CERN-ACC-NOTE-2023-0007

30 May 2023

A.MenorDeOnate@student.tudelft.nl

Algorithms for Tune Estimation and Damper Control

Adrián Menor de Oñate
CERN, CH-1211 Geneva, Switzerland

Keywords: Synchrotron, Transverse Tune, Transverse Feedback, Machine Learning, System Identification, Model Predictive Control, Feedback Algorithms

Summary

This note explores different methods of extracting the tune from transverse feedback and pick-up data streams. Advanced algorithms such as the extended Kalman Filter, techniques employing machine learning algorithms and the Fourier synchrosqueezed transform (FSST) are compared. Reinforcement learning is used to propose an alternative to the usual proportional feedback controller for the transverse feedback systems and performances are compared. Lastly, model predictive control (MPC) is used in combination with online identification of the accelerator's dynamics, and compared with the aforementioned proportional controller.

European organisation for Nuclear Research

Algorithms for Tune Estimation and Damper Control

Author:

Adrián Menor de Oñate [1][2]

Supervisor:

Gerd Kotzian [3]

May 29, 2023

[1]adrian.menor.de.onate@cern.ch
[2]A.MenorDeOnate@student.tudelft.nl
[3]gerd.kotzian@cern.ch

Preface

This report describes the works done as a CERN Technical Student, in the Geneva area, Switzerland, for the duration of fourteen months; from the first of March 2022 to the end of April 2023. I would like to thank my supervisor Gerd Kotzian for trusting me with this internship position, guiding me throughout the internship, and providing a great deal of knowledge that I will take with me moving forward in my career. Lastly, I would like to thank all the members from the RF group, as well as the members of the ML community forum, in particular, Joel Axel Wulff, Verena Kain and Michael Schenk, with whom I had fruitful discussions that helped me during my internship.

Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Description	Acronym	Description
AM	Amplitude Modulation	ML	Machine Learning
A2C	Advantage Actor-Critic	PM	Phase Modulation
BBQ	Base-Band-Tune algorithm	PPO	Proximal Policy Optimisation
CERN	European Organisation for Nuclear Research	PU	Pick-up
DDPG	Deep Deterministic Policy Gradient	RL	Reinforcement Learning
DQN	Deep Q Network	RNN	Recurrent Neural Network
EKF	Extended Kalman Filter	SAC	Soft Actor Critic
FFT	Fast Fourier Transform	SNR	Signal-to-noise Ratio
G	Feedback gain	SPS	Super Proton Synchrotron
LHC	Large Hadron Collider	TD3	Twin Delayed DDPG
LSTM	Long short-term memory		

List of Symbols

Symbol	Unit	Description	Symbol	Unit	Description
B	Tesla	Magnetic field	ρ	meters	Bending radius
e	Coulomb	Electric charge	σ	units	Standard deviation
G	-	Gain	θ	Radians	Rotation angle
Q_b	Hz	Transverse tune	μ	-	Octupole factor
Q_L	Hz	Longitudinal tune	ξ	-	Chromaticity
v	m/s	Velocity			

Abstract

This report describes the process and results of the different tasks carried out as a CERN engineer intern in the Feedbacks and Beam Control section.
CERN, the European Organisation for Nuclear Research, investigates fundamental questions in particle physics. To do so, particle accelerators and colliders accelerate beams of particles to then collide them against each other, or against a fixed target. To successfully carry out acceleration and collisions, it is of paramount importance to appropriately control the particle beams. Particularly, the transverse oscillations (oscillations in the plane perpendicular to the particle's direction of travel) need to be damped to obtain a stable beam. These oscillations are characterised by the so-called betatron tune; the number of betatron oscillations done by a particle bunch over one turn. Correctly tracking and controlling the transverse tune evolution is crucial to provide good quality beams to the Large Hadron Collider (LHC), and other accelerators and experiments.

Measuring the tune can be challenging, given that the amplitude or frequency data of the beam's oscillations can be polluted with noise or artefacts. This report has the objective of exploring the potential of advanced algorithms such as machine learning in determining the transverse tune from recorded data. Furthermore, reinforcement learning control of the transverse dynamics of the accelerators is also investigated.

