge of saturation even though it has the highest number of neurons and balanced input scaling factors. During the training phase nearly half of the neurons are at fringe values, becoming saturated at 80 cycles with three quartes of the neurons at fringe values and afterwards recovering to more evenly distributed internal states during the latter stages of the phase. These states still retain separability but can, as seen at 80 cycles, dip into saturation and take a hit in the prediction. The predictions for the third set are stable and feature-rich and can improve if tweaking the input scaling factors, as size is not the defining hyperparameter, by reducing the influence of the Sensor signals as in the fourth set.

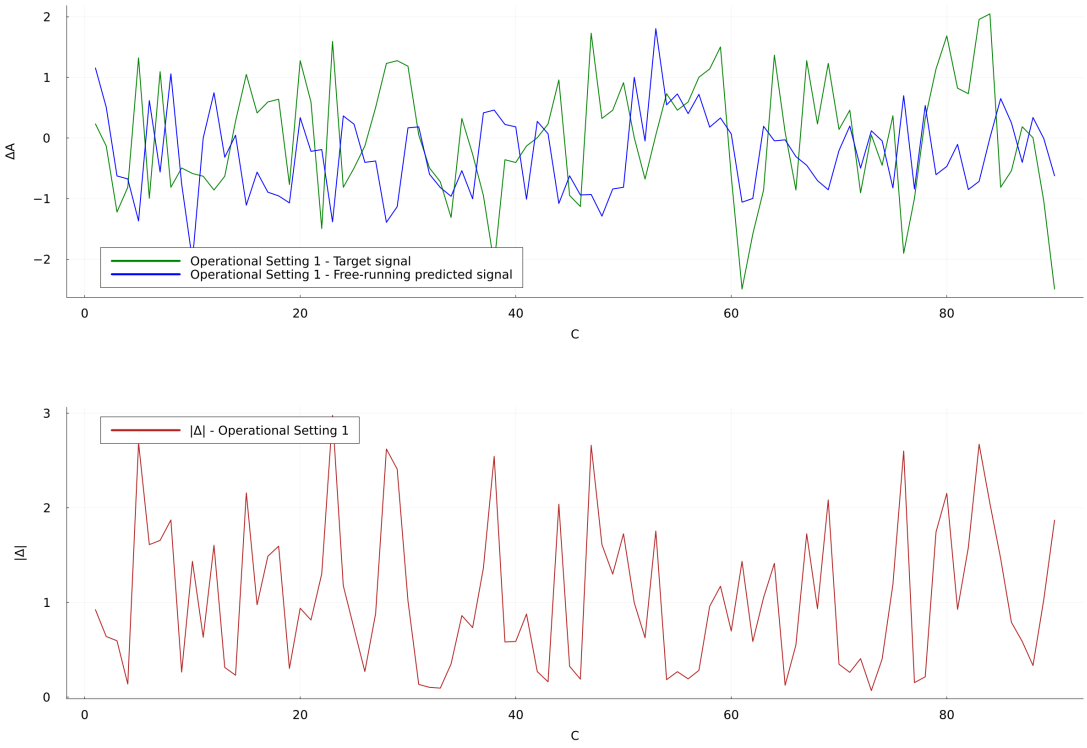


Figure A.20: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

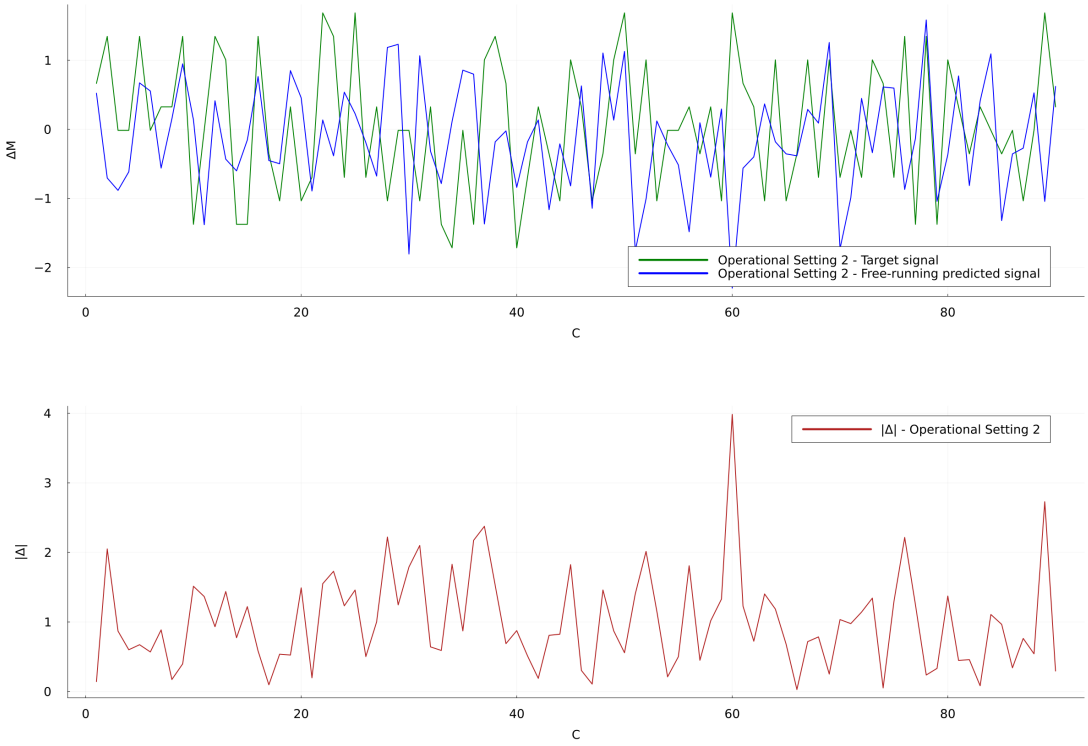


Figure A.21: Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

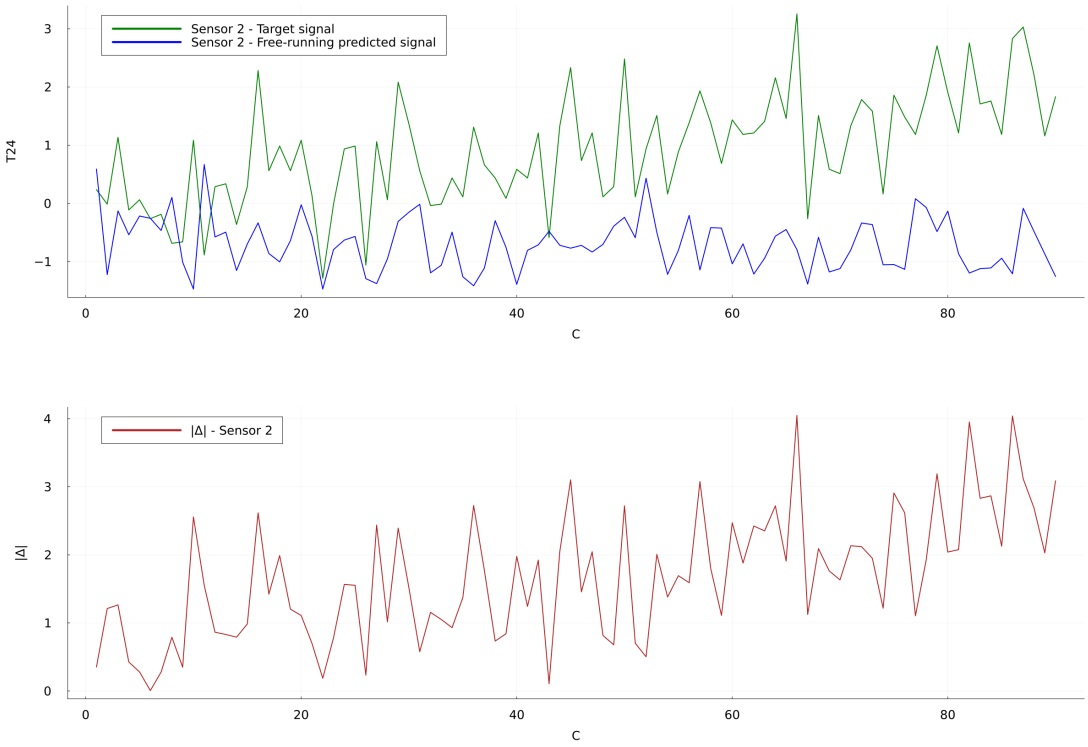


Figure A.22: Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

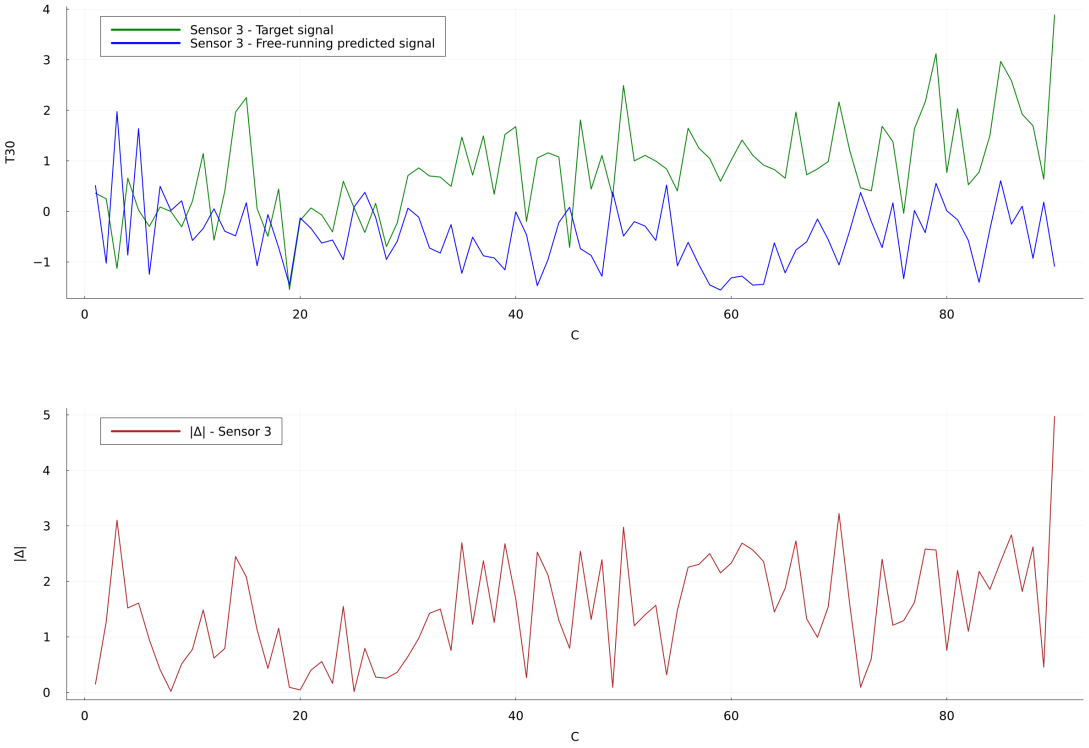


Figure A.23: Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

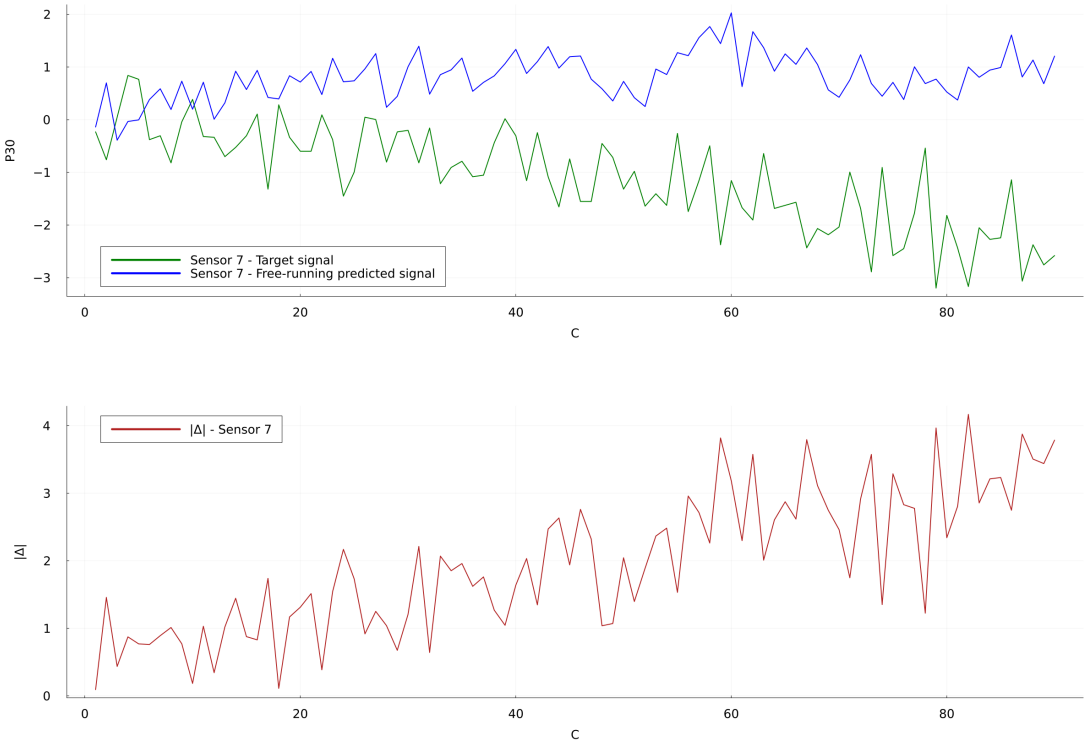


Figure A.24: Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

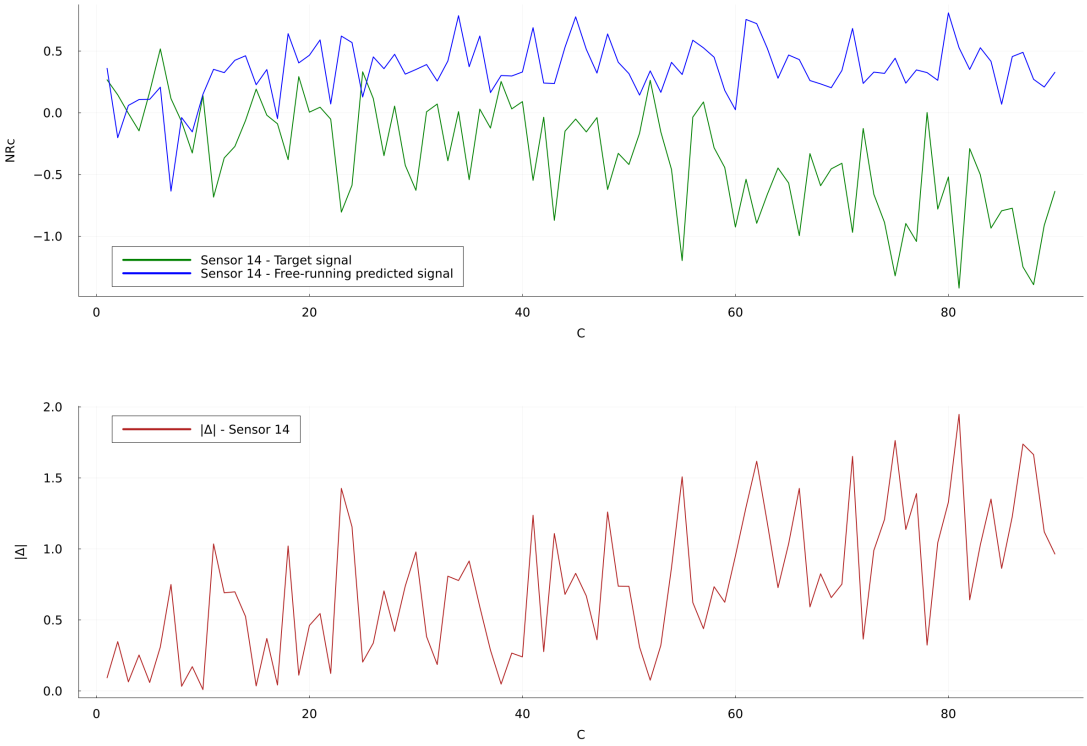


Figure A.25: Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the third set of hyperparameters

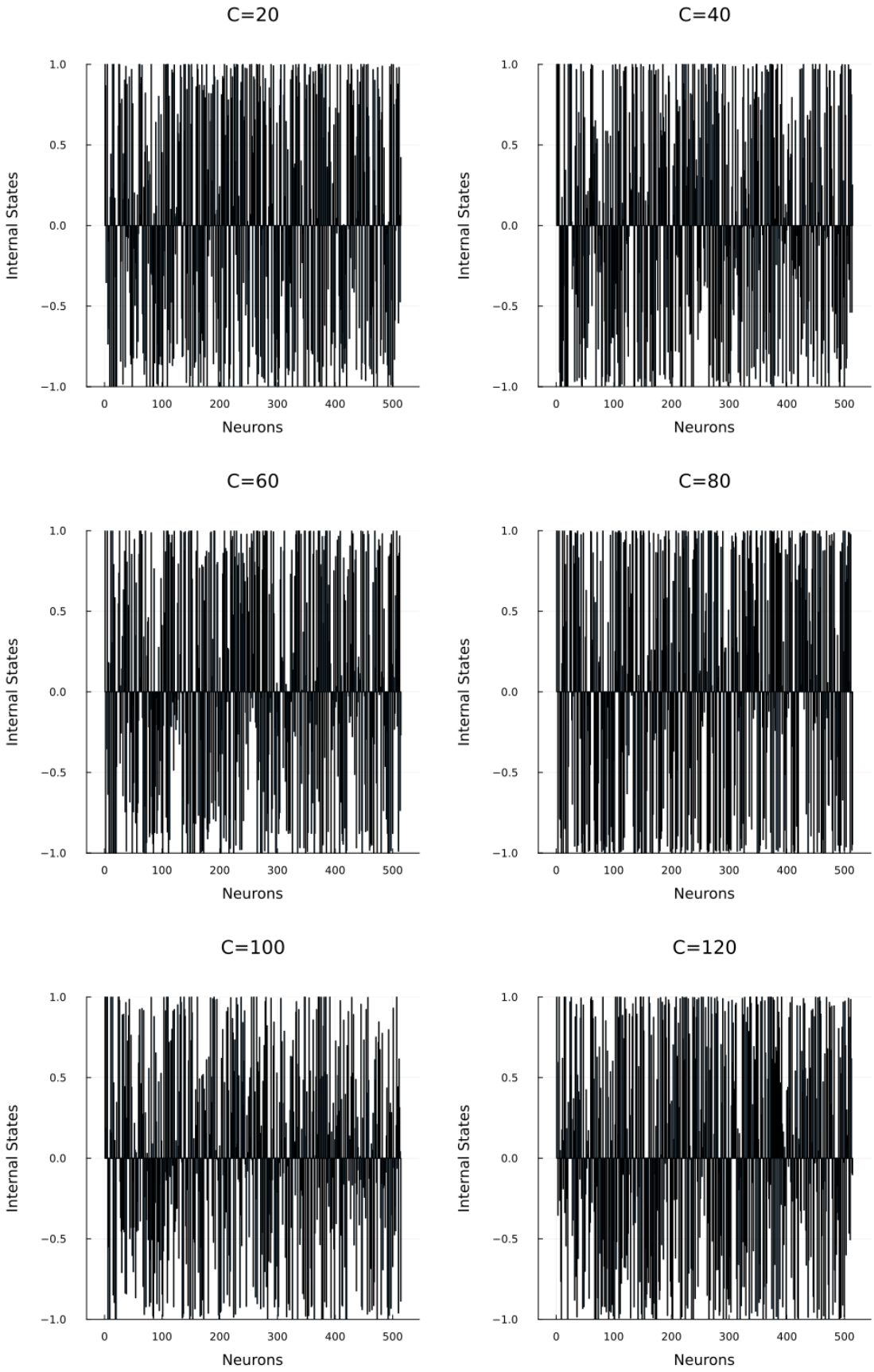


Figure A.26: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the third set of hyperparameters

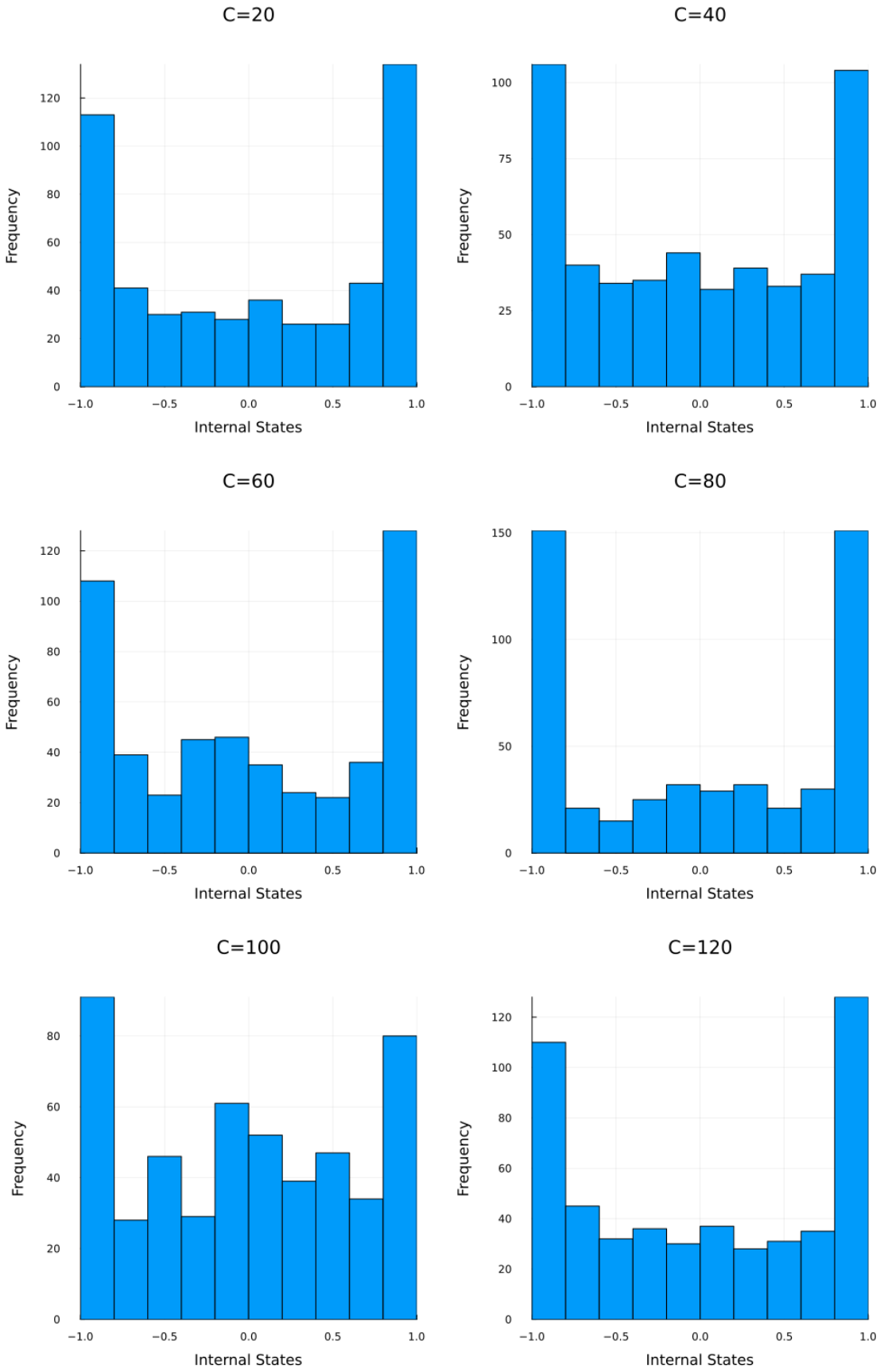


Figure A.27: Distribution of the reservoir internal states various moments of training cycle C for the CMAPSS time series for the third set of hyperparameters

A.1.4 Fifth set of hyperparameters for the CMAPSS dataset

The fifth set of hyperparameters under study has just 61 neurons more than the fourth set, Subsection 4.2.1, and the smallest short-term memory (highest leak rate), along with the worst nRMSE value of the lot. With 166 neurons this reservoir is slightly bigger than the case study and shows that not because more computational power is expended the results will be better. Short-term memory is theoretically the lowest, as the leak rate is the highest of the selected configurations while the spectral radius scaling factor is the highest, which will induce quick dynamics in the reservoir which retain little information from the previous inputs even with the spectral radius attempting to improve the memory. The input scaling factors are the same order of magnitude for the Operational Settings and the Sensors, giving slightly higher priority to the second group of signals, while the unitary value is attenuated. Therefore, saturation should not be a problem in this reservoir.