To measure the tune, three methods are investigated. The first method involves reconstructing the true amplitude of the oscillation data from the noisy measurements, to then estimate the tune, by using an Extended Kalman Filter (EKF). The EKF is verified with simulated data, and then validated with real Super Proton Synchrotron (SPS) data.
The second method uses machine learning models to estimate the tune from the frequency domain data of the particle oscillations. The following machine learning models are used: feed forward artificial neural networks (FF), convolutional neural networks (CNN), recurrent neural networks (RNN), and a combination of the different models; the CNN-RNN.
The third method uses synchrosqueezing techniques to sharpen the tune spectrograms. In particular, the Fourier synchrosqueezed transform (FSST) is used.

The EKF algorithm successfully tracks the tune with simulated data, until the oscillations are damped, where the error induced by converting from amplitude modulation data to phase modulation data increases the variance in the estimations of the tune. Moreover, the EKF successfully tracks the tune only in isolated regions of the measured accelerator data. This occurs due to inaccuracies in the accelerator dynamics model used in the EKF to estimate the tune.

Regarding the machine learning methods, all methods achieve tune estimation accuracy similar to the used human labels, used as ground truth, when training on spectra with constant tunes. Furthermore, the CNN-RNN model shows the best performance when a diversity of operational beams are used for training and validation, closely matching human performance, albeit some outliers.

The FSST algorithm helps sharpen the conventional BBQ spectrogram, reducing the tune line to fewer frequency bins and heavily reducing the noise of the spectrogram.

The reinforcement learning controller is implemented by using Soft Actor Critic (SAC) agents in two different scenarios; controlling the gain of a proportional controller, and directly kicking the beam. The agents observe the amplitude oscillation of the beam and apply a transverse kick on it to damp the oscillations. Furthermore, this agents control a simulated accelerator during training and validation, where their performance is compared against a proportional controller.

Both agents successfully damp the beam, although with similar performance to that of the proportional controller.

Finally, a model predictive controller with online system identification capabilities is shown to outperform the proportional controller when damping the transverse beam oscillations in simulation.

1 Introduction

This report describes the works carried as a CERN engineer intern in the Feedbacks and Beam Control section. CERN, the European Organisation for Nuclear Research, utilises particle accelerators and colliders to investigate fundamental questions in particle physics. Usually, the experiments carried out at CERN involve colliding a high-energy particle beam against another, or against a fixed target. To correctly achieve such collisions, it is of paramount importance to control the particle beam appropriately. One of such control goals is to correctly damp any transverse (that is, in the plane perpendicular to the particles' direction of travel) oscillation that the particle beam may have. Specifically, the particles tend to oscillate at an oscillation frequency described by the so-called betatron tune.

During the internship, the objective has been to **investigate new algorithms to detect the betatron tune**, of which an accurate detection is essential to properly dampen the particles' oscillations in the transverse plane and improve the beam quality. CERN's Feedbacks and Beam Control Section is thus heavily interested in improving the accuracy of the tune detection.
Particularly, tune detection is attempted by implementing a well-known algorithm used for state estimation (the Extended Kalman Filter), and also making use of machine learning techniques. Moreover, **new ways of controlling the particle beam have been investigated by testing reinforcement learning and model predictive controllers**. Thus four main research objectives are established; implementing the Extended Kalman Filter and estimating the tune in time series data, implementing machine learning tune estimation algorithms and correctly identifying the tune in spectral data, investigating alternative algorithms to post-process the spectral data, and successfully damping beam oscillations with reinforcement learning and model predictive control.

The report is structured as follows. Firstly, section 2 explains the background information needed to understand the betatron tune, the challenges of measuring it, and describes the research objectives. Thirdly, the methodology and results for each of the four main research topics are shown in sections 3, 4, 5, 6, and 7. Then, the conclusion is discussed in section 8, and the future recommendations in section 9.

2 Problem Statement and Background Information

In this report, work has been made to estimate the betatron (transverse) tune Q_b in the CERN accelerators. This chapter discusses the background information needed to understand the transverse motion of particles in an accelerator (and why the tune is important in this regard), what the tune is, as well as the current methods used to estimate the tune (and the challenges that come with it). This is done in accordance to S. Baird's *"Accelerators for pedestrians"* [1]. Finally, the research objective is described together with the steps taken to create new algorithms to estimate Q_b.