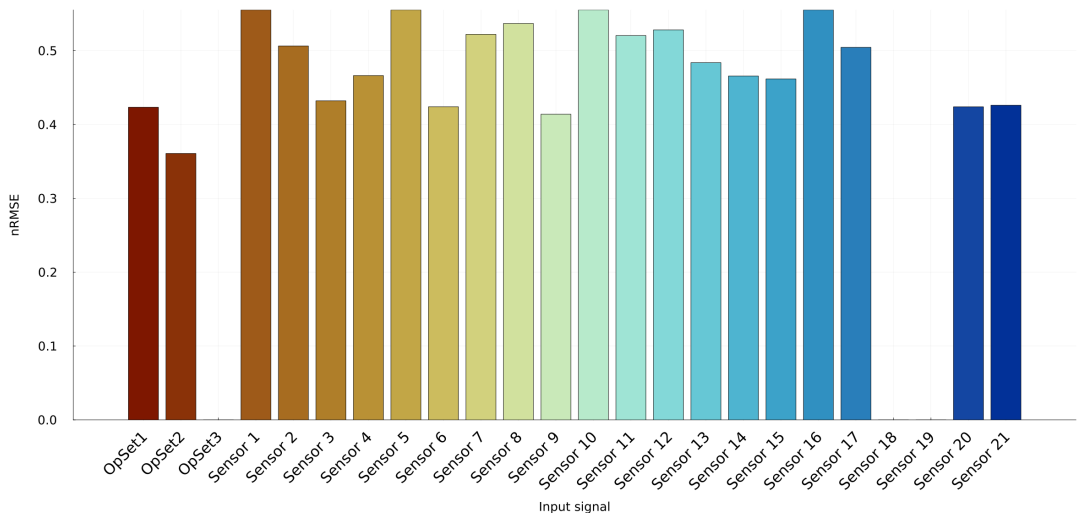


Figure A.28: RMSE value of each of the signals for the prediction generated by the reservoir with the fifth set of hyperparameters.

The nRMSE of each of the signals (Figure A.28) shows a global prediction where most values are at the 0.3 to 0.6 range, slightly higher than the other four sets. Once again there are null errors for Operational Setting 3 and Signals 18 and 19, but Signals 1, 5, 10, 16 and 17 present an error value close to 0.6 which is a clear indication that the reservoir has once again not interpreted correctly the dynamics of the turbofan engine. The average nRMSE value is 0.42 as most signals hold an error value in the vicinity of 0.4, leading to the highest error average of the five sets.

Operational Setting 1, depicted in Figure A.29, is predicted with an nRMSE value of 0.39 and a loose performance at predicting the trends of the original. The $|\Delta|$ variation clearly indicates that only during certain cycles the generated signal approaches the target. There are few remarkable ranges, the first being the one between 10 and 15 where the drop is accurately predicted and the other from 40 to 60 cycles in which the predicted rises and drops are at a slight phaseout with the target. Moreover the prediction of Operational Setting 2, shown in Figure A.30, presents the smallest non-zero nRMSE of the whole set: 0.34; while representing accurately certain trends from the original. The $|\Delta|$ variation is fairly steady an clearly signals a lack of divergence from the target, at some moments the variation reaches 3 but this is repeated throughout the testing phase. This prediction is feature-rich and within the target range but does not succesfully mimic its dynamics.

Sensor 2, as seen in Figure A.31, has been predicted with an nRMSE of 0.4 and follows the target up to 10 cycles with moderate success while not fully predicting any particular peak or drop except

one, around 9 cycles, and then steadily undershoots. The absolute difference variation, $|\Delta|$, is homogeneous during the testing phase as the divergence increases until reaching a 4 $|\Delta|$ at the end of the testing phase. In Figure A.32, Sensor 3, the prediction is once again undershooting from the original most of the time without mirroring its trends in any specific point. The nRMSE is 0.38 and the absolute difference variation reaches values of up to 4 towards the end of the phase.

As for the last two examples, Sensor 7 in Figure A.33 and Sensor 14 in Figure A.34, the generated prediction for both of them does not match the original and both overshoot the target since the initial cycles. Sensor 7 is the worst of the two, as the absolute difference is higher, with an increasing divergence reaching up to 4. The prediction of Sensor 14 has similar features, at an nRMSE of 0.42 whereas the one for Sensor 7 is 36, which can be clearly seen as during most of the testing phase the prediction overshoots the original in its downward trend.

When checking the internal states of the reservoir during training, Figure A.35 and the histogram at Figure A.36, the saturation is near in all six of the examples as half of the neurons are at fringe values. During the first four snapshots of training, cycles 20 to 80, the reservoir seems to be saturated as there is no separability. At the latest phases of training the internal states become unsaturated, allowing for the full activation function range of values. As in the third set of hyperparameters, the reservoir size has increased when compared to the fourth, and yet it has a more saturated reservoir. With the key laying on the input scaling factors, saturation is avoided when the twenty-one Sensor signals have a lower scaling factor than the Operational Settings.

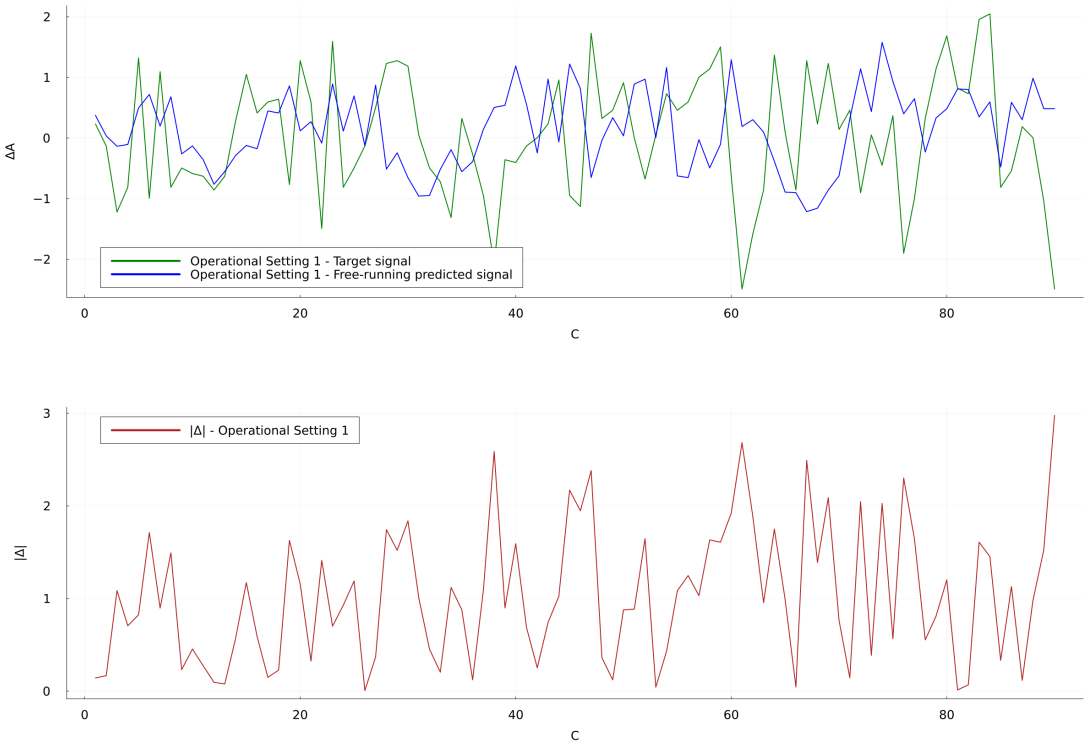


Figure A.29: Operational setting 1 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

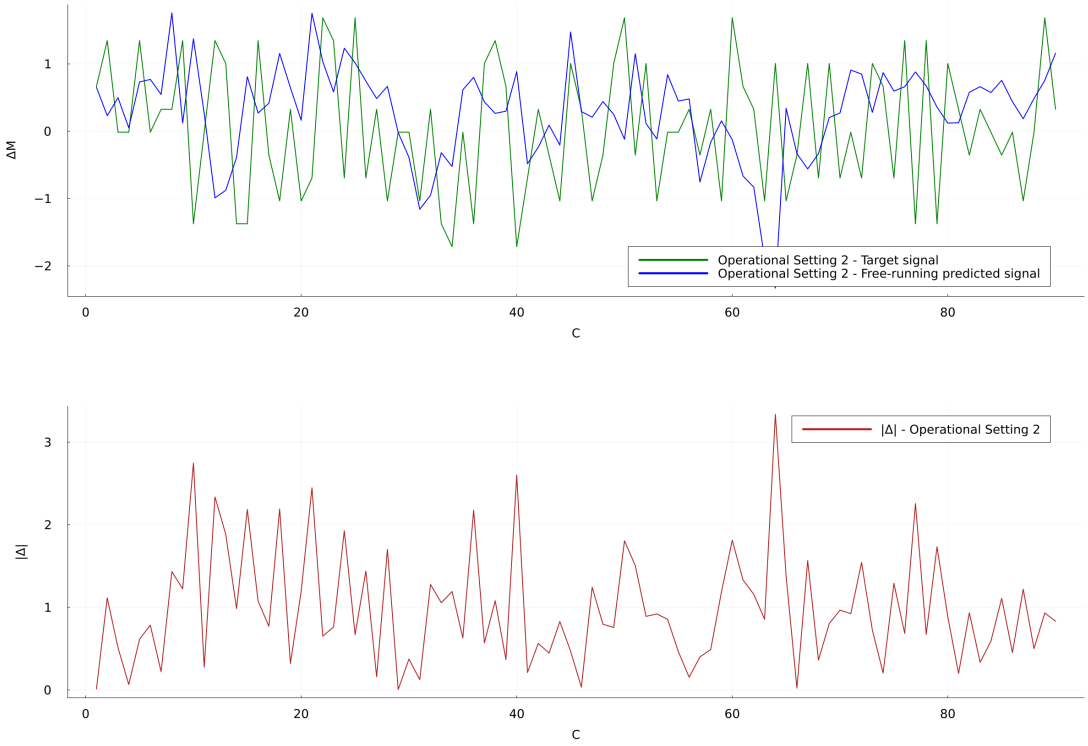


Figure A.30: Operational setting 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

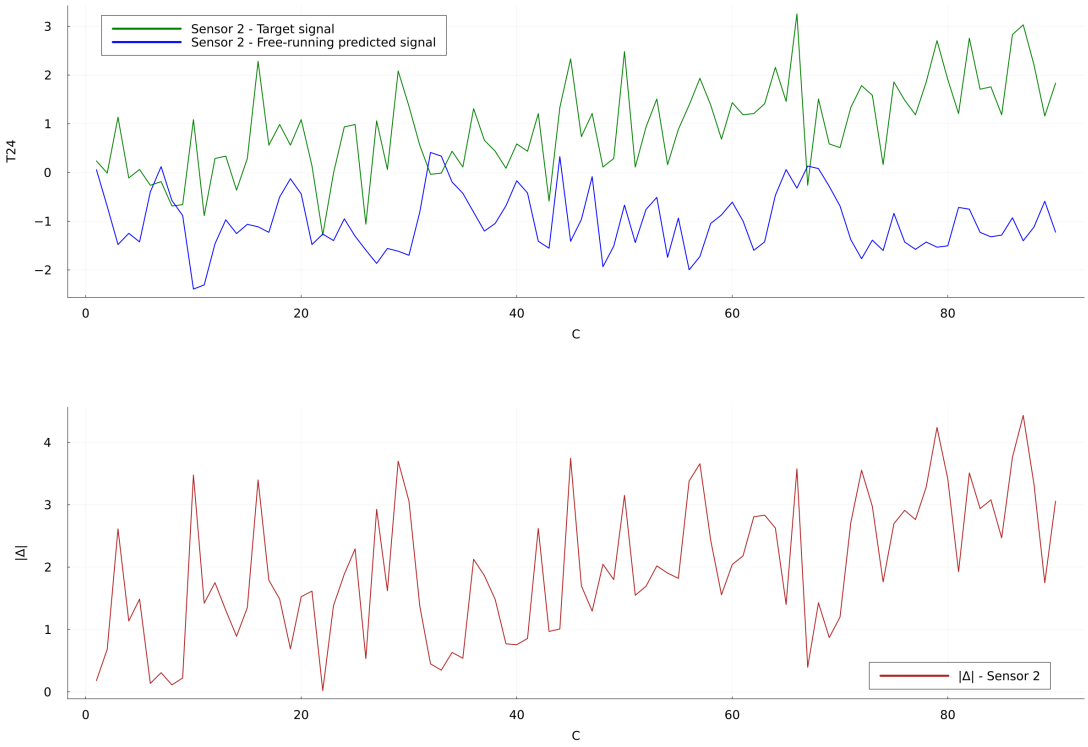


Figure A.31: Sensor 2 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

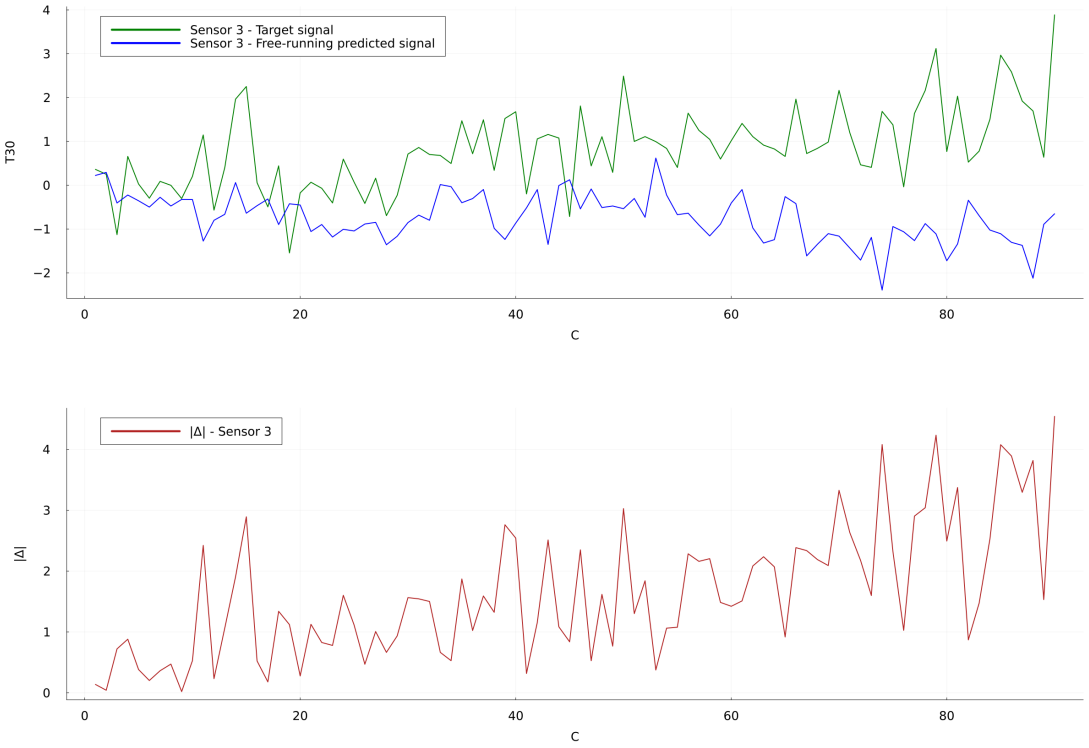


Figure A.32: Sensor 3 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

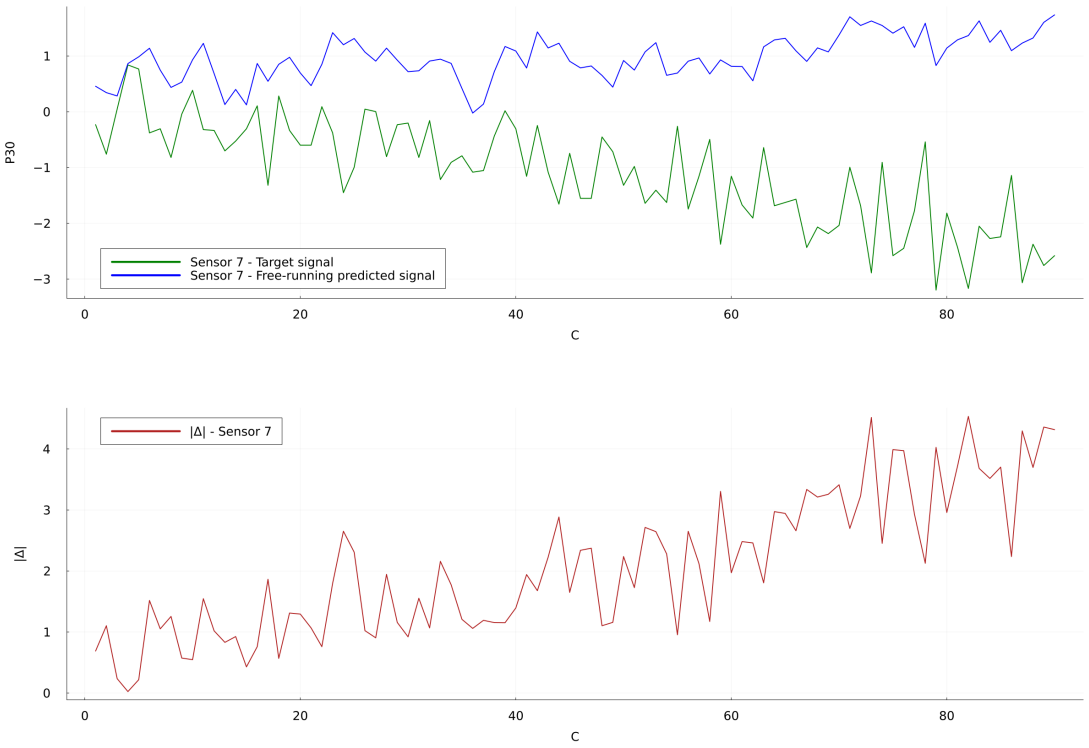


Figure A.33: Sensor 7 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

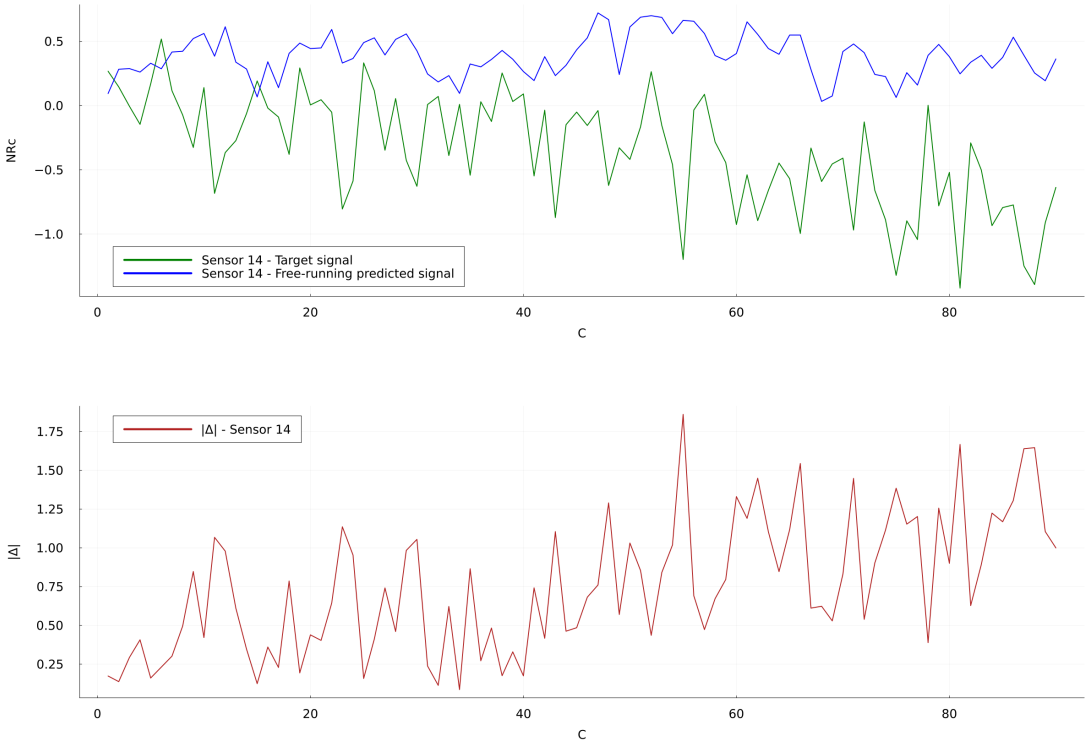


Figure A.34: Sensor 14 CMAPSS time series along with the predicted signal (above) and the absolute difference variation in time (below) for the reservoir with the fifth set of hyperparameters

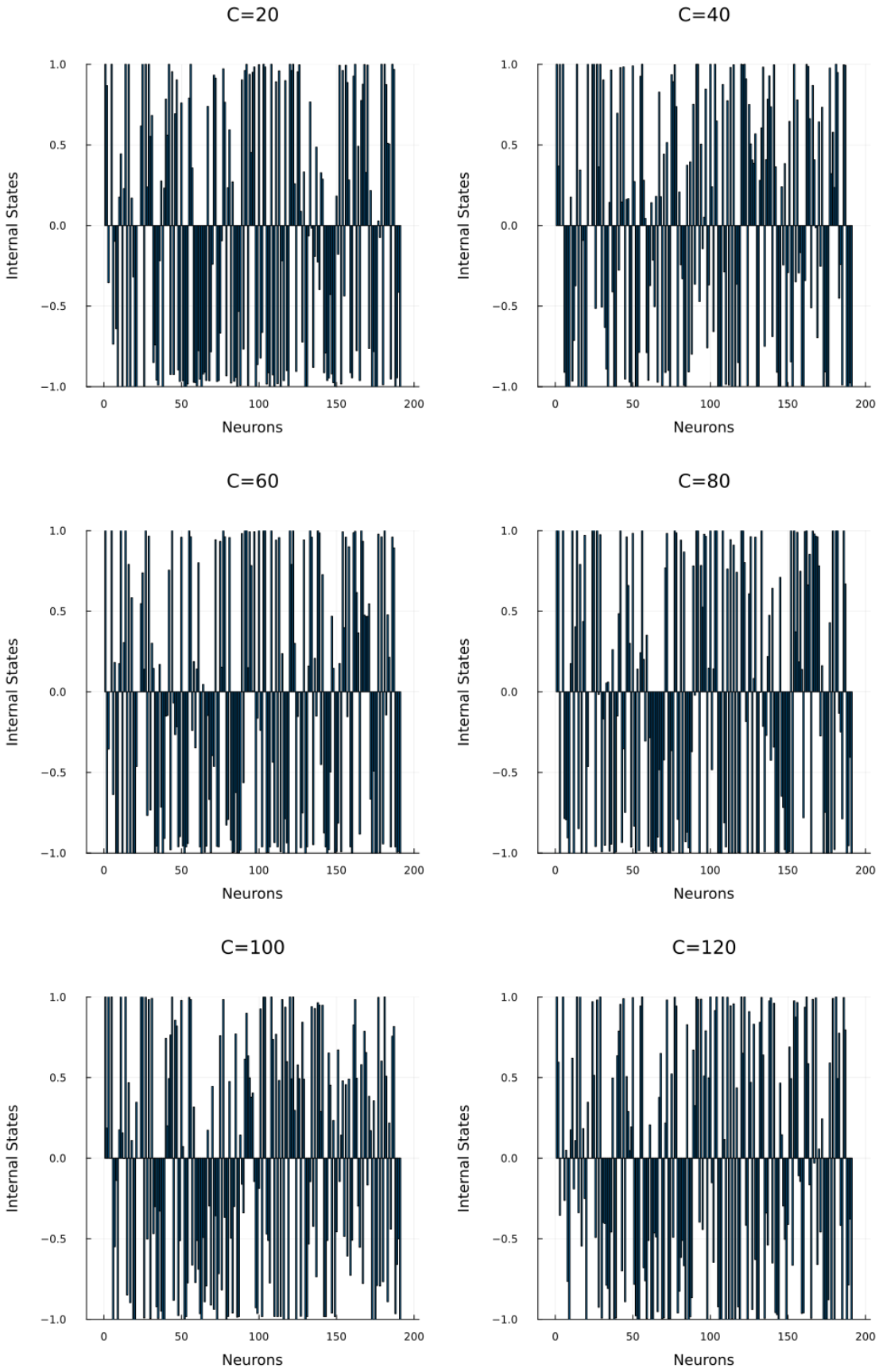


Figure A.35: Reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fifth set of hyperparameters

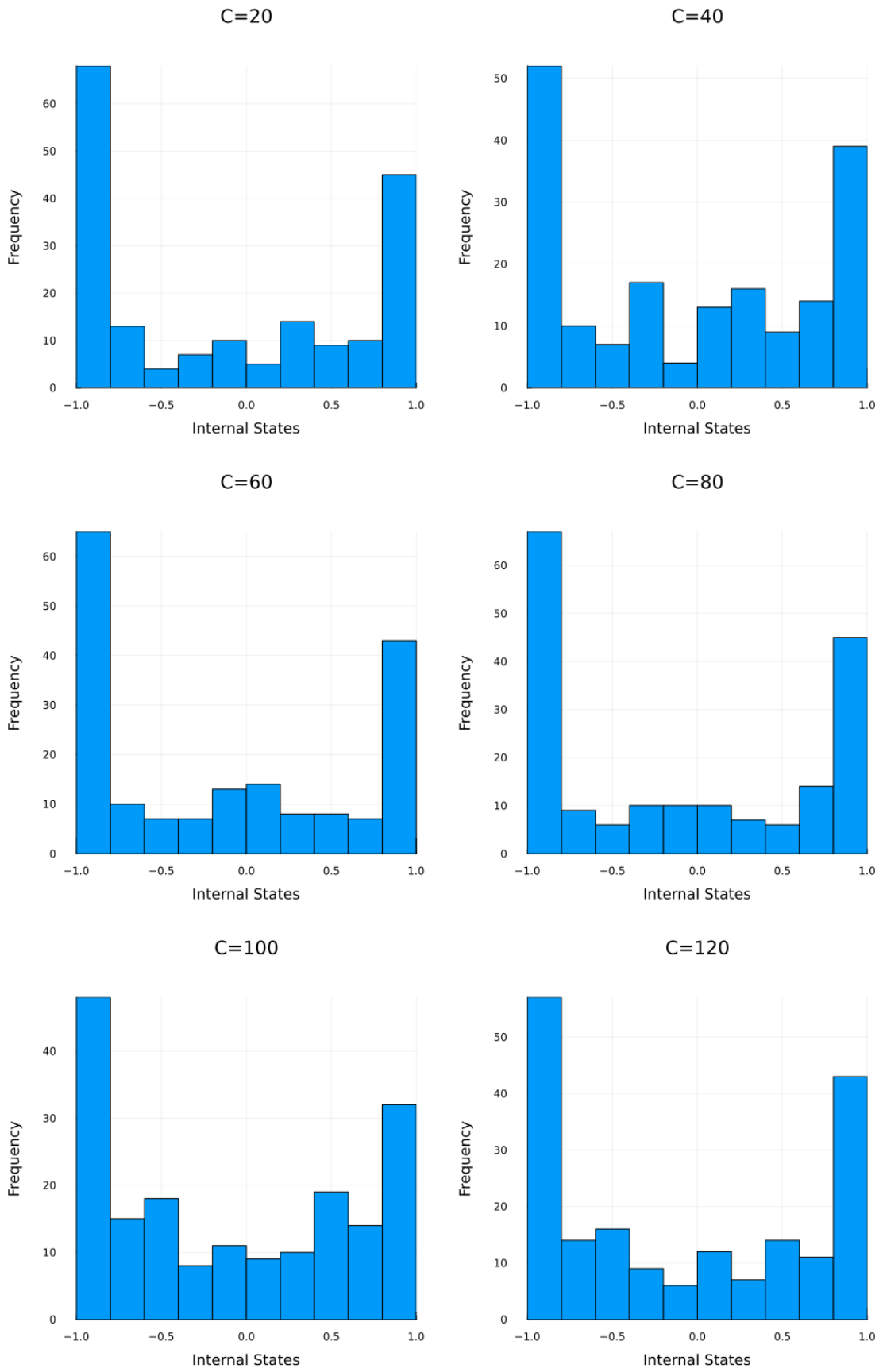


Figure A.36: Distribution of the reservoir internal states at various moments of training cycle C for the CMAPSS time series for the fifth set of hyperparameters

A.2 Punta Cana dataset

A.2.1 First set of hyperparameters for the PC dataset

The first set of hyperparameters has the highest nRMSE at 0.51 along with a high amount of neurons, the second highest, which is a computationally expensive reservoir to run compared to the third set. The reservoir sparsity for the first case is the only outlier for the sets on the PC dataset, with it being at the middle of the optimization range. The first key aspect to focus on is the short-term memory, which is affected by the leak rate and the spectral radius scaling factor. As for this set the leak rate is at 0.38, the lowest, with an spectral radius scaling of 0.80, close to the optimization upper bound, which leads to slow dynamics in the internal states with a lot of information from the previous inputs resonating in the reservoir. An expanded short-term memory is, as Lukoševičius [29] claims, fundamental for predicting such a complicated time series as a lot of the involved dynamics take a long time to repeat.

Once again a potential downside to this optimization is the resulting input scaling factors, as seen on the previous datasets having them so relatively close to each other when compared to the activation function extreme values will for certain cause saturation. Even in this case, with the factor for the power input being higher than those from the unitary or δ inputs, the separability condition in the reservoir is not going to be achieved as it is not assisting the neurons in selecting the most important input signal, but flooding the activation functions with unnecessary information. The regularization coefficient is once more not a defining factor when studying the results, as its main purpose is to avoid overfitting and requires further prediction than what has been done in this thesis.

Figure A.37 shows the prediction generated for the first set of hyperparameters with a CMA filter of $k = 100$ and the results can look bleak when focusing only at the graph for the entire testing phase, but the initial prediction shows great potential. During the testing phase the prediction starts off with an erroneous trend, underscoring the smoothed out power consumption of the resort slightly, fortunately it quickly catches on and from 200 time steps onward the upward trend is followed closely. This close match continuous for 100 time steps more before the instability kicks in and the prediction looses track of the original, entering a periodic fluctuation. This behaviour goes along the remaining testing phase, with a change in periodicity and amplitude at around 3000 time steps with no resemblance to the target. As the absolute difference, $|\Delta|$, indicates the prediction initially follows the original with little difference, which quickly increases once it becomes unstable.

Comparing it to the prediction without any filtering in Figure A.38, with a nRMSE of 0.35, certain elements are amiss. For instance, on the unfiltered prediction there is no sudden change as can be seen on the filtered one, but there is no time during the testing phase in which it actually resembles the trends of the original. The period from the generated signal looks like it could coincide with the target, but most of the time misses completely. The filtered prediction, even with its evident shortfalls is a better result than the one generated with the unfiltered input.

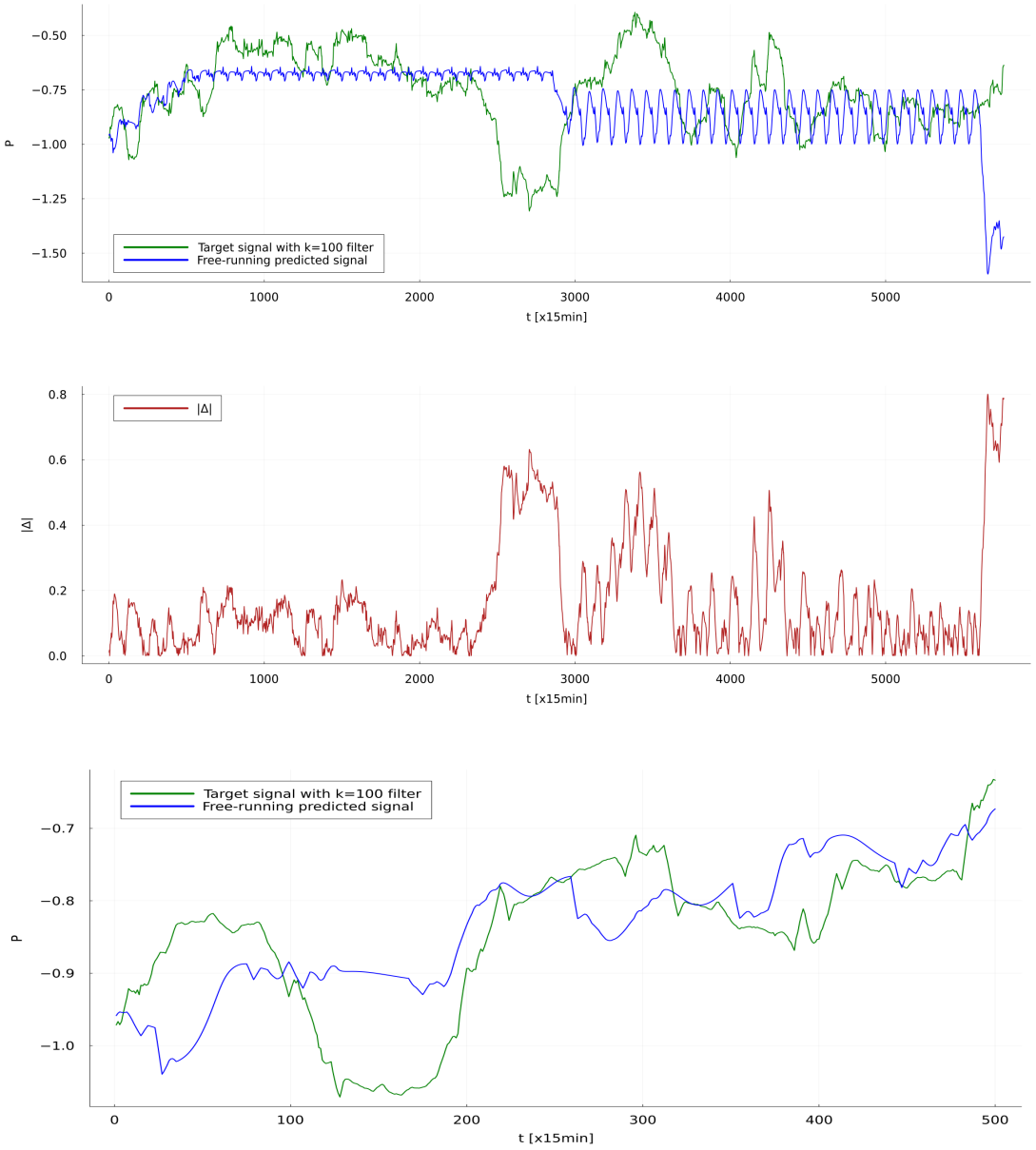


Figure A.37: PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the first set of hyperparameters

The predictions of δ_1, δ_{10} and δ_{100} signals depicted in Figure A.39 are a clear example that even when attempting to give the reservoir more information on the input it may not be enough to assimilate the complicated dynamics, as rather than feeding it new information that may bridge the gaps during training it is the same signal but on a different format. It may be helpful but only as an increase in performance not as a solution when the original data lacks contextual support. Take, for instance, the prediction of the δ_{100} signal with a behaviour similar to the one from the power signal and the same abrupt changes and still the prediction misses completely its target. The first 500 time steps of the auxiliary signals at Figure A.40 show that even if the predictions can follow the average value of the original the fluctuations are near impossible to mirror. Improving the prediction performance on these aggregated signals, added to help the reservoir in understanding the underlying dynamics is fundamental to increase the range before the generated signal changes abruptly.

The reservoir internal states are saturated as evident in Figure A.41 and the histogram at Figure

A.42, showing most neurons during the training phase at activation function fringe values to the point where most of the snapshots are unrecognizable between each other. This poses a serious problem, as separability is not present and impacts heavily on the ability of the output layer to interpret the reservoir dynamics. Moreover looking at the histogram the internal states are mostly at the extremes, with a reservoir of close to a 1000 neurons almost have are valued -1, while those at 1 vary depending on the time step, but make up mostly of whatever remains in the reservoir. Values in the range between them are almost non existent, close to none for instance at $t = 6000 \times 15min$. Once again, the reservoir size is not what caused the situation but the input scaling factors, driving the internal states to saturation and preventing from the output layer from being trained correctly.

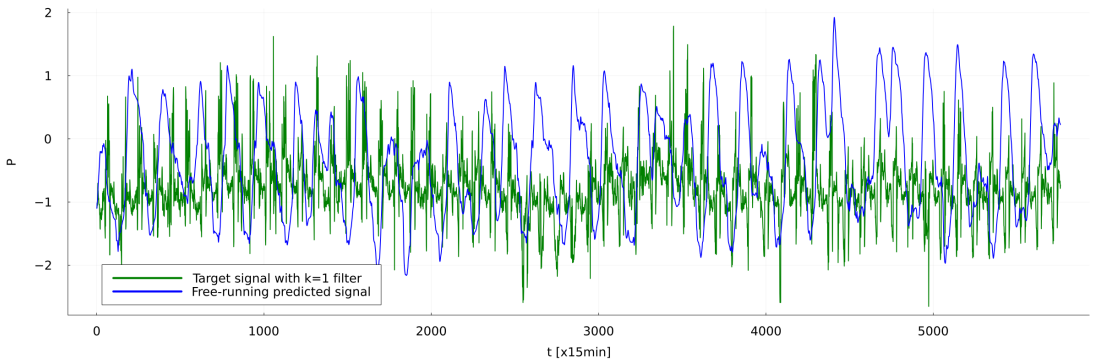


Figure A.38: PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the first set of hyperparameters

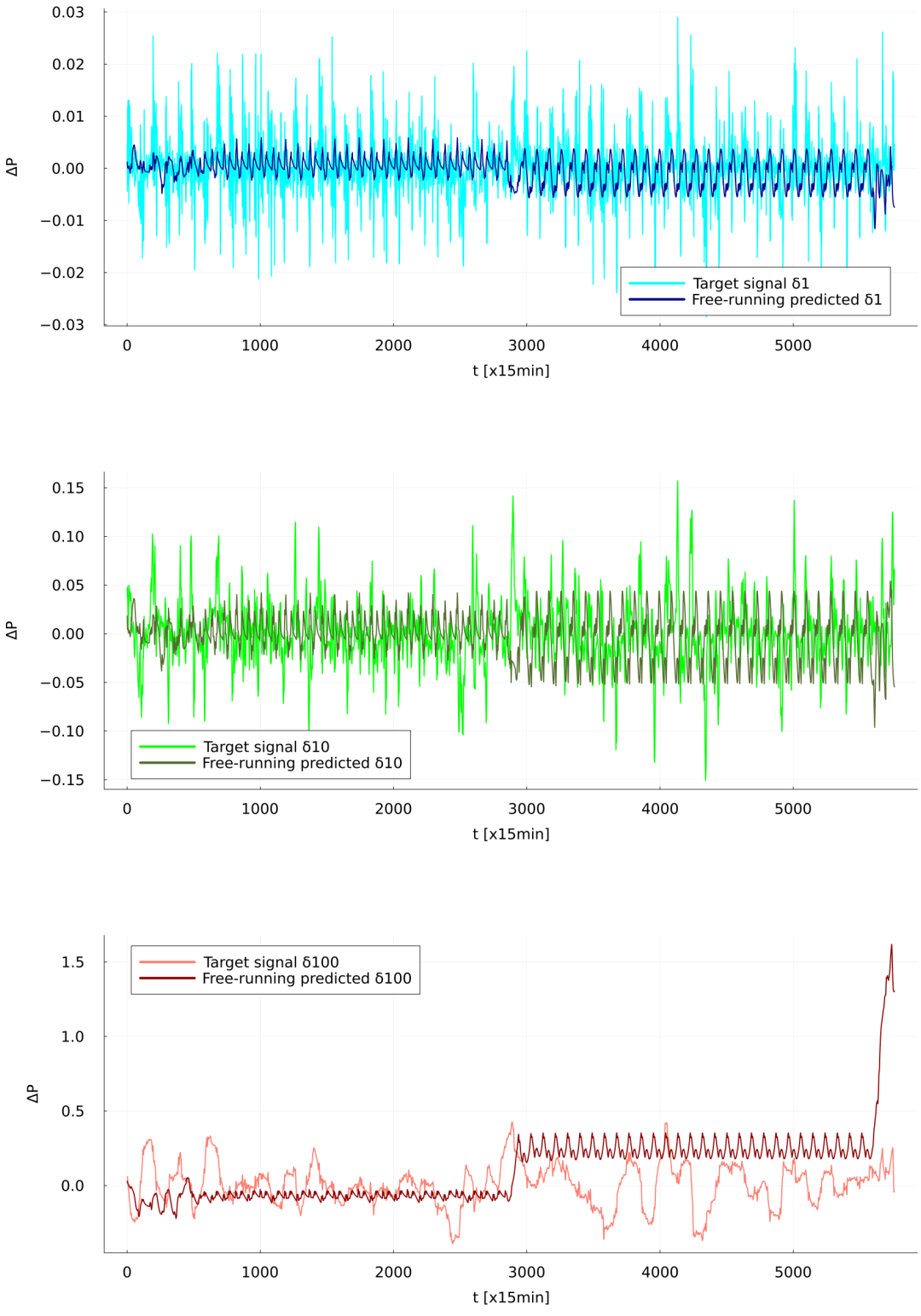


Figure A.39: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the first set of hyperparameters