2.1 Background: Transverse beam dynamics

In a circular accelerator, the particles are steered by using dipole magnets [1]. This is illustrated in Figure 1, where the magnetic field induces a centripetal force that makes the particle follow a circular path. For applications in large accelerators, several magnets are used to steer the particles, and since the magnets have a fixed geometry with bending radius ρ, the magnetic field B needs to increase to accelerate the particles towards higher velocities within the accelerator [1]. Thus, the physical bending radius of the magnet (for a given maximum B) poses a physical constraint to the maximum momentum that the particles can achieve, and is one of the reasons why CERN has built accelerators of increasing size over time.

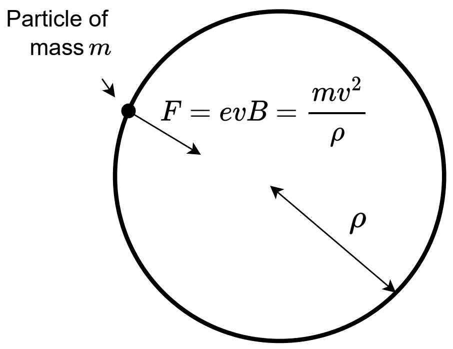


Figure 1: Motion of a particle in a dipole magnetic field, the field is in/out of the page [1].

Moreover, accelerators have bunches of particles, and each particle can start its motion from a slightly different initial horizontal angle. This causes them to follow different paths across a dipole magnet, as shown in the left of Figure 2. As a result, particles oscillate with respect to each other (right of Figure 2). This is called a **Betatron Oscillation** and is the foundation of the transverse dynamics in an accelerator [1].

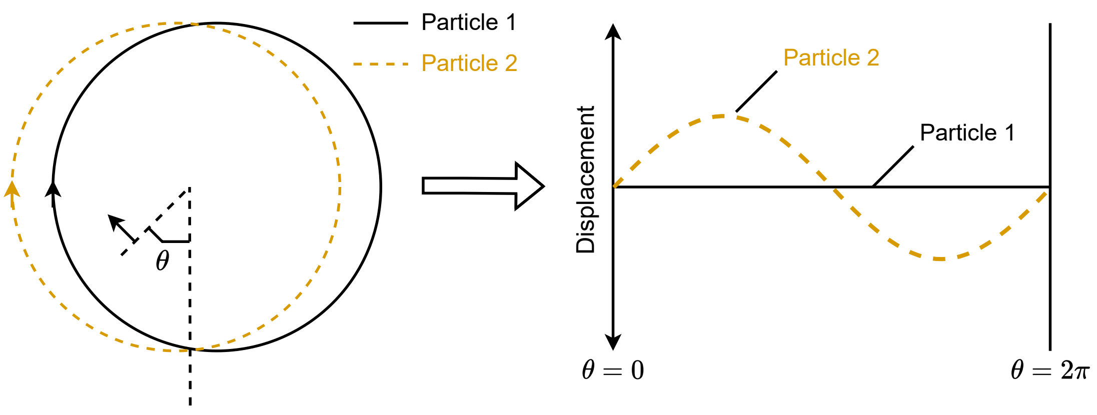


Figure 2: Left: two particles in a dipole field, with different initial angles. Right: Transverse oscillation of the second particle around the first.

So far, only the horizontal motion of the particles has been considered. When looking at the vertical plane, if the particles start with different initial vertical angles, the displacement plot shown in Figure 2 turns into *"a spiral that never closes. Therefore although particle motion in our simple dipole magnet is "stable" i.e. the* *trajectories close in the horizontal plane, it is "unstable" i.e. different trajectories do not close in the vertical plane"*[1, p. 22]. Thus, a different kind of focusing (forcing the particles to follow the ideal trajectory on the accelerator, that is, perfectly centred) is needed other than the dipoles. This is why quadruple magnets are used. These magnets focus the particles in one plane but defocus them in the other [1]. To compensate for the undesired defocusing, quadrupole magnets are placed rotated 90° with respect to each other along the accelerator; focusing and defocusing the horizontal and vertical planes in alternate fashion (see Figure 3).