Figure A.40: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the first set of hyperparameters

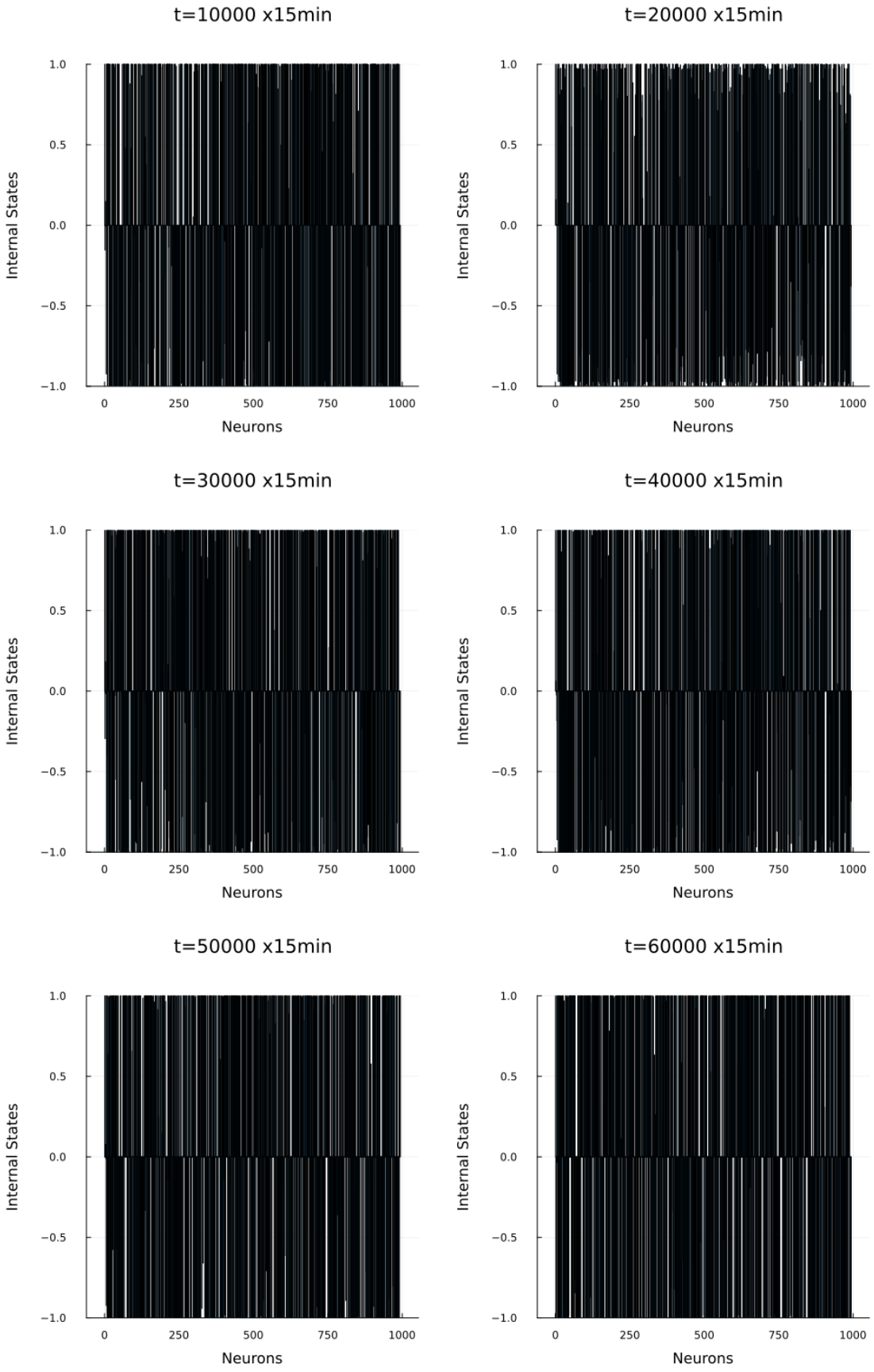


Figure A.41: Reservoir internal states at various moments during training for the PC time series for the first set of hyperparameters

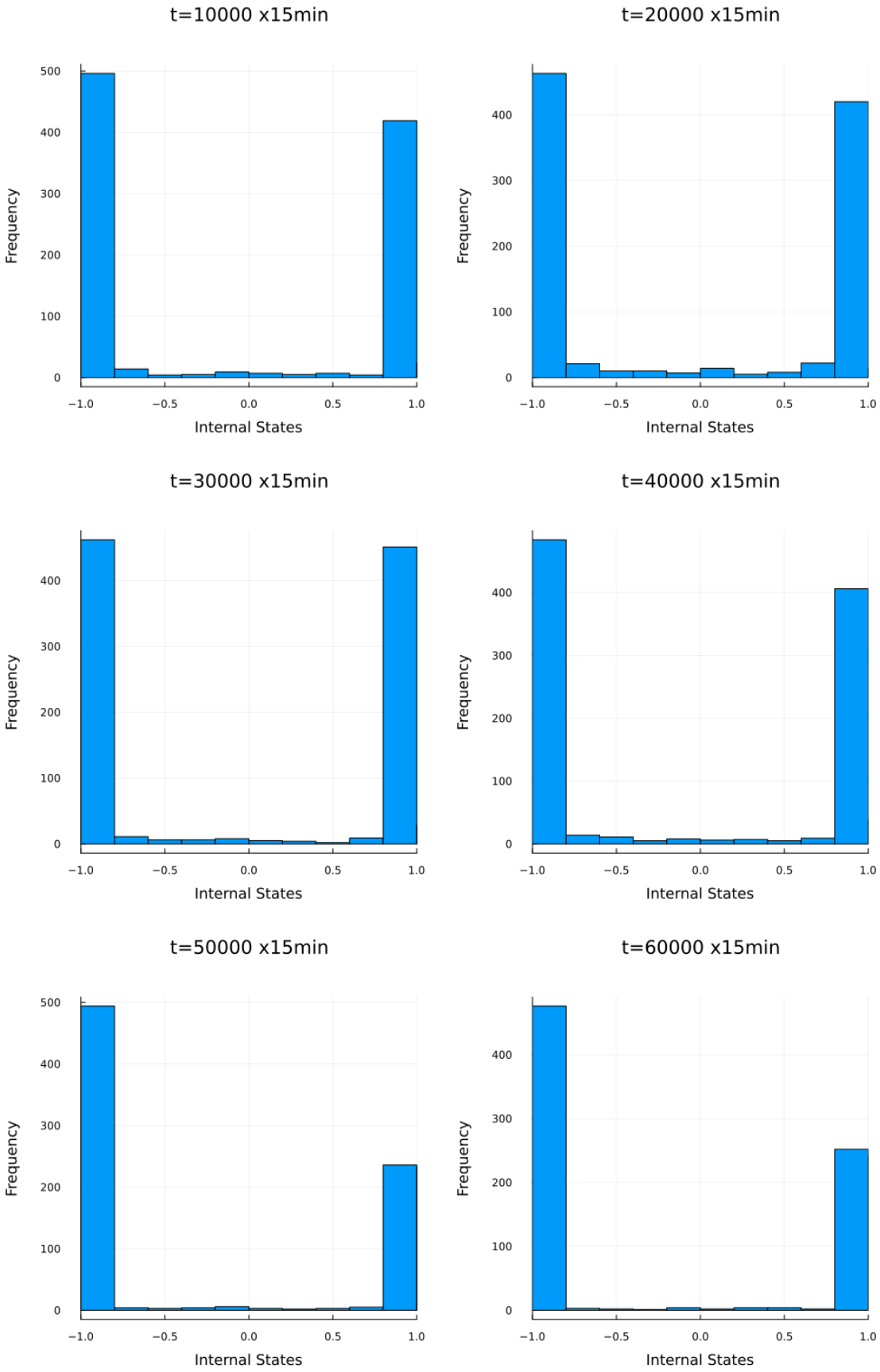


Figure A.42: Distribution of the reservoir internal states at various moments during training for the PC time series for the first set of hyperparameters

A.2.2 Second set of hyperparameters for the PC dataset

The second set of hyperparameters under study generates the second smallest reservoir size and the second highest nRMSE value, 0.50, keeping the trend on light reservoirs capable of generating the least error prediction. As mentioned before, certain characteristics of the reservoir configuration are common, for instance the connectivity is found to be around 20% of the size for all sets being high for an sparse reservoir. The short-term memory, determined by the leak rate in combination with the spectral radius scaling factor, should have similar characteristics to the first set. The generated dynamics in the reservoir is slow once more, preferred for inputs whose characteristics may take long to appear again.

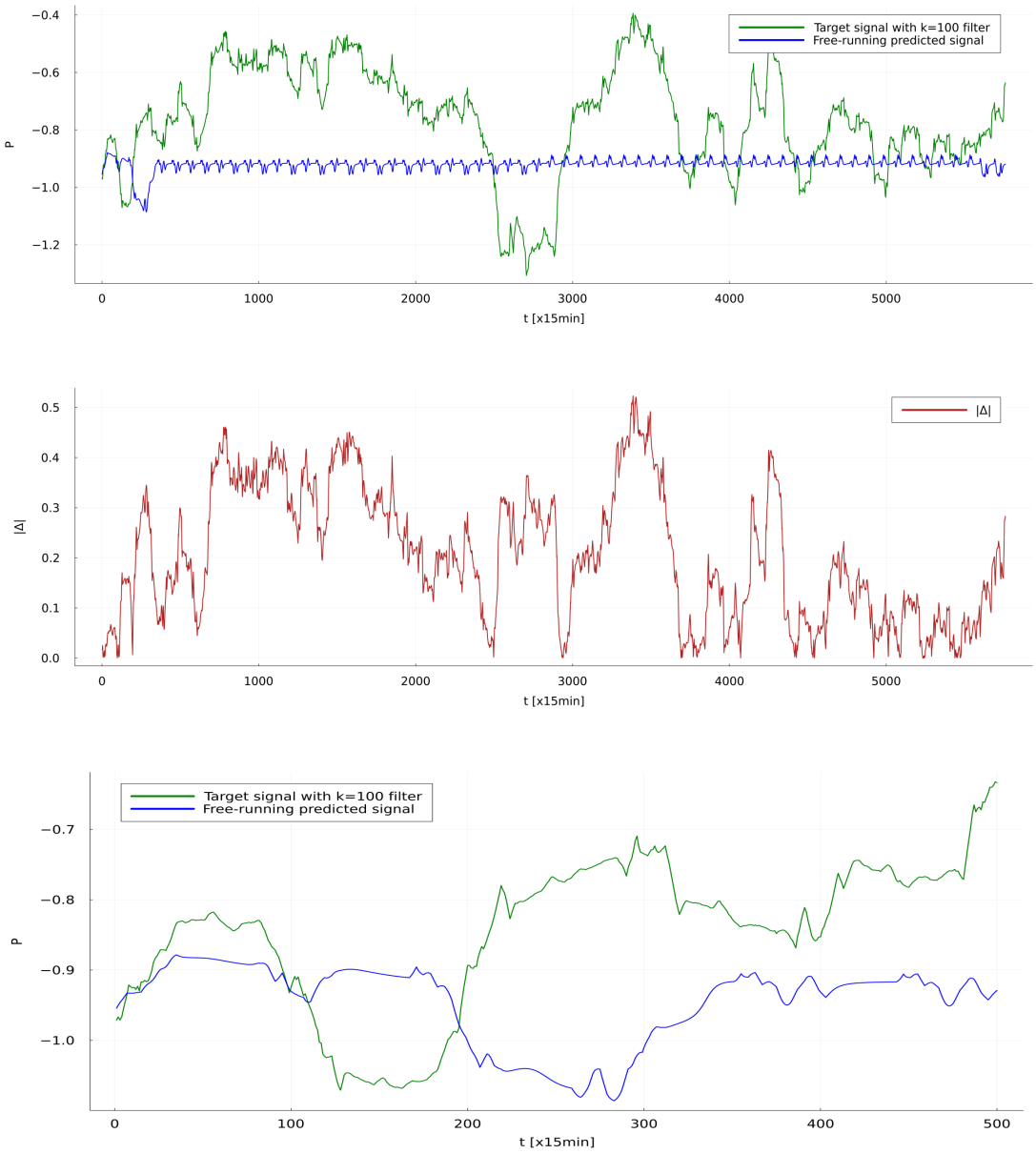


Figure A.43: PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the second set of hyperparameters

This specific reservoir presents an anomalous behaviour as it has the highest input scaling factors with the one with the higher weight the unitary input rather than the power or δ ones. Results will show that the prediction presents an abrupt change as in the first set, but the range at which it is

well behaved the results will not be as precise. Separability will be compromised in this reservoir as well, as internal states will present even more saturation due to the magnitude of the scaling factors.

Figure A.43 represents the prediction generated for the testing phase, with an abrupt change at around 500 time steps, driving the reservoir into generating a signal fluctuating around the median value of the target. The initial prediction under the 500 time steps mark has characteristics similar to the original, even if it does not even follow its trend from 100 time steps on. At the beggining the prediction mirrors the target, and then quickly looses its track and undershoots missing on the upward trend. As for the rest of the testing phase, the instability remains until the end. At 3000 time steps, as happened for the first set, the periodicity changes slightly but does not deviate from the median value.

The unfiltered prediction at Figure A.44 does not present any abrupt change as does the filtered, but it neither approaches the complexity of the original nor its trends. The nRMSE is 0.33, a substantial increase from the error on the filtered prediction. The only characteristic that may be interesting is the generated period matches at times the ones from the original, but its not usable for determining future power consumption.

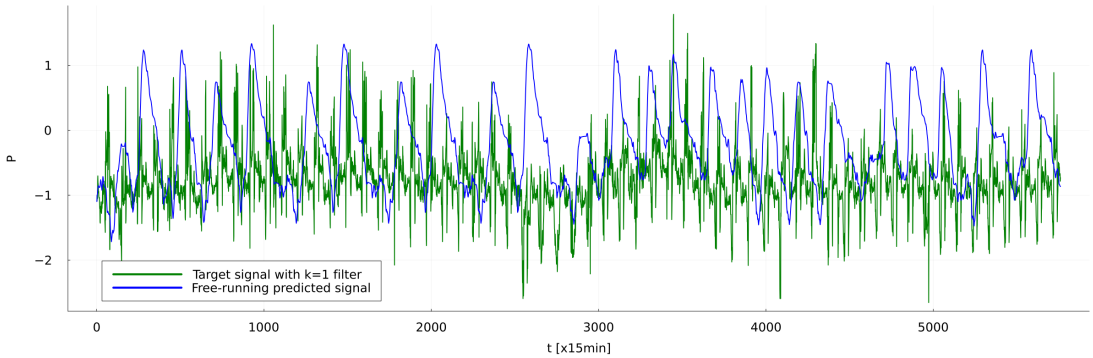


Figure A.44: PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the second set of hyperparameters

The δ auxiliary signals are predicted once more without much success, as can be observed in Figure A.39, and none of them can actually follow the target variations except for the underlying periodic behaviour as Figure A.40 shows. The RMSE results on Table 4.5 are promising, but take for instance the error on δ_1 (0.21). It would be indication that the prediction is close to perfect to the original, but falls short on actually capturing the maxima and minima variations. Once again, the target δ_{100} signal resembles the target power signal and both their predictions change abruptly at around the same time. This coincidence suggests that 500 time steps is the actual stability range for this dataset and from this point on the reservoir looses track of the original dynamics.

The reservoir internal states represented at Figure A.47 are heavily saturated as expected by the magnitude of the input scaling factors. The separability condition is virtually lost throughout training, as most inputs will soon overwhelm the neurons and generate fringe values. When looking at the histograms in Figure A.48 the situation is much more clear, as at the shown time steps over 75% of the neurons are at activation function fringe values with more than 50% at 1. There is no change as training advances, as the frequency for 1 is constantly over 400 across the six time steps. Having the unitary input with such an elevated factor is even worse for the reservoir than having the power input factor to be high, as in this case a reference value that only serves to give stability to the reservoir drowns any other signal. Just as data normalization helped avoid higher valued data to "squash" out the rest [29], having an input scaling factor of such magnitude at the unitary

input is preventing the Echo State Property from properly separating inputs within the reservoir.

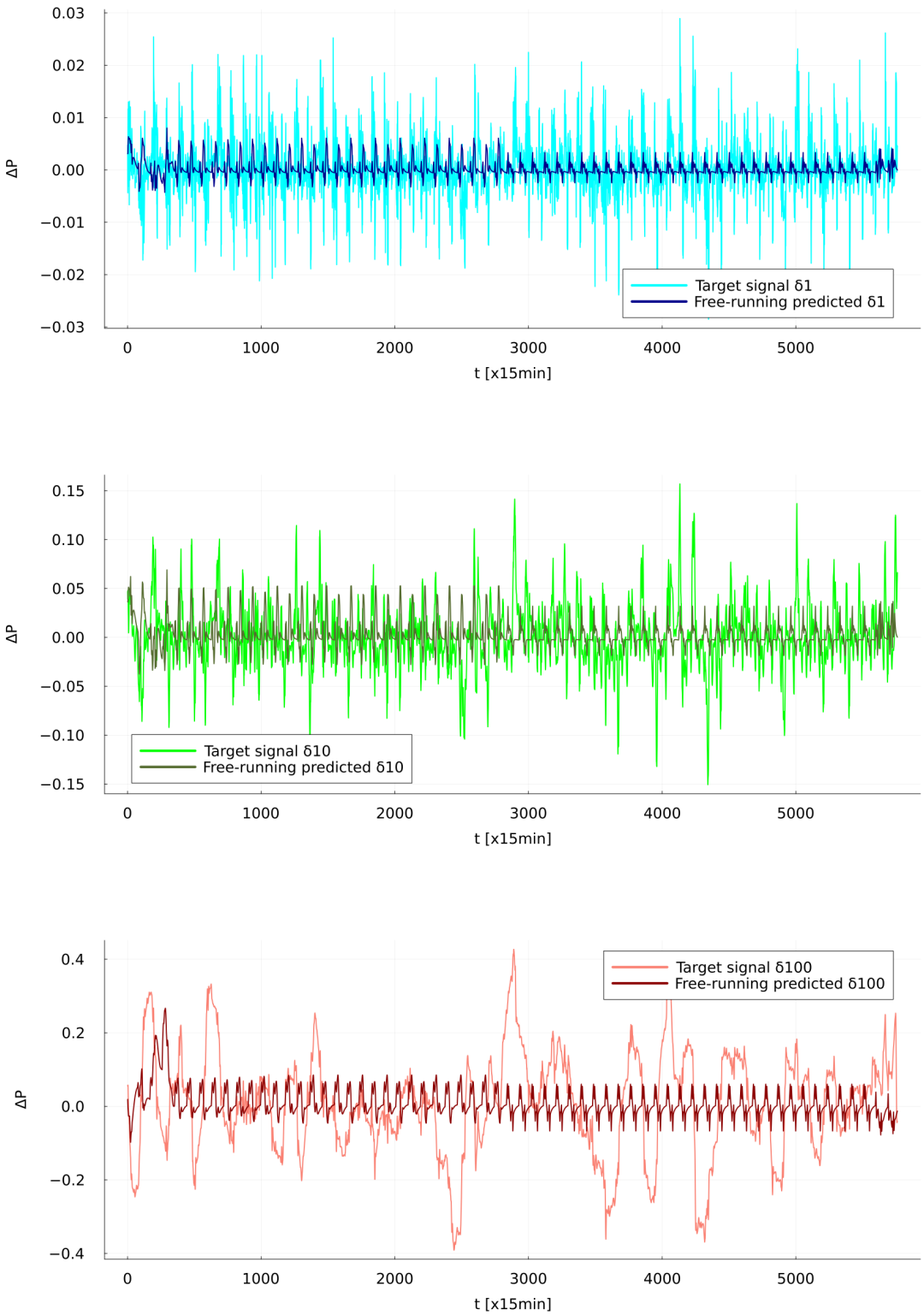


Figure A.45: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the second set of hyperparameters



Figure A.46: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the second set of hyperparameters

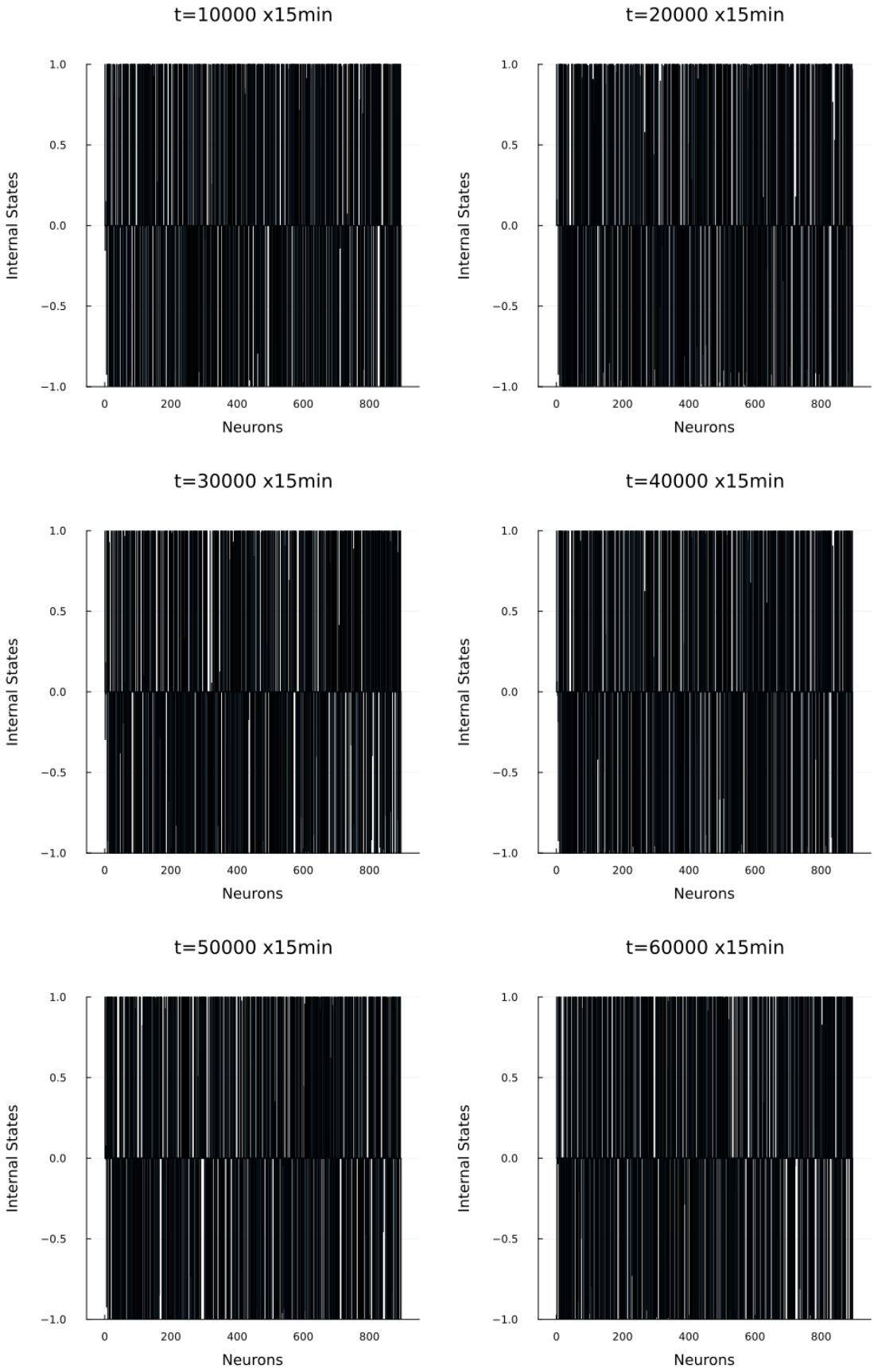


Figure A.47: Reservoir internal states at various moments during training for the PC time series for the second set of hyperparameters

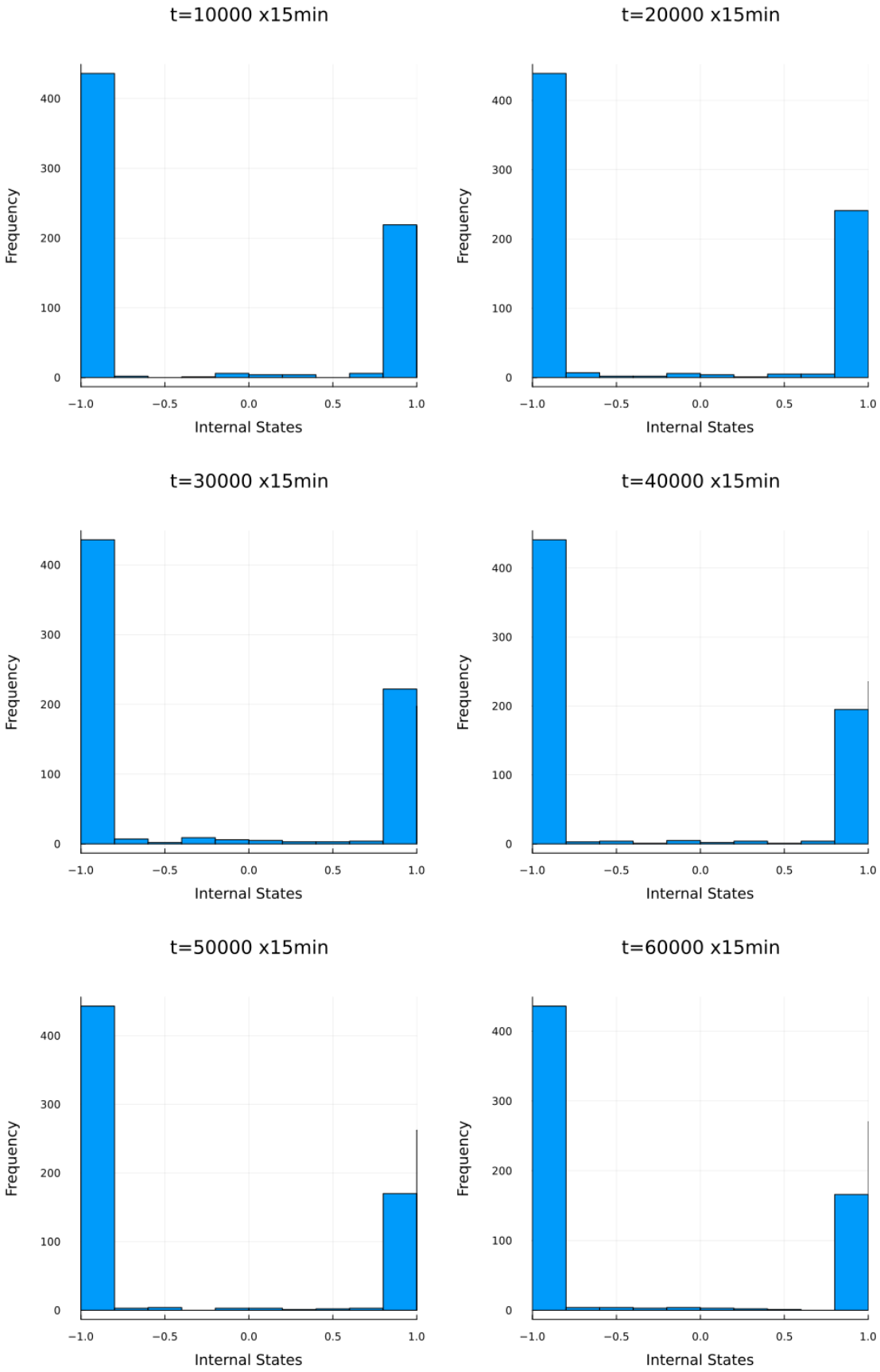


Figure A.48: Distribution of the reservoir internal states at various moments during training for the PC time series for the second set of hyperparameters

A.2.3 Fourth set of hyperparameters for the PC dataset

The last set of hyperparameters generates the biggest reservoir from the ones optimized for the PC dataset as well as the lowest RMSE at 0.45. With more than 3500 neurons it is almost an order of magnitude bigger than the smallest reservoir from the sets. The leak rate is the highest and along a remarkably low spectral radius scaling factor of around 0.4 leads to the short-term memory to be significantly smaller than the ones found in the other reservoirs. The internal states on this reservoir will have quick dynamics and not much knowledge on previous inputs.

The input scaling factors are similar to those found on the third set, this time lower than 10 but not a whole order of magnitude bigger than the fringe values. This time the unitary factor makes that input to be negligible before the power signal by halving its value, while the auxiliary input is given almost as much weight as the power. Saturation therefore should be avoided, except it will be present and will be seen once the internal states are studied.

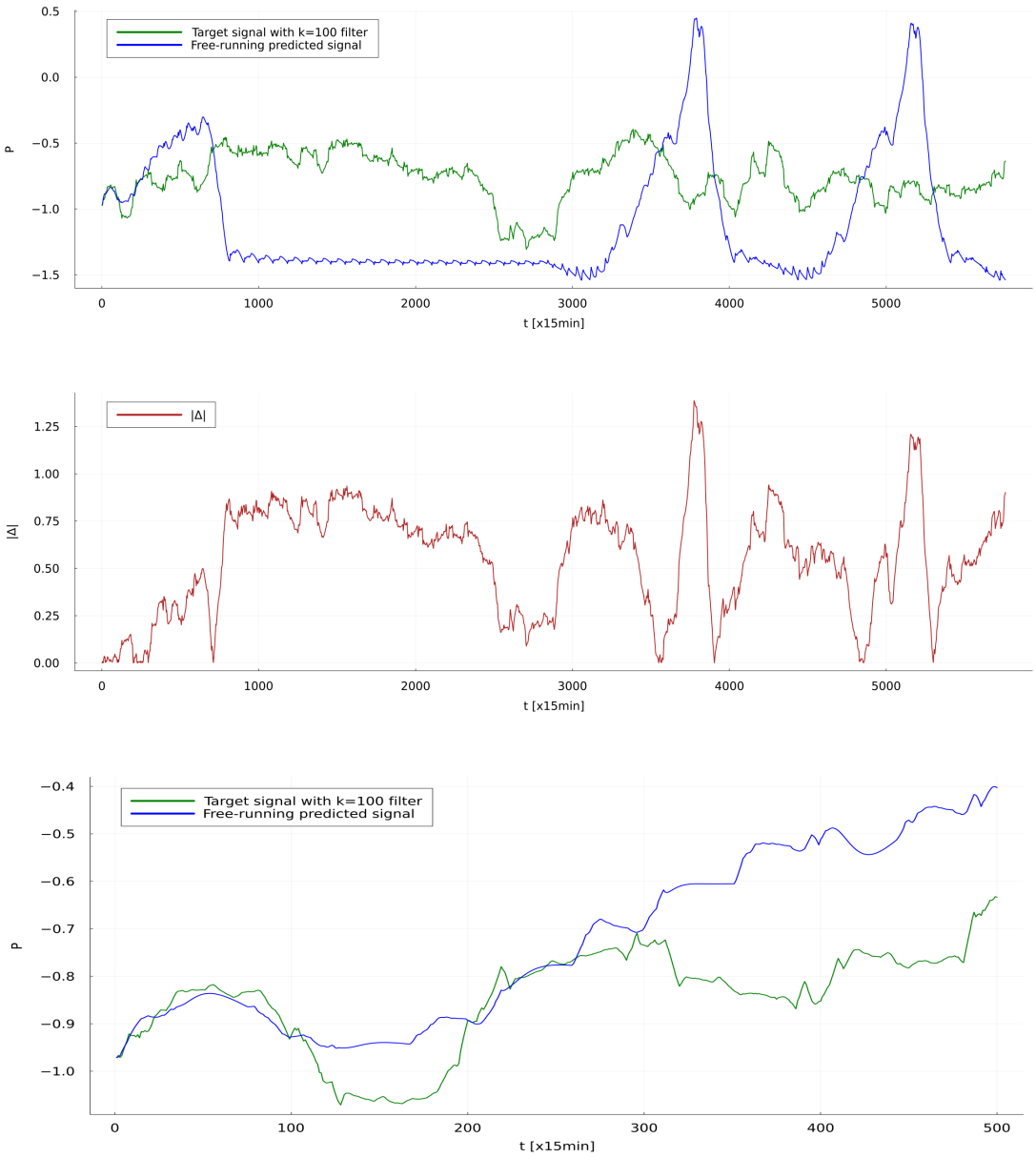


Figure A.49: PC time series with a CMA filter of $k = 100$: original (green) and predicted (blue) (above); the absolute difference variation in time (middle); and zoomed in on the first 500 time steps (below) for the reservoir with the fourth set of hyperparameters

Power consumption prediction during the testing phase is shown in Figure A.49 as well as the range at which the signal is considered stable, the first 500 time steps. As the rest of the sets, an abrupt change is present at this prediction even if it begins later on,at around 800 time steps. The prediction initially follows the rise in power consumption, mirroring the original behaviour and a slight divergence appears between a 100 and 200 time steps when the prediction does not follow the drop, later catching on and matching the target until the 300 time steps mark. From that point onwards, the predicted signal starts diverging by overshooting, following the upward trend until the instability begins. As the absolute difference shows, $|\Delta|$, the margin between both signals is negligible initially, and once the prediction becomes unstable it increases dramatically . At first the signal remains fluctuating periodically until the 3000 time steps once more, after which an anomalous behaviour pops up and two maxima appear, not matching any of the previously seen instabilities.

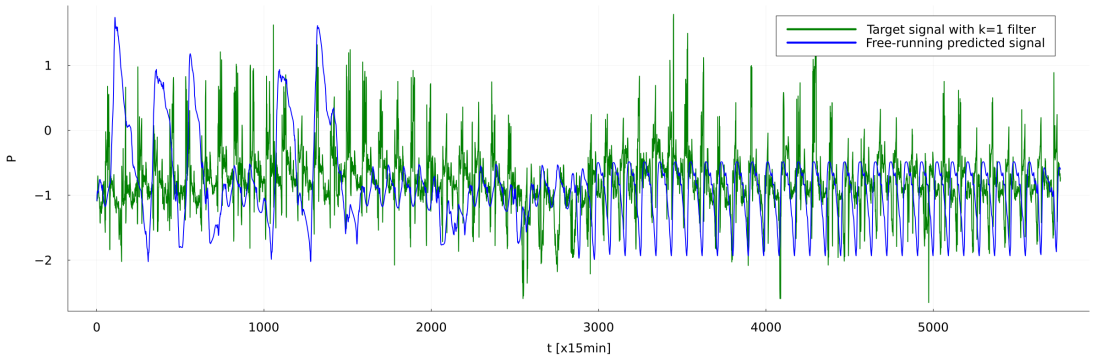


Figure A.50: PC time series with a CMA filter of $k = 1$: original(green) and predicted (blue) for the reservoir with the fourth set of hyperparameters

The unfiltered prediction is devoid of any instability present on the filtered one, once again not managing to represent neither the trends nor the amplitude fluctuations of the original. Right at the 3000 time steps mark the generated signal changes behaviour, matching the change in behaviour from the filtered signal.

The prediction on the δ_1, δ_{10} and δ_{100} auxiliary signals, depicted at Figure A.51, present the same disturbances as the power consumption signal, furthering the theory that it is due to a generalized instability in the internal states which cause this shift in behaviour. The predictions on the δ_1 and δ_{10} signals where closely matching the variations on the originals up to the disturbance at time step 800 when observing Figure A.52, driving the signals into fluctuating blinding round the original signal median. As for the δ_{100} the effect from the first disturbance was noticeable even if the prediction was missing the target, and was the culprit on the two following massive disturbances. The δ_{100} predicted signal is the first to go unstable, dropping severely twice and causing the two maxima seen in the power signal prediction.

The internal states in Figure A.53 are clearly saturated, reaching a point where most of the inputs are indistinguishable from each other. The histogram at Figure A.54 shows that this time half the neurons are at 1 and the other half at -1 during the entire training phase. Essentially there is no separability condition and training the output layer successfully becomes near impossible. Why is it then that even when the input scaling factors are low does the reservoir saturate? The answer lies in the size, it has been observed that saturation may also happen with a higher amount of neurons. Therefore the unsaturated reservoir may lie in a specific size range along with the appropriate input scaling factors. As for the PC dataset, saturation is avoided when N is between 350 and 2000 neurons. The second set of hyperparameters is on the verge of saturation, while this case is completely saturated.

The appearance of saturation on a bigger reservoir, with input scaling factors below 10, is due to the increased number of connections between neurons. When the reservoir grows so does the total number of connections, as for this reservoir the sparseness is determined by parameter c which is a fraction from the total size in which a neuron is connected to others. A single neuron receives inputs not only from the input layer itself, but from every other neuron connected to it through the random generation of the adjacency matrix. Therefore saturation occurs when each neuron is bombarded with scaled inputs that are not properly scaled to the volume of traffic the reservoir can handle. Hyperparameters do not act independently on the reservoir and juggling its values to assimilate stable dynamics is a non-trivial problem. When increasing the size of the reservoir it becomes necessary to either reduce the connectivity or the input scaling factors proportionately to avoid saturation.

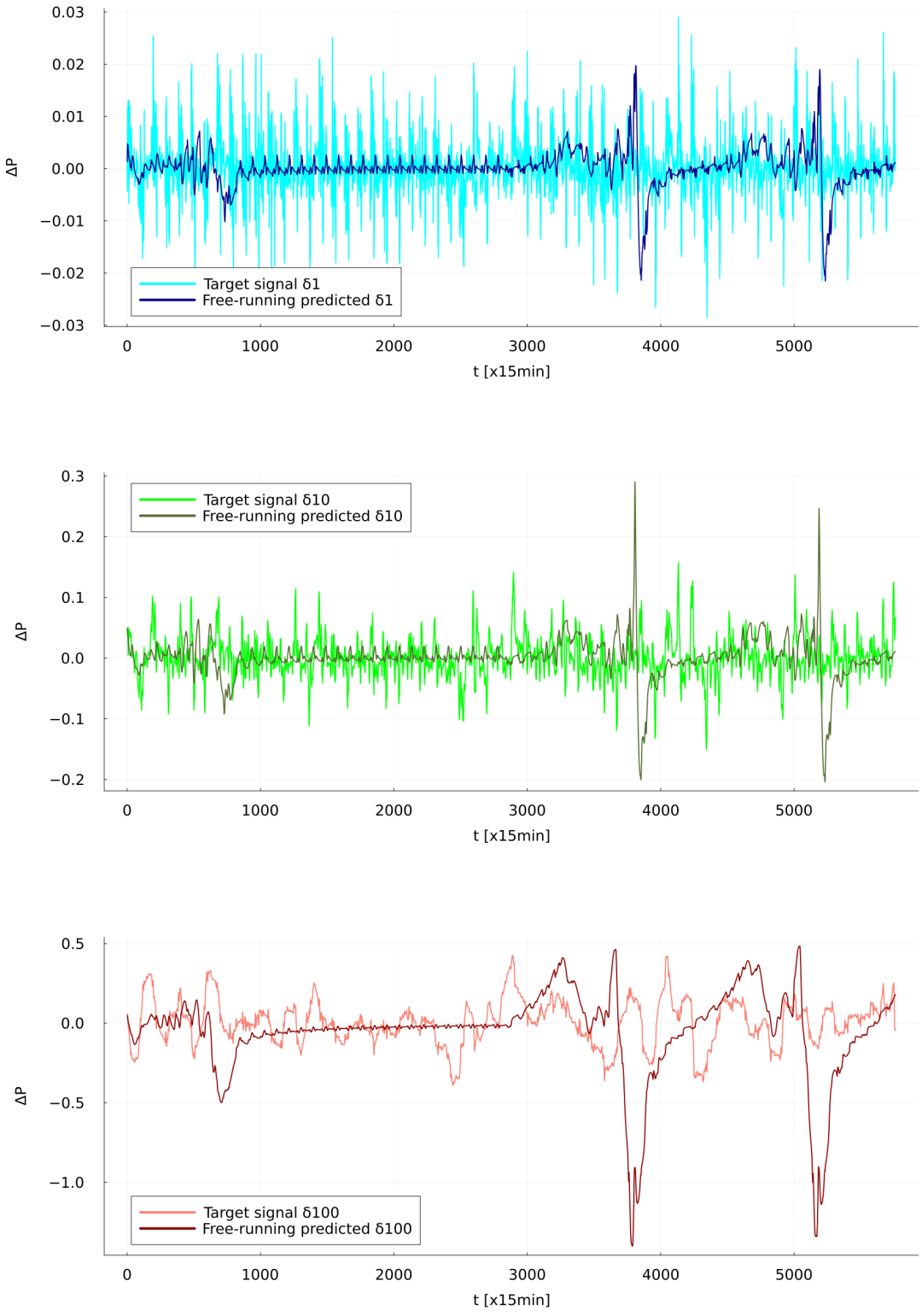


Figure A.51: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} prediction for the reservoir with the fourth set of hyperparameters



Figure A.52: PC time series with a CMA filter of $k = 100$: δ_1, δ_{10} and δ_{100} initial 500 time steps prediction for the reservoir with the fourth set of hyperparameters