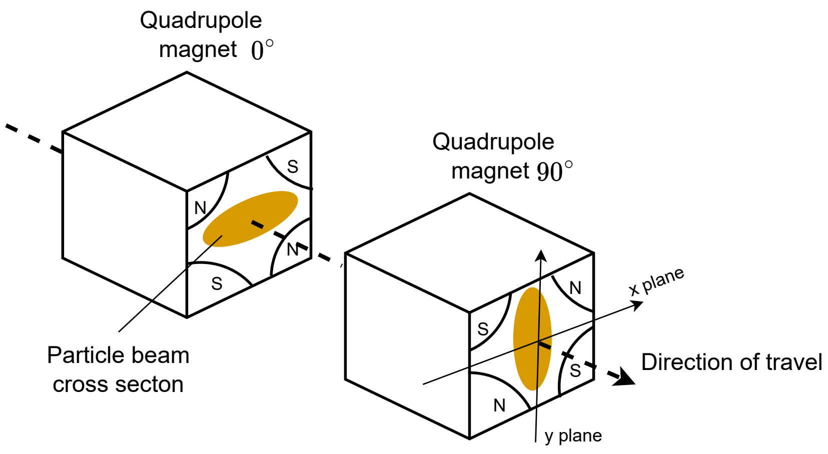


Figure 3: Conceptual impression of two quadrupole magnets focusing and defocusing the beam in different planes.

Thus, in essence, the CERN circular accelerators use dipole magnets to steer the particles in an approximately circular path, while using quadrupole (and other higher order magnets, such as octupoles) magnets to contain the particle beam in the horizontal and vertical (x and y) planes [1]. As the particles diverge from the central path in the transverse planes (shown in Figure 3), the quadrupoles "force" them back to the central trajectory [1]. This makes them oscillate with respect to the central trajectory along the accelerator length s, as shown on the left of Figure 4. In this manner, the **Betratron Tune** Q_b is defined as the number of betatron oscillations per turn [1], which consists of an integer number of oscillations, and a fractional number of oscillations. Here, the motion of the particle is parameterised by using the displacement x from the central path as well as the angle displacement x' of the particle [1].

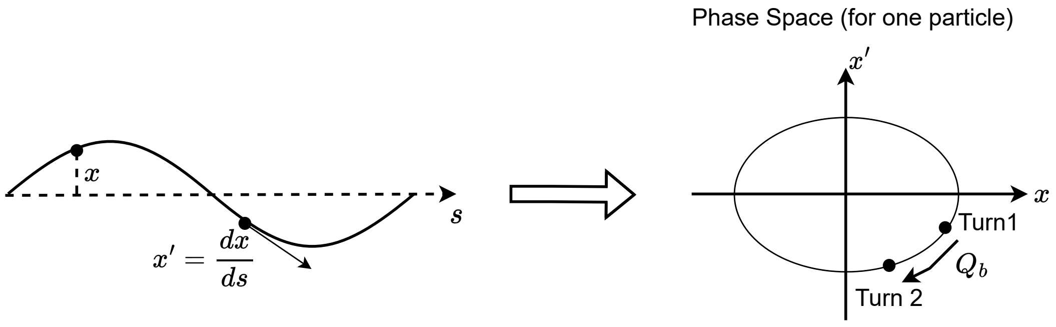


Figure 4: Left: Transverse position and angle displacement of a particle moving through the accelerator [1]. Right: Phase space diagram (measured every turn) of a single particle, with the highlighted effect of Q_b.

In the accelerators, the average x and x' of a particle bunch is measured every turn at the same location by using Pick Ups (PU) measurement devices. This measurement can be used to construct a phase space plot (right part of Figure 4). By using the measurements of a single Pick Up, only the fractional part of the tune can be measured, as there is ambiguity with respect the integer amount of oscillations done by the bunch. In this context, the (fractional) betatron tune (referred to as Q or Q_b in this report) is defined as the phase advance (in the phase space diagram) done every turn by the particle bunch. This is an important parameter for describing the transverse dynamics of a particle bunch. Particularly, the FB section uses kickers (which can be seen as control actuators to change the transverse position of the particle bunch) to damp the betatron oscillations. Thus, knowing (and suppressing) the tune is crucial for obtaining an adequate damping (that is, to ideally reach the point $(0, 0)$ in the phase space). Finally, it is important to realise that there are two different betatron tunes; horizontal and vertical, corresponding to the two different transverse planes, as shown in Figure 3.