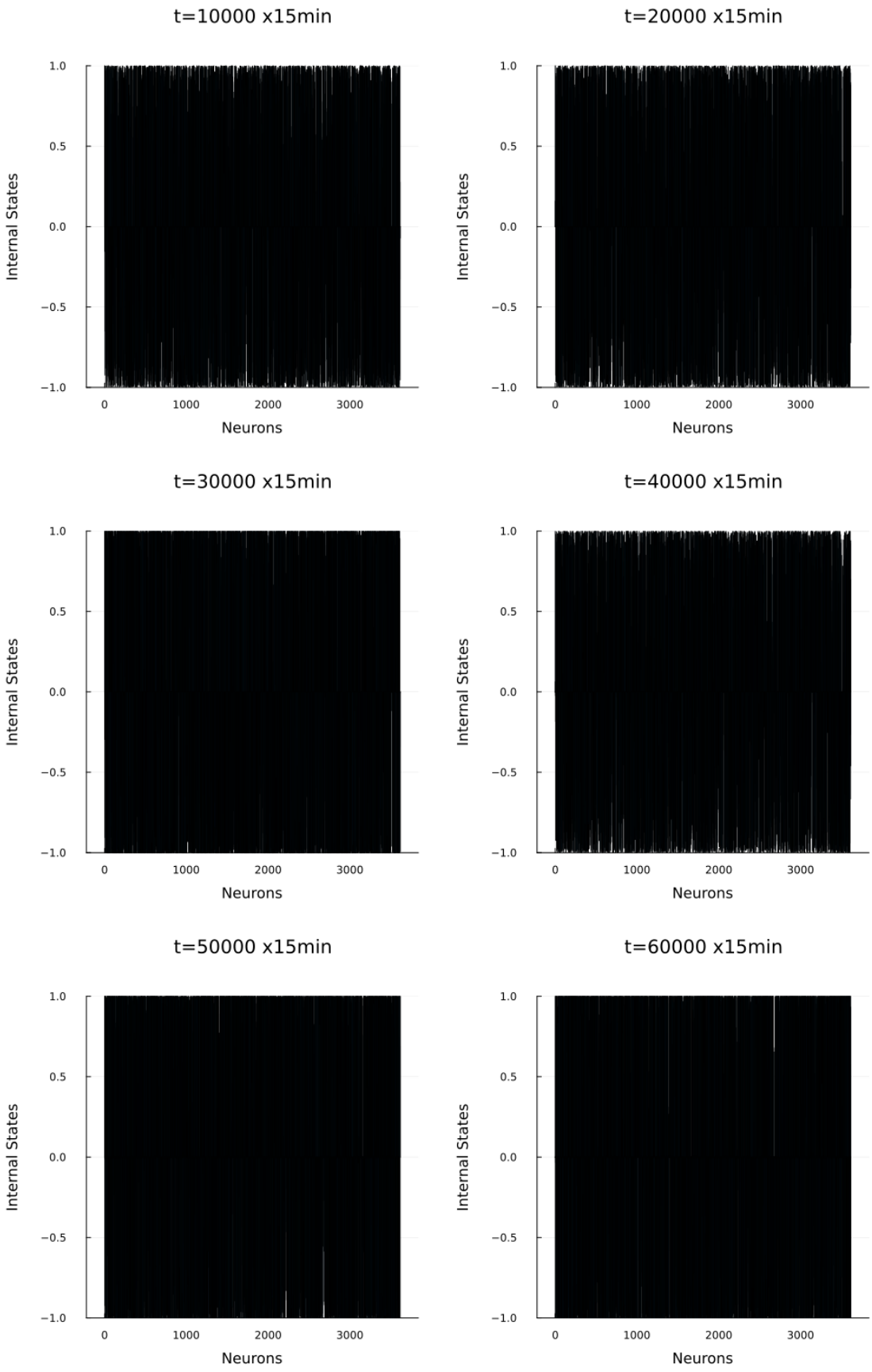


Figure A.53: Reservoir internal states at various moments during training for the PC time series for the fourth set of hyperparameters

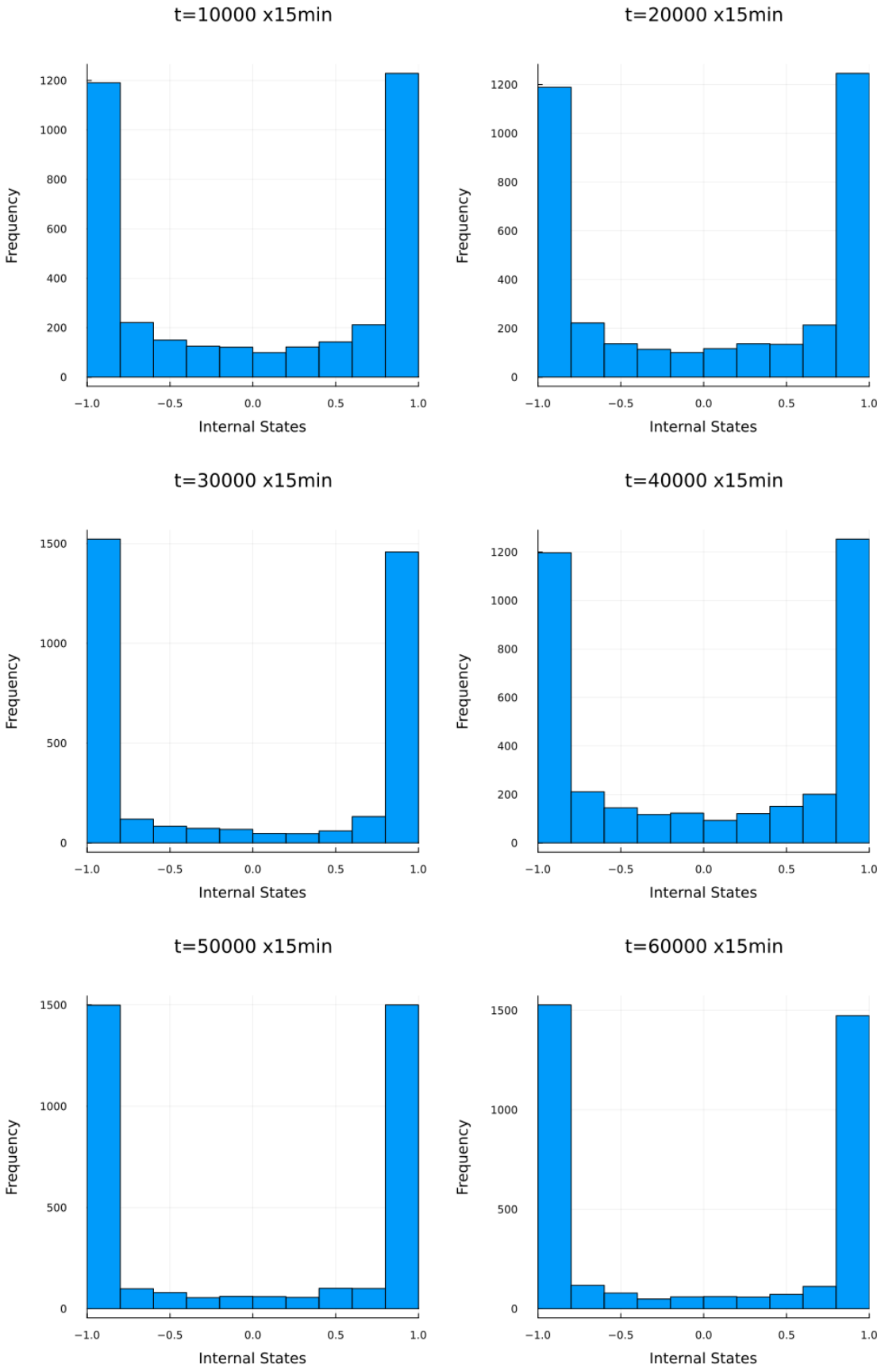


Figure A.54: Reservoir internal states histogram at various moments during training for the PC time series for the fourth set of hyperparameters

B | Published article

engineering proceedings

MDPI

Proceeding Paper

Reservoir Neural Network Computing for Time Series Forecasting in Aerospace: Potential Applications to Predictive Maintenance †

Juan Manuel Rodríguez Riesgo and Juan Luis Cabrera Fernández *

Departamento de Física Aplicada, ETSI Aeronáutica y del Espacio, Universidad Politécnica de Madrid, Pza. Cardenal Cisneros 3, 28040 Madrid, Spain; juan.rriesgo@alumnos.upm.es

* Correspondence: juanluis.cabrera@upm.es

† Presented at the 10th International Conference on Time Series and Forecasting, Gran Canaria, Spain, 15–17 July 2024.

Abstract: Coupling a reservoir neural network and a Grey Wolf optimization algorithm the system hyperparameters space is explored to find the configuration best suited to forecast the input sensor from the NASA CMAPSS dataset. In such a framework, the application to the problem of predictive maintenance is considered. The necessary requirements for the system to generate satisfactory predictions are established, with specific suggestions as to how a forecast can be improved through reservoir computing. The obtained results are used to determine certain common rules that improve the quality of the predictions and focus the optimization towards hyperparameter solutions that may allow for a faster approach to predictive maintenance. This research is a starting point to develop methods that could inform accurately on the remaining useful life of a component in aerospace systems.

Keywords: reservoir computing; grey wolf optimizer; aerospace; time series forecasting

check for updates

Citation: Riesgo, J.M.R.; Cabrera Fernández, J.L. Reservoir Neural Network Computing for Time Series Forecasting in Aerospace: Potential Applications to Predictive Maintenance. *Eng. Proc.* **2024**, *68*, 17. https://doi.org/10.3390/engproc2024068017

Academic Editors: Olga Valenzuela, Fernando Rojas, Luis Javier Herrera, Hector Pomares and Ignacio Rojas

Published: 4 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Experience is the foundation for any modern maintenance process. Thus all prognostics on complicated systems are either model-based or data-driven. The former depends on the use of physics-based models of the degradation processes and are typically applied to safety-critical and slow-degrading equipment whose degradation mechanisms have been extensively studied. The latter, only used when no accurate physics model is available, requires enormous amounts of data for training, only being applied to non-safety critical systems with short mean time to failure [1].

Model-based prognostics have been the preferred choice during the last century, resting the required actions to be taken as much from the physical knowledge behind the model as from the experience of the engineers tasked to maintain it. At the dawn of this millennia though, new technologies originated providing a far more powerful process to predict failure of a mechanical system given enough data on it. The advent of Machine Learning brought a whole new set of tools to predict, simulate and diagnose events that would be catastrophic if occurring on a real-life scenario, allowing an increase in reliability across all modern industrial fields.

Data-driven models are therefore gaining significant traction in the engineering landscape, specially in Aerospace, as even reduced improvements through computing-intensive simulations can yield millions through efficiency savings on the long run. Not only Machine Learning is being used extensively, but digital models are being widely employed so as to combine both technologies and speed up greatly project development. Industrial applicability of data-driven frameworks [2] have been prominently favored by causes as: the increase in data availability; high performance computing; improved data storage and

sensor technologies and a greater academic focus on algorithms and applied mathematics; not to mention a flood of investment.

Predictive maintenance is also gaining traction through the combination of increased capabilities in sensor networks as well as computational power. The ability to pour through high dimensional and multivariate data and classify its internal relationships is key to determine the potential failures bound to happen [3].

The advantages of using Reservoir Neural Networks Computing (RC from now on) for predictive maintenance in aerospace have been explored before, especially on components with easy to acquire data, e.g.,: to predict medium and long-term fluctuations on avionic modules with an Echo State Network (ESN) [4], allowing for preventive maintenance in case it strides off from its nominal value; or using an ESN to predict the Remaining Useful Life from the operational data of several turbofan engines, analysing operational settings and sensor information to determine how many cycles are left before catastrophic failure [5].

Accurate signal prediction opens up the door for efficient and safe control, automatically adjusting and correcting any kind of output sensor whether the operational settings (OP) of an engine, the flight-path of a drone, the coordination of a bipedal robot or human controlled situations [6]. If fully integrated into a system, it will prevent from failures caused by sluggish control algorithms that simply react once the action has happened, rather than adjusting the system to prevent it.

2. Materials and Methods

2.1. RC-GWO System

In this work we report the construction of an hybrid system involving the coupling of a reservoir neural network and a Grey Wolf optimization algorithm which we will call the Reservoir Computing-Grey Wolf Optimizer system (RC-GWO from now on). Such a system receives an input composed by sensor signals and explores the hyperparameter space to find the configuration best suited for a specific prediction. Once the system finds the optimal settings for a reservoir it is not necessary to run it again for generating new predictions of an updated dataset. For instance, let us consider the example of the power consumption of a device during three years, the training data could be selected to be all available data apart form the last two months and the validation data those last two months. When the appropriate reservoir hyperparameters configuration has been found, new results could be quickly generated bimonthly without having to optimize the neural network. This can be repeated until the normalizedmean square error of the predictions differs from a given tolerance and the process begins anew. To optimize the reservoir certain hyperparameters must be defined. Ref. [7] serves as a foundation for defining the optimal configuration vector which the wolves will try to hunt down.

2.1.1. Reservoir Computing

RC is a specific type of Recurrent Neural Network (RNN) consisting of two components: a nonlinear dynamical system that accepts an input signal (the reservoir) and one trainable layer (called the readout layer) producing the output. If the reservoir is complex enough, in principle, any computation can be achieved by just training the readout layer to achieve a desired functionality [8]. This approach was first introduced in [9,10], using the terms Echo State Networks (ESN) and Liquid State Machines (LSM) but nowadays it is better known by the general term reservoir computing (RC), to stress the role of the reservoir but also to make a difference with Liquid Neural Nets (LNNs) [11]. RC has been shown to be particularly useful to handle chaotic time-series prediction and other complex tasks effectively [12].

A reservoir can be composed by a random combination of activation functions that is capable of encoding the original dynamics into higher dimensions. Any system can be considered a reservoir given an input, an output and an interpretation layer transforming the output into readable information. Take for instance the use of Brain Organoids—brain-like 3D in vitro cultures—as a reservoir due to its capacity for nonlinear dynamics and

fading memory properties [13]. In the case of RC the relationships between the elements of the reservoir (neurons characterized by an activation function) define a graph given by an adjacency matrix. Given an input the resulting state of an activation function promote specific paths for the propagation of the stimulus. Training a reservoir is nothing but observing during enough time how it reacts to a given set of information and making a linear correlation with a desired output, not delving in the complex dynamics that develops inside and only skimming the surface reaction for answers.

The reservoir employed on the RC-GWO system is based on the ESN, which establishes the required mathematical properties of a valid reservoir through the Echo State Property (ESP). A good reservoir is one that is able to separate inputs; different external inputs should drive the system to different regions of the configuration space. This is called the separability condition [14], which relates asymptotic properties of the excited reservoir dynamics to the driving signal. Intuitively, the ESP states that the reservoir will asymptotically wash out any information from initial conditions. The ESP is guaranteed for additive-sigmoid neuron reservoirs, if the reservoir weight matrix (and the leaking rates) satisfy certain algebraic conditions in terms of singular values. For such reservoirs with a tanh sigmoid, the ESP is violated for zero input if the spectral radius of the reservoir weight matrix is larger than unity. Conversely, it is empirically observed that the ESP is granted for any input if this spectral radius is smaller than unity. This has led to a far-spread but erroneous identification of the ESP with a spectral radius below 1. Specifically, the larger the input amplitude, the further above unity the spectral radius may be while still obtaining the ESP [15]. For a reservoir to hold the ESP it must resonate information of all the inputs through the consequent internal states.

Given an N neurons reservoir, their links strength (weights) are represented by an $N \times N$ matrix, $\mathbf{W}$, with values selected from an uniform random distribution in the interval $[-1, 1]$. Therefore the reservoir is described by $\mathbf{W} \in \Re^{N \times N}$. Let's consider an input vector containing the sensor time series, $\mathbf{u}(t) \in \Re^{M}$ with $t = 1, \ldots, T$. This input sensor data is introduced into the reservoir by means of an input-reservoir coupler, i.e., a randomly generated matrix, $\mathbf{W}_{in} \in \Re^{N \times (1+M)}$, whose elements are drawn from an uniform random distribution in the interval $[-1, 1] - \{0\}$, i.e., it is generated as in the case for $\mathbf{W}$, but all elements of $\mathbf{W}_{in}$ should be non-zero, additionally, $\mathbf{W}_{in}$ does not require normalization.

Once $\mathbf{W}_{in}$ and $\mathbf{W}$ have been set up, the internal states for each of the inputs are harvested. Typically the neurons used for this reservoir are discrete-time leaky-integrate units, ruled by the update equations:

$$\hat{\mathbf{x}}(t) = \tanh(\mathbf{W}_{in} \cdot [1; \mathbf{u}(t)] + \mathbf{W} \cdot \mathbf{x}(t-1)), \tag{1}$$

$$\mathbf{x}(t) = (1 - \alpha)\mathbf{x}(t-1) + \alpha \hat{\mathbf{x}}(t). \tag{2}$$

where, $\mathbf{x}(t) \in \Re^{N}$ is the vector with the internal states of neurons in the reservoir, $\hat{\mathbf{x}}(t) \in \Re^{N}$ is its update vector for new inputs, all at time step t, $\alpha \in (0, 1]$ is the leaking rate and $[1; \mathbf{u}(t)]$ is the vertically concatenated unitary value with the input at time step t. Then the network output, $y(t)$, is given by:

$$\mathbf{y}(t) = \mathbf{W}_{out} \cdot \mathbf{x}(t). \tag{3}$$

The computation of $\mathbf{W}_{out}$ is a linear equation system. It is fixed by means of a ridge-regression, also known as Tikhonov regularization:

$$\mathbf{Y}^{target} = \mathbf{W}_{out} \cdot \mathbf{X}, \tag{4}$$

$$\mathbf{W}_{out} = \mathbf{Y}^{target} \cdot \mathbf{X}^{T} \left(\mathbf{X} \cdot \mathbf{X}^{T} + \beta \mathbf{I} \right)^{-1}. \tag{5}$$

where, $\mathbf{X} \in \Re^{N \times T}$, is the internal states matrix in which the columns are state vectors $\mathbf{x}(t)$, for $t = 1, \ldots, T$; $\mathbf{Y}^{target} \in \Re^{M \times T}$ is the output objective used for training, $\mathbf{W}_{out} \in \Re^{M \times N}$ is the output weight matrix needed to interpret the internal states back to a usable output, $\mathbf{I}$ is

the identity matrix and β is the regularization coefficient. Once the training is finished, the internal states have been harvested and the ridge-regression applied, the prediction phase runs on Equations (1)–(3), where the input $\mathbf{u}(t)$ is the predicted output of the preceding step $\mathbf{y}(t - \Delta t)$.

2.1.2. Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) is a meta-heuristic method proposed by Mirjalili et al. [16], based on the hunting patterns of grey wolves, mimicking their leadership hierarchy and hunting procedures. It is based on the fact that wolves have a rigid social structure, with a lead individual, known as alpha, making the decisions. It comes followed by the beta wolf, which enforces the alphas' decisions and gives feedback. The omega wolf is the lowliest of all, being the last to eat and basically acting as an scapegoat to prevent infighting.

The main phases of grey wolf hunting are: tracking, chasing and approaching prey; pursuing, encircling and harassing the prey until it stops moving; and attacking towards the prey. A mathematical model of this collective behaviour is what drives the optimization algorithm, candidates sharing information about the drive space, assisting in avoiding locally optimal solutions and exploring the objective space.

The first step is to establish that the fittest solution is the α wolf, followed respectively by the β and δ. Whatever remains of the population at that point are considered ω wolves, and follow the three lead wolves. The motion of wolves is described as:

$$\mathbf{D} = |\mathbf{C} \cdot \mathbf{X}_p(t) - \mathbf{X}(t)|, \tag{6}$$

$$\mathbf{X}(t+1) = \mathbf{X}_p(t) - \mathbf{A} \cdot \mathbf{D}. \tag{7}$$

Here, t, is the discrete iteration, $\mathbf{A}$ and $\mathbf{C}$ random vectors, $\mathbf{X}_p$ the position of the prey and $\mathbf{X}$ the position of the wolves, and:

$$\mathbf{A} = 2\mathbf{a}.\mathbf{r}_1 - \mathbf{a}, \tag{8}$$

$$\mathbf{C} = 2\mathbf{r}_2, \tag{9}$$

$$\mathbf{a} = \mathbf{a}_0(1 - t/T). \tag{10}$$

with, $\mathbf{a}$ decreasing linearly from $\mathbf{a}_0$ to $\mathbf{0}$ over the course of the iterations until the maximum number of them, T, is reached. As in the original algorithm only its initial and final value are specified, for the sake of simplicity it is usual to set $\mathbf{a}_0$ to a scalar, e.g., $a_0 = 2$ so that a decrease linearly from 2 to 0. $\mathbf{r}_1$, and $\mathbf{r}_2$ are random vectors uniformly distributed in $[\mathbf{0}, \mathbf{1}]$, and are of a dimension matching that of the wolves position vectors.

To hunt for the prey we need the location of the optimum, unfortunately this is an unknown. To simulate the hunting behaviour [16], the α, β and δ wolves are believed to have the most information on where that optimum is. Whenever a new population needs to be spawned, the three lead wolves are taken as a reference and the rest are forced to update their positions accordingly. In this process the distance between the α, β and δ wolves and the prey is modeled as:

$$\mathbf{D}_\alpha = |\mathbf{C}_1 \cdot \mathbf{X}_\alpha - \mathbf{X}|, \tag{11}$$

$$\mathbf{D}_\beta = |\mathbf{C}_2 \cdot \mathbf{X}_\beta - \mathbf{X}|, \tag{12}$$

$$\mathbf{D}_\delta = |\mathbf{C}_3 \cdot \mathbf{X}_\delta - \mathbf{X}|, \tag{13}$$

where $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$ and $\mathbf{X}_\delta$ are the locations of leading wolves; $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$ represent random vectors, and $\mathbf{X}$ specifies the location of the present solution. The estimation of the distances allows for the update rules:

$$\mathbf{X}_1 = \mathbf{X}_\alpha - \mathbf{A}_1 \cdot \mathbf{D}_\alpha, \tag{14}$$

$$\mathbf{X}_2 = \mathbf{X}_\beta - \mathbf{A}_2 \cdot \mathbf{D}_\beta, \tag{15}$$

$$\mathbf{X}_3 = \mathbf{X}_\delta - \mathbf{A}_3 \cdot \mathbf{D}_\delta, \tag{16}$$

and,

$$\mathbf{X}(t+1) = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3}, \tag{17}$$

where $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ are random vectors and t is the recent iteration. The parameter a as given by (10) is modelled to signify the end of the hunt once the prey stops moving. The norm of $\mathbf{A}$ fluctuates in the interval $[-2a, 2a]$, where the elements of a are linearly decreased from 2 to 0. To promote exploration $\mathbf{A}$ may take different values; $|A| < 1$—as just mentioned—focuses the wolves on the hunt, $|A| > 1$, drives them to diverge from the prey and to find a better solution. The random vector $\mathbf{C}$ also plays part in the optimization exploration phase [16]: when $|C| > 1$, it encourages exploration while for $|C| < 1$, de-emphasizes it. In particular, the exploitation trend improves for $|A| < 1$ and $|C| < 1$. Thus, $\mathbf{C}$ provides random weights for the prey term in order to stochastically promote ($C > 1$) or impede ($C < 1$) the effect of prey in defining the distance in (6); opposite to $\mathbf{A}$, $\mathbf{C}$ does not linearly decrease with a in order to always enforce exploration.

The algorithm has been altered with respect to [16], as in our case some parameter bounds need to be established during the exploration phase. When a new population is generated their location are required to be within bounds. If wolves are not within bounds, the generation is repeated again and again until they are. Testing has proven that in some edge situations, a new population is always out of bounds. When that happens a countdown is activated, and once a wolf has been repopulated a certain amount of times the out of bounds coordinates in the exploration space are changed by a random value within the space defined by the three lead wolves. Such a procedure allows for exploration while guaranteeing wolves will not remain stuck in an out of bounds position.

2.1.3. System Architecture

The RC-GWO system, receives an input signal and runs through the hyperparameter space to find the best suited configuration for an specific prediction. Once the system finds optimal settings for a reservoir it is not needed to run it again to generate new predictions for an updated dataset. To optimize the configuration of a reservoir a set of hyperparameters must be defined. Ref. [7] serves as a foundation for defining the optimal configuration. Once $\mathbf{W}$, and $\mathbf{W}_{in}$, are generated—as explained above—the following reservoir parameters are needed: (1) The size of the reservoir, N. It is fundamental to appropriately capture the inherent complexity of the input sensor time series. As a rule of thumb, the more challenging the task the bigger the reservoir should be. Some limitations should be established to avoid overfitting and to avoid unnecessary computations. (2) The leak rate, α. It represents the speed of the reservoir update dynamics discretized in time. The closer to 1 the more it will forget about the previous internal states the reservoir went through. (3) The input matrix column scaling factor, a_{Win}. It needs to be as many as different inputs are ingested in the reservoir, always being at least two for the present ensemble. Each of the scaling factors apply to a specific column of the input weight matrix, with the first always being the one for the unitary value that accompanies the input and from the second onward for the input itself. In case the input sensor data is more than a single time series it is recommended to group them and apply the same factor for those that contribute differently to the task. (4) The sparsity of the reservoir, c. It represents the connectivity among the neurons in the reservoir. In [7] it is stated that when connecting each reservoir node to a small number of other nodes (e.g., 10) on average, irrespective of the reservoir size, computation speeds up. Sparse matrix representations are recommended as they give slightly better performance, but in general this hyperparameter does not affect

much the quality of the prediction. As indicated in [15], some authors have developed fully connected reservoirs which work as well as sparsely connected ones. (5) The spectral radius scaling factor, K_ρ. It is used to enforce the echo state property through establishing the maximal absolute eigenvalue of the internal weight matrix, $\mathbf{W}$. Once a random sparse matrix is generated, its spectral radius, $\rho(\mathbf{W})$, is computed. Then $\mathbf{W}$ is divided by the spectral radius to yield a matrix with unitary spectral radius and scaled with $\mathbf{K}_\rho$ to adapt to the specific input ingested in the reservoir, i.e.,

$$\mathbf{W}_{scaled} = \mathbf{W}_{initial} \frac{\mathbf{K}_\rho}{\rho(\mathbf{W}_{initial})}. \tag{18}$$

(6) The last hyperparameter to be tuned is the Tikhonov regularization coefficient, β, which serves to obtain the output layer through ridge-regression. A $\mathbf{W}_{out}$ matrix with high values is an indication of a very sensitive and unstable solution.

Once the ensemble is initialized it will determine autonomously the best suited set of hyperparameters for a given input sensor data with minor tweaking necessary. Figure 1 shows the system workflow. The GWO algorithm is called and then the reservoir runs for evaluating the wolves. Once the data is ingested the normalized sensor data is calculated,

$$x_{norm}(t) = \frac{x(t) - \mu}{\sigma}, \tag{19}$$

where $x(t)$ is the sensor data at t, and μ and σ are the mean and standard deviation. The data needs is normalized, to guarantee the outliers do not "squash out" the rest of the data when applied to the activation function.

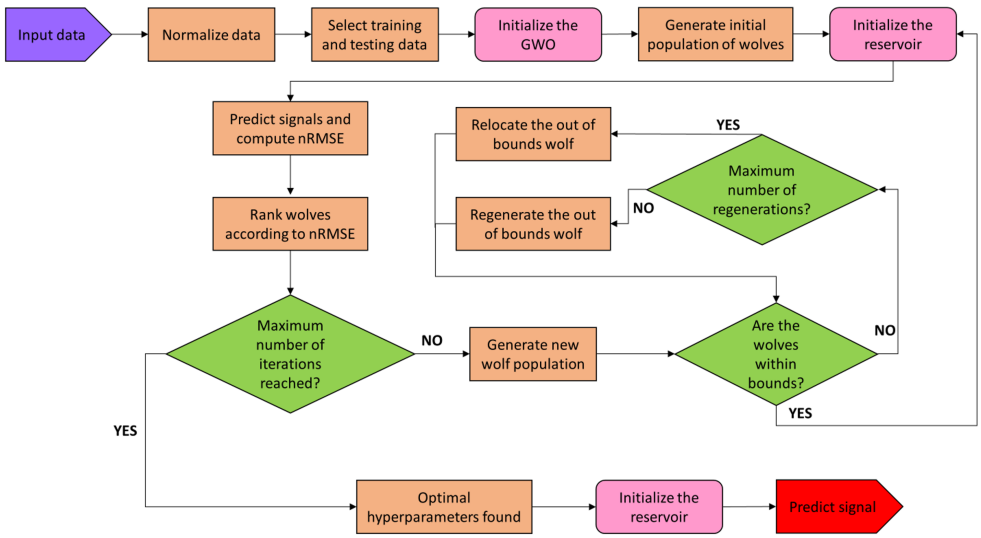


Figure 1. RC-GWO system flowchart. The sensor data enters as an input for pre-processing and hyperparameters optimization to find a compatible reservoir to generate the prediction upon.

Training and testing data selection follows the traditional optimal relation: 70% is devoted to training and 30% to testing. Another important aspect is establishing the initial transient to be discarded during training, as stated in [7] the initial internal states are affected by the fact that the first internal state is null. How much of those initial internal states is discarded falls under the "find out" doctrine, as sometimes with short sequences not much can be discarded without nothing being left. In our case the initial transient for the training to ignore has been fixed at 10% of the total input data, so that the reservoir may forget the unnatural states that are not often activated, while keeping the "warmed-up" states that hold better information on the dynamics of the system.

Once the GWO has been initialized, a starting population of wolves is created by randomly assigning hyperparameters values in the allowed bounded set. This favors an increased coverage of the configuration space. Now the process of optimization begins, wolves are run through the hyperparameter space and ranked according to the normalized Root Mean Square Error (nRMSE) calculated along the length of the testing phase, n_{test},

$$\delta_i \equiv y_i - Y_i^{target} \tag{20}$$

$$\Delta_i \equiv \frac{\delta_i}{\max|\delta_i|} \tag{21}$$

$$nRMSE = \sqrt{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \Delta_i^2}. \tag{22}$$

The process continues until reaching T. The production of a new wolf population after discarding those out of bounds occurs mainly at the latest stages of the optimization, where most out of bounds wolves are generated, as those that try to explore the configuration space further from the prey begin probing the limits set to the configuration. Once T has been reached, the GWO provides the hyperparameter configuration best suited to the input sensor data and it is run once more. When more forecasts on updated data are required, the GWO is not needed anymore and the reservoir is able to generate new predictions based on the obtained reservoir with the optimal hyperparameters configuration.

2.2. Experimental Data

The *Commercial Modular Aeropropulsion System Simulation* (CMAPSS) dataset, was developed by Saxena et al. [17] as part of the NASA's Prognostic Center of Excellence Data Repository [18]. This dataset contains sensor information from a turbofan engine obtained from a thermodynamical simulation model. Several runs are contained in this dataset, each considered to be different turbofan engines of the same make and model with different initial degradation levels in their components. This degradation will affect how the engines behave over time, as damage and backlash can severely impact performance. These time series are measured in cycles, C, an unspecified unit of time that does not defeat the purpose of the investigation as the state on the previous time-step affects the next.

For the simulations three operational settings determine the turbofan behaviour: altitude, Mach and the Throttle Resolver Angle (TRA). Altitude provides the environmental conditions of the flow coming into the engine as also does the flight Mach, while the TRA is the fraction from maximum amount of fuel injection into the combustion chamber. Twenty-one sensors comprise the output for the CMAPSS simulation, in which data such as temperatures and pressures, both dynamic and total, are registered at different stages of the turbofan. Each of the sensors is identified by a "Dataset ID" as can be observed on Table 1.

In the CMAPSS dataset the 21 output sensors are directly affected by each other, with the operational settings determining the conditions under which the turbofan engine is working. Predicting how the system is going to react under certain conditions, and specially how each of the components behave, can assist in detecting a specific failure in the engine when certain sensors are off from their nominal values. The time series under study belongs to Unit 49 in the "Test 1" file, focusing on the first two operational settings and two compressor sensors as seen in Figure 2. This has been the selected data for the available length, as it provides enough information for the reservoir to parse through and learn on the behaviour of the turbofan engine.

Table 1. Operational settings and output Sensors for the CMAPSS dataset [17].

Symbol	Description	Units	Dataset ID
A	Flight cruise altitude	ft	Operational Setting 1
M	Flight Mach number	-	Operational Setting 2
TRA	Throttle Resolver Angle	%	Operational Setting 3
T2	Total temperature at fan inlet	^{o} R	Sensor 1
T24	Total temperature at LPC outlet	^{o} R	Sensor 2
T30	Total temperature at HPC outlet	^{o} R	Sensor 3
T50	Total temperature at LPT outlet	^{o} R	Sensor 4
P2	Pressure at fan inlet	psia	Sensor 5
P15	Total pressure in bypass-duct	psia	Sensor 6
P30	Total pressure at HPC outlet	psia	Sensor 7
Nf	Physical fan speed	rpm	Sensor 8
Nc	Physical core speed	rpm	Sensor 9
EPR	Engine Pressure Ratio (P50/P2)	-	Sensor 10
Ps30	Static pressure at HPC outlet	psia	Sensor 11
ϕ	Ratio of fuel flow to Ps30	pps/psi	Sensor 12
NRf	Corrected fan speed	rpm	Sensor 13
NRc	Corrected core speed	rpm	Sensor 14
BPR	Bypass Ratio	-	Sensor 15
farB	Burner fuel-air ratio	-	Sensor 16
htBleed	Bleed Enthalpy	-	Sensor 17
Nf_dmd	Demanded fan speed	rpm	Sensor 18
PCNfR_dmd	Demanded corrected fan speed	rpm	Sensor 19
W31	HPT coolant bleed	lbm/s	Sensor 20
W32	LPT coolant bleed	lbm/s	Sensor 21

In Table 2 the hyperparameters bounds are shown. These values depend mainly on the size and the complexity of the time series. The selection of these values have followed the standard criteria. In particular, as already said, the size of the reservoir, N, should be high enough to capture the dynamics of the sensor without compromising computational costs. It is usual for N to be at least equal to the estimate of independent real values the reservoir has to remember from the input to solve its task [7]. Thus a reasonable lower estimate is at least $N = 50$ nodes. Arbitrarily, we established 10 times that quantity, i.e., $N = 500$, as an upper bound. About the input matrix scaling factor, a_{Win}, it needs to be threefold for this dataset. The first factor is for the unitary value appended to the input, the second is for the OPs and the third is for the output sensors. The establishment of bounds followed, [19] allowing the scaling factors to vary wildly between them so the inputs that are more relevant to the reservoir dynamics are present the most. The rest of the bounds have been established arbitrarily.

Table 2. Upper and lower bounds for the hyperparameter optimization of the CMAPSS dataset.

	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β
Lower bound	50	0.01	0.01	0.01	0.01	0.001	0.01	1×10^{-8}
Upper bound	500	0.6	100	100	100	0.3	0.9	100

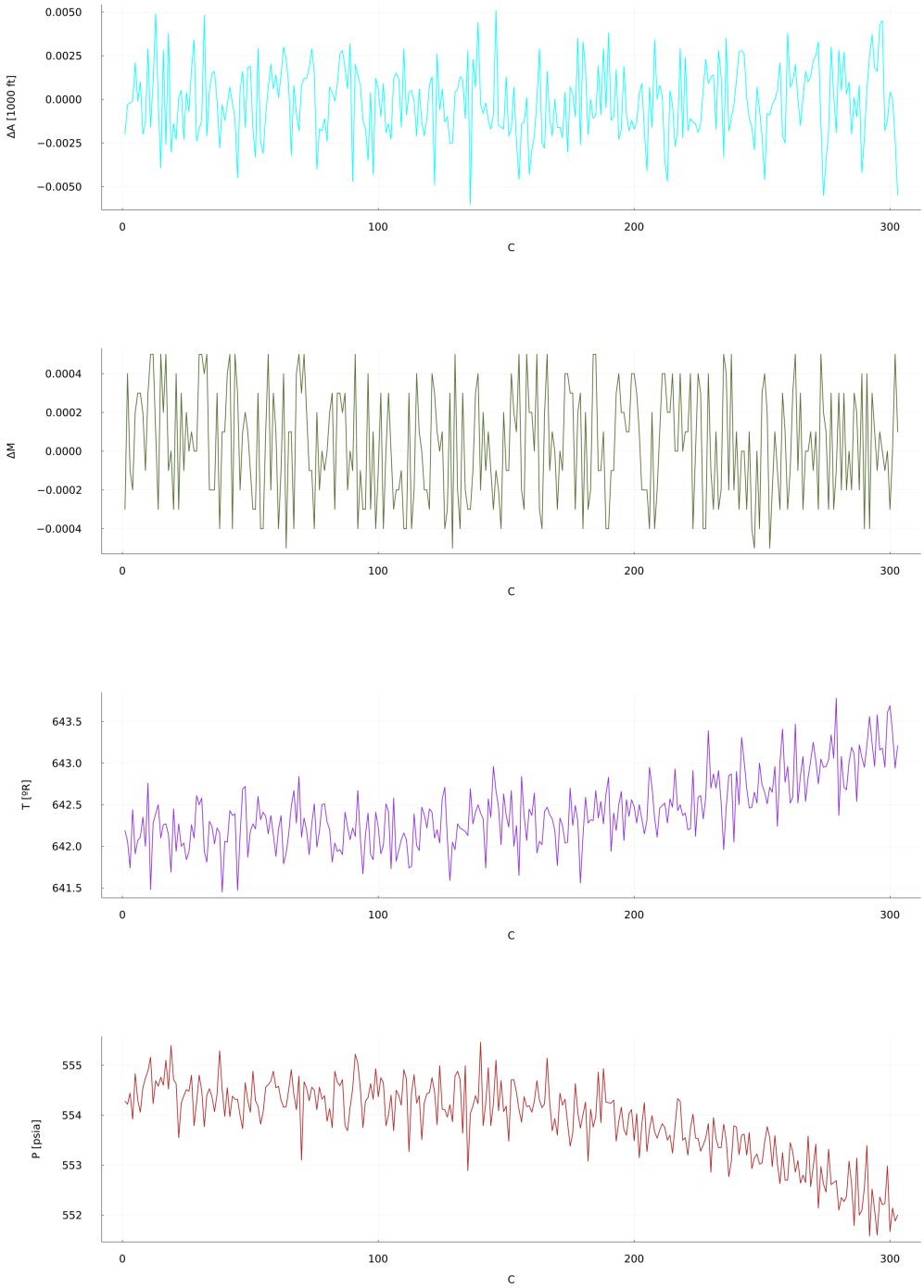


Figure 2. CMAPSS dataset Test 1, Unit 49 (from top to bottom): OS 1: Altitude, OS 2: Mach Number, Sensor 2: Total temperature at LPC outlet, Sensor 7: Total pressure at HPC outlet. Time units are in operating cycles.

3. Results

The CMAPSS dataset grants the option to analyse and minimize the impact of having a limited amount of data, only about three hundred discrete values for each signal. The system has been run for 300 iterations with a population of 50 at each one. The 24 predicted signals (OPs + sensors) differ from the originals, as the reduced amount of training data affects the capacity of the reservoir to track the precise behaviour of the turbofan engine. Nevertheless, some signals are better predicted than others, providing insights into the turbofan engine.

Several runs have been set up for the dataset, to minimize the effect of chance on finding the optimum configuration. From these runs five main sets are of particular notice (shown in Table 3), as most results gravitate towards the viccinity of their hyperparameter sets. The averaged nRMSE over all considered signals is the collective error from the 3 OPs and the 21 sensors, as all are jointly fed to the reservoir and a prediction is generated simultaneously for the 24 time series.

Table 3. RC-GWO Ensemble optimized hyperparameter sets for the CMAPSS dataset. The nRMSE is averaged over the 24 signals.

Set	N	α	$a_{Win}(1)$	$a_{Win}(2)$	$a_{Win}(3)$	c	K_ρ	β	Average nRMSE
1	51	0.43	75.76	73.00	71.65	0.08	0.61	0.92	0.36
2	481	0.39	89.33	91.69	82.83	0.13	0.59	0.39	0.37
3	489	0.36	0.86	7.93	7.79	0.23	0.73	0.89	0.39
4	105	0.40	0.49	8.39	1.89	0.25	0.51	0.89	0.36
5	166	0.57	0.91	7.96	9.30	0.29	0.74	0.42	0.42

Sets 1 and 2 ($S1$ and $S2$) generated the predictions with smaller nRMSE. The stark difference resides in the input scaling factors which is over an order of magnitude higher than in other sets. For $S1$, with the smallest reservoir size, the three folded scaling factors have similar values, meaning the activation functions will give the unitary, OPs and sensor inputs approximately the same weight towards the calculation of the internal states. Not only the a_{Win}'s have the same importance, but actually they take high values forcing the activation functions to behave like binary switches. A similar behaviour has been detected for $S2$, which even with a greater size still suffers from too high input scaling factors which is detrimental to its predictive capabilities.

Thus, in spite $S1$ and $S2$ show smaller averaged nRMSE their reservoir internal states does not guarantee a better individual prediction as, for both of these sets the reservoir internal states are saturated, i.e., most of the neurons yield extreme $|1|$ values (Figure 3 top), instead of being distributed around 0 (Figure 3 bottom). As a consequence the network looses the advantages of the separability granted by the ESP. Reservoir saturation is an aspect that warrants a discussion of its own as it is a phenomenon that severely impacts performance. Saturation is present when the internal states of the reservoir loose scope of the input signal features, which in turn lead to a reduction of its learning capacity. In order to mitigate the chances of this behaviour it is necessary to increase the amount of neurons in the reservoir and limit the input scaling factors to a lesser order of magnitude.

Sets $S3$, $S4$ and $S5$ suggest that reservoir size is not the main hyperparameter acting on the average nRMSE as even though $S3$ is nearly ten times the size of $S1$—and due to the sparsity ten times the computational cost [7]—the error does not reduce proportionately. Due to data shortness increasing the quantity of neurons does not report a decrease in nRMSE, therefore other hyperparameters must be at play to contribute to the prediction error. The leak rate plays the biggest role when generating predictions on limited data. As can be seen on $S3$ to $S5$ a higher value may drive the reservoir into forgetting too quickly the original features as the previous internal states are not as influential when updating.

The input scaling factors are similar, or the same order, for $S3$–$S5$, as the main takeaway is the necessity to attenuate extreme $|1|$ values and augment the actual signals, thus making them the primary drivers of activation. Sparsity does not affect the quality of the predictions as the echo state property can be achieved with fully connected reservoirs. It is known that the only advantage found is the linear increase in computation costs rather than quadratic when increasing the reservoir size.