2.2 Background: Measuring the tune, and the challenges that come with it

In theory, the tune is estimated by gathering the transverse amplitude data from the PU (i.e. x in Figure 4) for a certain amount of turns. This time-domain data can then be Fourier transformed to detect the tune in the frequency domain, as it will be the frequency with the largest magnitude. In practice, measuring the tune proves to be more challenging than simply taking the *max* operator of the Fourier spectrum.
Firstly, the time domain signal contains artefacts from errors in the accelerator magnets, as well as other dynamics occurring in the particle beam (bunch to bunch wake interactions, etc). This, combined with the bias and errors in the PU measurement devices, results in a signal that contains noise. Furthermore, since the transverse amplitude of the particles is damped by the feedback systems (which also add broadband noise [5]), the signal-to-noise ratio is greatly reduced. When the Fourier analysis is performed, all these effects result in a frequency spectrum containing noise, and sometimes peaks other than the tune (coupling between the horizontal and vertical tunes can also occur, leading to tune peaks corresponding to the other transverse plane). Detecting the tune in such a spectrum is thus non-trivial, and requires algorithms more advanced than the aforementioned *max* operator.

One possible approach is to actively excite the particle beam to make the tune peak visible above the background noise [5], however, this is an invasive method that can also create emittance blow-up (emittance is proportional to the area of the ellipse of the phase-space diagram, see Figure 4) [5]. A second option is to average the frequency spectrum over different particle bunches [5], greatly increasing the tune resolution, although, if bunch-by-bunch tunes are to be estimated, a different algorithm is needed. Finally, the Base-Band-Tune *BBQ* algorithm [9], which uses a PU *"followed by a diode-based detection and acquisition system"*[9, p. 1], is used at CERN. Although generally providing accurate measurements, the *BBQ*'s signal accuracy has been a concern during the LHC ramp mode [5], as well as other operation modes. Furthermore, the algorithm needs a narrow frequency window in the Fourier spectrum in which to search for the tune, thus greatly limiting its flexibility in beams where the tune changes drastically over time.

2.3 Research objective

At the beginning of the internship, the research objective was established:

"The hardware deployed for the transverse feedback systems in the CERN synchrotrons permits recording of long time series of oscillation data of the individual bunches of a beam. Several techniques are available to extract the transverse tune from these data. These include techniques using observation of the residual oscillations during and after injection of the beam or following a kick. ***The project work will explore the potential of advanced algorithms such as machine learning in determining the transverse tune from recorded data****. The project work gives the opportunity to test the invasive algorithms during machine developments and process data online during normal operating periods. Matching of algorithms to the particularities of the accelerator is also part of the study. Machine developments focus on the PSB, PS and SPS accelerators with data also being available from LHC."*

It should be noted that together with the tune estimation task, **it was deemed that investigating reinforcement learning techniques applied on transverse feedback systems was also of interest, as well as online model predictive control approaches**.

3 Model Based Tune Estimation: Extended Kalman Filter

The first step in the investigation of tune measurement algorithms was to consider methods that include physical knowledge about the particle accelerator's dynamics. With this in mind, the Extended Kalman Filter (EKF) algorithm was implemented. In this section, the EKF algorithm is explained in subsection 3.1. Secondly, the algorithm is implemented, as explained in subsection 3.2. Finally, the algorithm is verified and validated in subsection 3.3.

3.1 The Extended Kalman Filter (EKF)

The Extended Kalman Filter is used to estimate the true states of a system in the presence of uncertainty. The EKF is the version of the Kalman Filter [17] used in systems where the dynamics (state and observations) are non-linear [14]. The EKF works by linearising the system dynamics around the last filtered estimate, then applying the prediction step of the Kalman filter to the linearised system dynamics just obtained. The full EKF algorithm is shown in Algorithm 1.

Furthermore, the EKF is implemented using **time domain oscillation data** from the PUs.

Algorithm 1 Extended Kalman Filter

```
1: while running do
       $\hat{\boldsymbol{x}}_{k+1|k} = \mathrm{f}(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{u}_k)$                    ▷ One-step ahead prediction
       $\hat{\boldsymbol{z}}_{k+1} = \mathrm{h}(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{u}_{k+1})$                 ▷ Predicted observation
       $\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_{k+1|k}\boldsymbol{P}_{k|k}\boldsymbol{F}_{k+1|k}^{\top} + \boldsymbol{Q}$        ▷ Covariance matrix of state prediction error
       $\boldsymbol{S}_{k+1} = \boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top} + \boldsymbol{R}$            ▷ Covariance matrix of innovation
       $\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top}\boldsymbol{S}_{k+1}^{-1}$                  ▷ Kalman gain calculation
       $\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{K}_{k+1}[\boldsymbol{z}_{k+1} - \hat{\boldsymbol{z}}_{k+1}]$   ▷ Optimal state
       $\boldsymbol{P}_{k+1|k+1} = (\boldsymbol{I} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{k+1})\boldsymbol{P}_{k+1|k}$         ▷ Updated covariance matrix
       $\boldsymbol{P}_{k|k} \leftarrow \boldsymbol{P}_{k+1|k+1}$
       $\hat{\boldsymbol{x}}_{k|k} \leftarrow \hat{\boldsymbol{x}}_{k+1|k+1}$
2: end while
```