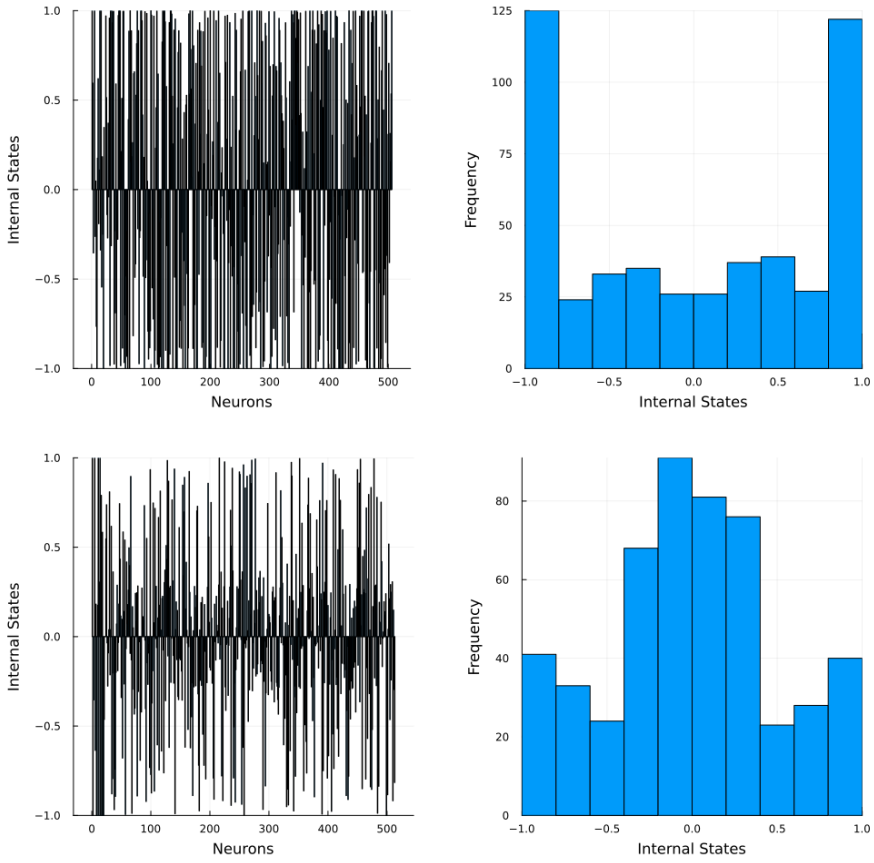


Figure 3. Internal states of a reservoir. **Left**: Internal states for each the neuron. **Right**: histogram for the internal states. Results obtained at an arbitrary step during training. **Top**: A saturated reservoir. **Bottom**: an unsaturated reservoir.

The spectral radius scaling factor is used to enforce the echo state property through establishing the maximal eigenvalue of the internal weight matrix. Even though it can still be achieved with values greater than 1 it is simpler to keep it bounded below it, taking into consideration that it should be greater in tasks that require longer memory. Therefore the scaling factors of *S*3–*S*5 serve as a support to the leak rate, improving the reservoir capacity to retain the time series features given the limited data. As for the regularization coefficient no clear effect has been found and has only served to ensure stability and prevent overfitting.

The generated predictions for *S*3 are shown in Figure 4. It is observed that the overall qualitative behavior of the predicted signal follows the trends of the original signal. In particular, the prediction is bounded between the actual data showing no divergences. The first operational setting at Figure 4 (top) yields an nRMSE of 0.44. Even with such an error, the target initial trend are followed closely up to 10 cycles. It is also noticeable the reservoir capacity to follow the drop at 60 cycles. The Mach signal prediction, shown at Figure 4 (bellow top), has an average nRMSE of 0.31. This results shows the reservoir capacity to approximately mimic the target data with such a limited training. During the initial 20 cycles the predicted signal closely follows the target, staying within the bounds of the original series. Similar coincidences are observed at the 50 cycles mark and between 60 and 80 cycles. As for the last two signals, Sensor 2 at Figure 4 (above bottom) and Sensor 7 at Figure 4 (bottom), the distance between the prediction and the target grows in an apparently linear fashion as time goes, and as the nRMSE values at Table 4 show. Sensor 2 prediction holds close to the original signal during the first 20 cycles but then undershoots, while Sensor 7 overshoots at the first 10 cycles. Both these predictions miss the mark at replicating the trends of the target. The particular set of hyperparameters for

*S*3 generates an stable and feature-rich prediction that can serve as a baseline for future developments. The behavior of the forecasted signal during the first 10 cycles is shown in Figure 5. In the OS 1, OS 2 and Sensor 2 data the predicted signal approximately follows the target with a small delay. However, sensor 7 data forecast clearly differs from the target. For these four situations the calculated nRMSE are 0.52, 0.47, 0.41 and 0.56, respectively.

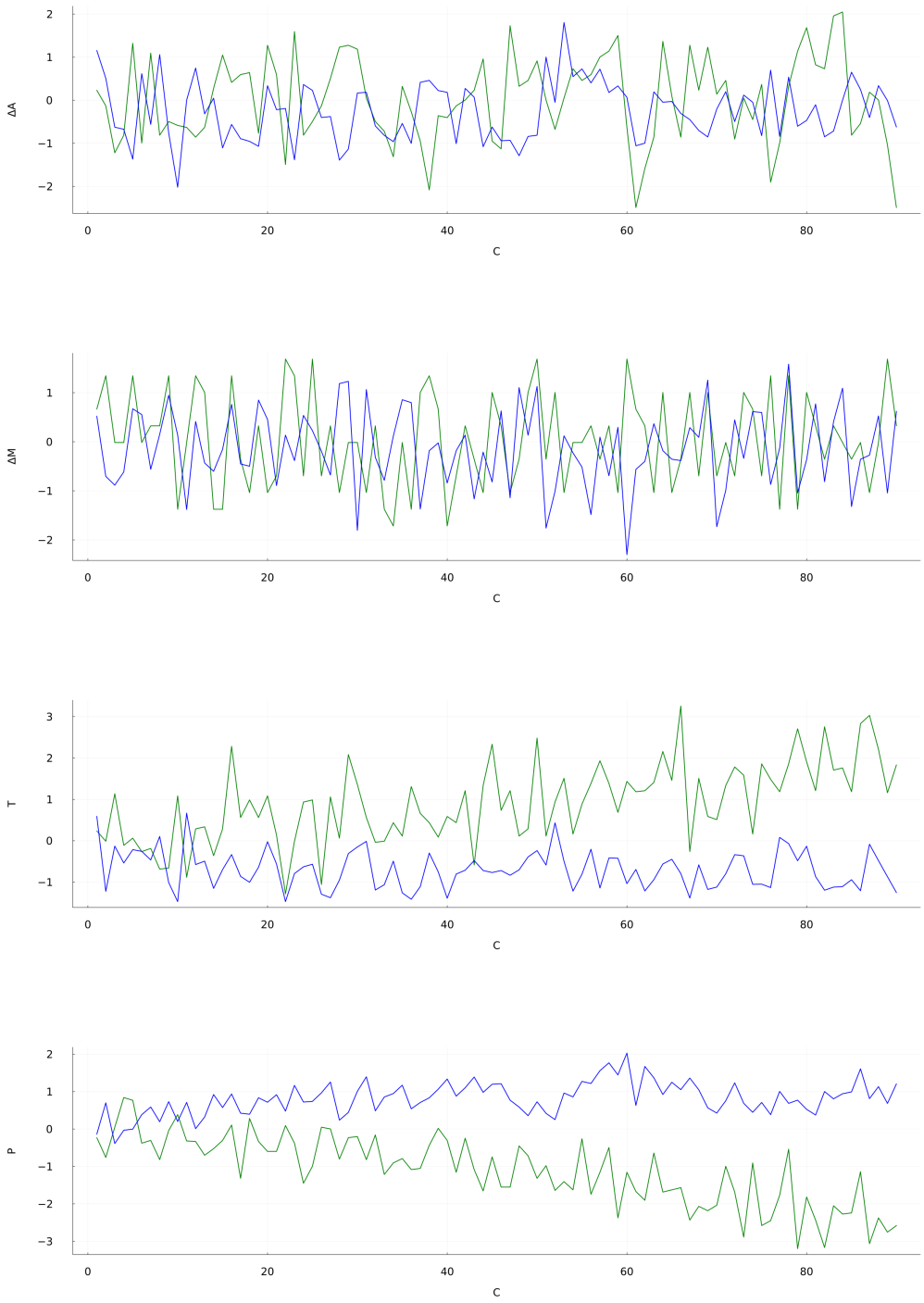


Figure 4. CMAPSS dataset Test 1, Unit 49 time series prediction for Set 3 (from **top** to **bottom**): OS 1: Altitude, OS 2: Mach Number, Sensor 2: Total temperature at LPC outlet and, Sensor 7: total pressure at HPC outlet. Time units are in operating cycles. Green: target signal; Blue: predicted signal. Results for the additional sets are available as Supplementary Materials.

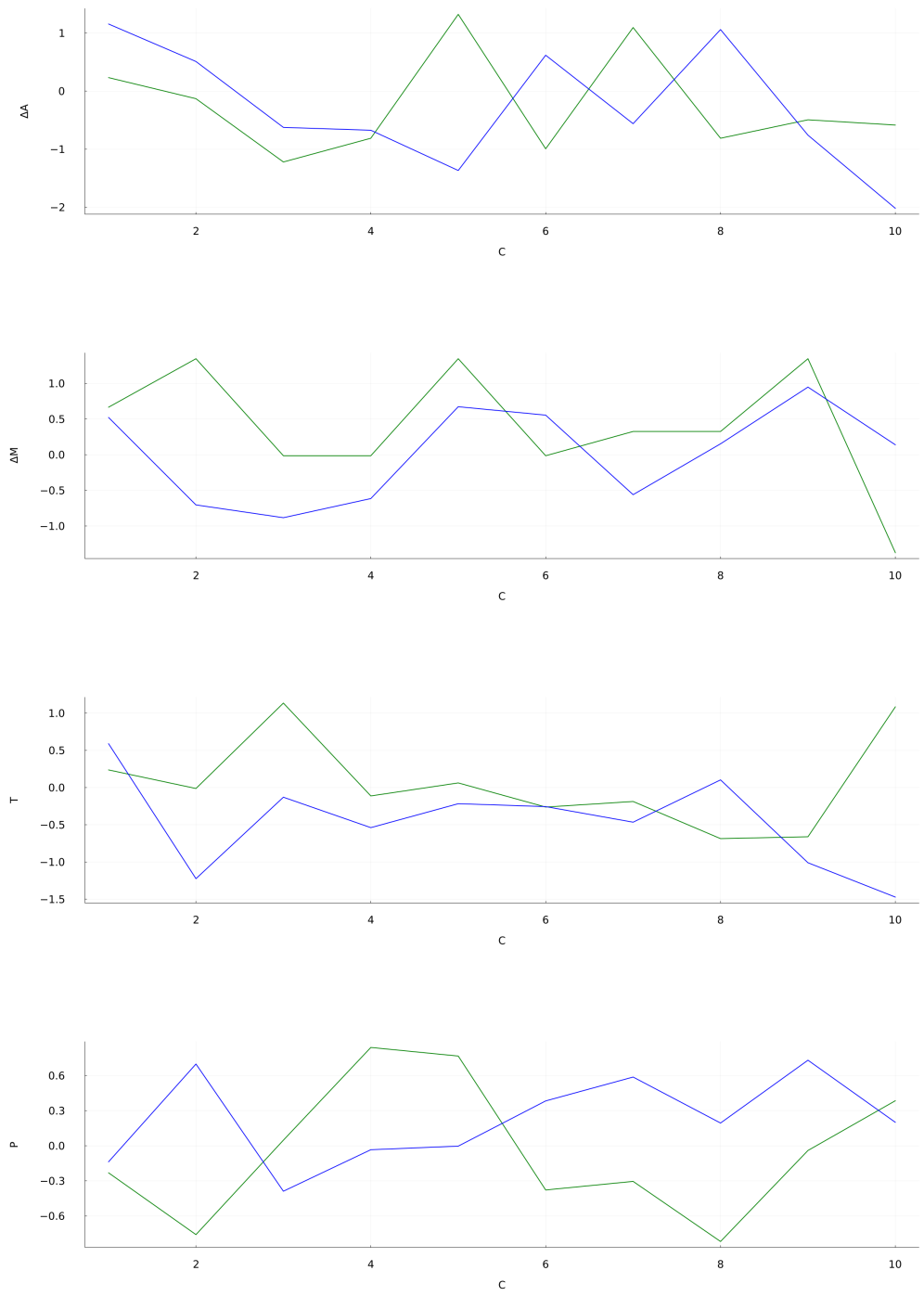


Figure 5. Zoom detail of the forecasting along the 10 initial cycles. Details are the same as in Figure 4.

Table 4. RC-GWO System nRMSE values for the predicted signal values of each of the sets. OS 3 and Signals 1, 5, 10, 16, 17, 18 and 19 remain at a constant value throughout the length of the dataset.

Dataset ID	Set 1	Set 2	Set 3	Set 4	Set 5	Mean
Operational Setting 1	0.35	0.33	0.44	0.39	0.42	0.39
Operational Setting 2	0.48	0.36	0.31	0.34	0.36	0.37
Operational Setting 3	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 1	0.50	0.32	0.43	0.50	0.56	0.46

Table 4. *Cont.*

Dataset ID	Set 1	Set 2	Set 3	Set 4	Set 5	Mean
Sensor 2	0.40	0.48	0.47	0.40	0.51	0.45
Sensor 3	0.36	0.46	0.36	0.38	0.43	0.40
Sensor 4	0.39	0.48	0.52	0.34	0.47	0.44
Sensor 5	0.50	0.32	0.43	0.50	0.56	0.46
Sensor 6	0.36	0.37	0.35	0.38	0.42	0.38
Sensor 7	0.43	0.43	0.53	0.36	0.52	0.45
Sensor 8	0.43	0.47	0.52	0.36	0.54	0.46
Sensor 9	0.41	0.47	0.39	0.38	0.41	0.41
Sensor 10	0.50	0.32	0.43	0.50	0.56	0.46
Sensor 11	0.44	0.44	0.47	0.38	0.52	0.45
Sensor 12	0.39	0.51	0.52	0.42	0.53	0.48
Sensor 13	0.38	0.49	0.53	0.46	0.48	0.47
Sensor 14	0.36	0.44	0.45	0.42	0.47	0.43
Sensor 15	0.34	0.48	0.44	0.43	0.46	0.43
Sensor 16	0.50	0.32	0.43	0.50	0.56	0.46
Sensor 17	0.40	0.43	0.37	0.36	0.50	0.41
Sensor 18	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 19	0.00	0.00	0.00	0.00	0.00	0.00
Sensor 20	0.30	0.47	0.48	0.38	0.42	0.41
Sensor 21	0.40	0.40	0.42	0.37	0.43	0.40

4. Conclusions

The optimized hyperparameter sets show promise for generating predictions on a complicated system such as a turbofan engine. Emulating simultaneously the behaviour of 24 signals on a small reservoir can lead to computationally effective and quick results, generating forecasts with the data at hand. The hyperparameters that configure the reservoir need to be finely tuned and understood. Establishing valid bounds for such a task grants the RC-GWO system a greater chance at finding an stable optimum for the time series under study. The sets are capable of generating stable predictions fluctuating at the same range of values of the target. Notoriously, in contrast to works devoted to chaotic dynamical systems that pay attention to the detailed forecasted signal target tracking [20], the few number of works committed to engineering systems usually summarize their finding using the RMSE or similar measures, without reporting tracking results. In our opinion, the actual tracking capabilities of the forecasted signal must be shown not to mask spurious results.

It should be emphasized that the present approach uses the simplest random reservoir. Improvements could allow to detect out of ordinary doings indicating the need for maintenance. Given the complex behavior of the turbofan engine it is expected that a reservoir tuned in a critical state shall improve the predictive capabilities, as criticality has been shown to be optimal for such complex situations [21]. Future works will consider simple networks configurations where criticality is easily achieved [22].

These early stages aim at developing a tool for a comprehensive evaluation of the entire process. It may help to avoid the black-box characteristics of most Machine Learning solutions so the predictions could be understood, evaluated and accepted minimizing the fear of failure that may well pose a safety risk to people, workers and companies. Transparency in the field of Artificial Intelligence is a requirement for the general well-being, assuring users and authorities that these systems are reliable tools rather than cheap shortcuts with special consideration to the specific necessities of the Aerospace industry.

Supplementary Materials: The following material are available online at https://github.com/JMRRiesgo/RC-GWO-System-Data, Figure S1: CMAPSS dataset Test 1, Unit 49 time series prediction for Set 1; Figure S2: CMAPSS dataset Test 1, Unit 49 time series prediction for Set 2; Figure S3: CMAPSS dataset Test 1, Unit 49 time series prediction for Set 4; Figure S4: CMAPSS dataset Test 1, Unit 49 time series prediction for Set 5.

Author Contributions: Conceptualization, J.L.C.F. and J.M.R.R.; methodology, J.L.C.F.; software, J.M.R.R. and J.L.C.F.; validation, J.L.C.F.; formal analysis, J.M.R.R. and J.L.C.F.; investigation, J.M.R.R.; resources, J.M.R.R.; data curation, J.M.R.R.; writing—original draft preparation, review and editing, J.M.R.R. and J.L.C.F.; visualization, J.M.R.R.; supervision, J.L.C.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting reported results can be found at the author's GitHub repository at https://github.com/JMRRiesgo/RC-GWO-System-Data, accessed on 1 May 2024. The original dataset analyzed during the study can be found at the "Prognostics Center of Excellence Data Set Repository" (NASA) at https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/, accessed on 15 March 2023.

Acknowledgments: The authors would like to thank the NASA Ames Prognostics Center of Excellence, and in particular to A. Saxena and K. Goebel for making available the data used in this study. This study was carried out as part of a JMRR Master thesis.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ESP	Echo State Property
ESN	Echo State Network
GWO	Grey Wolf Optimizer
LNN	Liquid Neural Nets
LSM	Liquid State Machines
nRMSE	normalized Root Mean Square Error
RC	Reservoir Computing
RNN	Recurrent Neural Network

References

1. Zio, E. Prognostics and Health Management (PHM): Where are we and where do we (need to) go in theory and practice. *Reliab. Eng. Syst. Saf.* **2022**, *218*, 108–119. [CrossRef]
2. Brunton, S. Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning. *AIAA J.* **2021**, *59*, 2820–2847. [CrossRef]
3. Carvalho, T. A systematic literature review of machine learning methods applied to predictive maintenance. *Comput. Ind. Eng.* **2019**, *137*, 106024. [CrossRef]
4. Gao, C. Medium and Long-Term Fault Prediction of Avionics Based on Echo State Network. *Mach. Learn. Deep. Learn. Optim. Tech. Heterog. Sens. Inf. Integr.* **2023**, *2022*, 5343909. [CrossRef]
5. Rigamonti, M.; Baraldi, P.; Zio, E.; Roychoudhury, I.; Goebel, K.; Poll, S. Echo State Network for the Remaining Useful Life Prediction of a Turbofan Engine. *PHM Soc. Eur. Conf.* **2016**, *3*, 5343909. [CrossRef]
6. Márquez, M.R.; Gutiérrez, E.D.; Álvarez, J.S.M.; Milton, J.G.; Cabrera, J.L. Machine learning forecasting of extreme fluctuations in a human balancing task. *Knowl.-Based Syst.* **2023**, *280*, 111000. [CrossRef]
7. Lukoševičius, M. A practical guide to applying echo state networks. *Lect. Notes Comput. Sci.* **2012**, *7700*, 659–686. [CrossRef]
8. Konkoli, Z. Reservoir Computing. In *Encyclopedia of Complexity and Systems Science*; Springer: Berlin/Heidelber, Germany, 2017. [CrossRef]
9. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. In *German National Research Center for Information Technology GMD Technical Report*; GMD: Bonn, Germany, 2001; Volume 148, p. 13.
10. Maas, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **2002**, *14*, 2531–2560. [CrossRef] [PubMed]
11. Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; Grosu, R. Liquid Time Constant Networks. *arXiv* **2020** arXiv:2006.04439.
12. Pathak, J.; Hunt, B.; Girvan, M.; Lu, Z.; Ott, E. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys. Rev. Lett.* **2018**, *120*, 024102. [CrossRef]

13. Cai, H.; Ao, Z.; Tian, C.; Wu, Z.; Liu, H.; Tchieu, J.; Gu, M.; Mackie, K.; Guo, F. Brain organoid reservoir computing for artificial intelligence. *Nat. Electron.* **2023**, *6*, 1032–1039. [CrossRef]
14. Martinuzzi, F. A Brief Introduction to Reservoir Computing. Available online: https://martinuzzifrancesco.github.io/posts/a-brief-introduction-to-reservoir-computing/#3 (accessed on 1 May 2024).
15. Jaeger, H. Echo state network. *Scholarpedia* **2007**, *2*, 2330. [CrossRef]
16. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
17. Saxena, A.; Goebel, K.; Simon, D.; Eklund, N. Damage propagation modeling for aircraft engine run-to-failure simulation. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008. [CrossRef]
18. Prognostics Center of Excellence Data Set Repository—NASA. Available online: https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/ (accessed on 4 March 2024).
19. Xu, M.; Baraldi, P.; Al-Dahidi, S.; Zio, E. Fault prognostics by an ensemble of Echo State Networks in presence of event based measurements. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103346. [CrossRef]
20. Lu, Z.; Pathak, J.; Hunt, B.; Girvan, M.; Brockett, R.; Ott, E. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos* **2017**, *27*, 041102. [CrossRef]
21. Mandal, S.; Shrimali, M.D. Achieving criticality for reservoir computing using environment-induced explosive death. *Chaos* **2021**, *31*, 031101. [CrossRef]
22. Cabrera, J.L.; Hoenicka, J.; Macedo, D. A Small Morris-Lecar Neuron Network Gets Close to Critical Only in the Small-World Regimen. *J. Complex Syst.* **2013**, *1*, 675818. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.