3.2 Algorithm implementation

In order to implement the EKF, the phase space diagram of a particle bunch (similar to the one depicted in Figure 4) is constructed from the oscillation data obtained from the PUs. Then, the phase changes in the aforementioned phase space are used to estimate the tune, i.e. the tune is regarded as the main contributor of a change in phase.

Thus, to estimate Q_b from the PU measurements, a model of the transverse system, observations of the states, and its noise characteristics (both from the system and measurements) is needed. The transverse phase-space dynamics of a bunch of particles are modelled as having a rotation component caused by the tune Q_b, and damping caused by the feedback system (see Equation 1). The feedback is assumed to have a constant gain. Furthermore, the states of the system are defined as the readings from the PUs ($\overline{\boldsymbol{x}} = [PU1, PU2]^T = [x, x']^T$). This way, the next state of the system can be calculated by stating $\overline{\boldsymbol{x}}_{k+1} = \mathbf{F}\overline{\boldsymbol{x}}_k$.

$$\mathbf{F} = \begin{bmatrix} \cos(2\pi Q_b) & -\sin(2\pi Q_b) \\ \sin(2\pi Q_b) & \cos(2\pi Q_b) - G \end{bmatrix} \tag{1}$$

Also, the states of the system are measured directly from the PUs, thus $\mathbf{H}$ is the identity matrix ($\overline{\boldsymbol{z}} = \mathbf{H}\overline{\boldsymbol{x}}$). Finally, the noise characteristics of the system are modelled to have a bias term, and a Gaussian random noise distribution. The covariance matrices of the system and measurement noise are shown in Equations 2 and 3 .

$$\mathbf{Q} = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix} \tag{2}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix} \tag{3}$$

With all the aforementioned definitions, an EKF routine can be constructed to estimate Q_b. The EKF is sensitive to changes in the parameters of the model, thus its prediction accuracy decreases if the real Q_b greatly differs from the estimated one. To prevent this, a crude Q_b update is performed for every measurement done by the system, as shown in Figure 5. This effectively results in a joint state-parameter estimator. Moreover, Q_b is estimated by using the current and previous states, and inverting the system dynamics described in Equation 1.

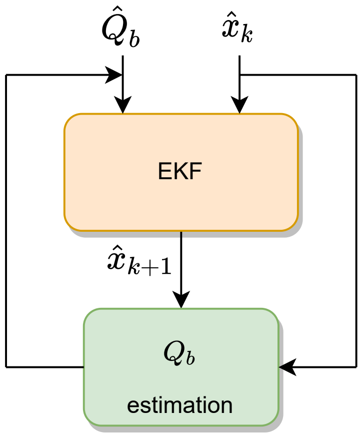


Figure 5: Joint state-parameter estimation routine.

3.3 Results

To asses the EKF algorithm, verification and validations analysis were done to make sure that the EKF accelerator model is implemented in the correct way, and that the EKF algorithm is correct enough when analysing real data.

3.3.1 EKF Verification

To ensure that the EKF algorithm is implemented properly, its performance can be analysed using simulated data with the same dynamics as in Equation 1, and known system and measurement noise statistics. Two scenarios are analysed, namely, simulating constant and varying tunes. Figure 6 shows the constant tune simulation, where the EKF starts by assuming an incorrect tune of 0.3, and later converges to the true tune value, after 30 turns (see bottom-right plot). Also, the EKF's PUs measurements estimations are not accurate when the estimated tune greatly differs from the true tune, as expected, since its model of the dynamics is inaccurate. Finally, when the signal from the PUs damps out, the tune prediction destabilises (as can be seen after 350 turns). This is due to several reasons. Firstly, when the SNR figure deteriorates, the EKF estimation is less accurate. Secondly, this state-parameter estimation routine effectively does an AM to PM conversion. For small signal amplitudes, this makes the EKF very sensitive to small changes to the amplitude of the signal for calculating Q_b, thus small errors in the estimation of the amplitude of the PU signals lead to large phase variations.

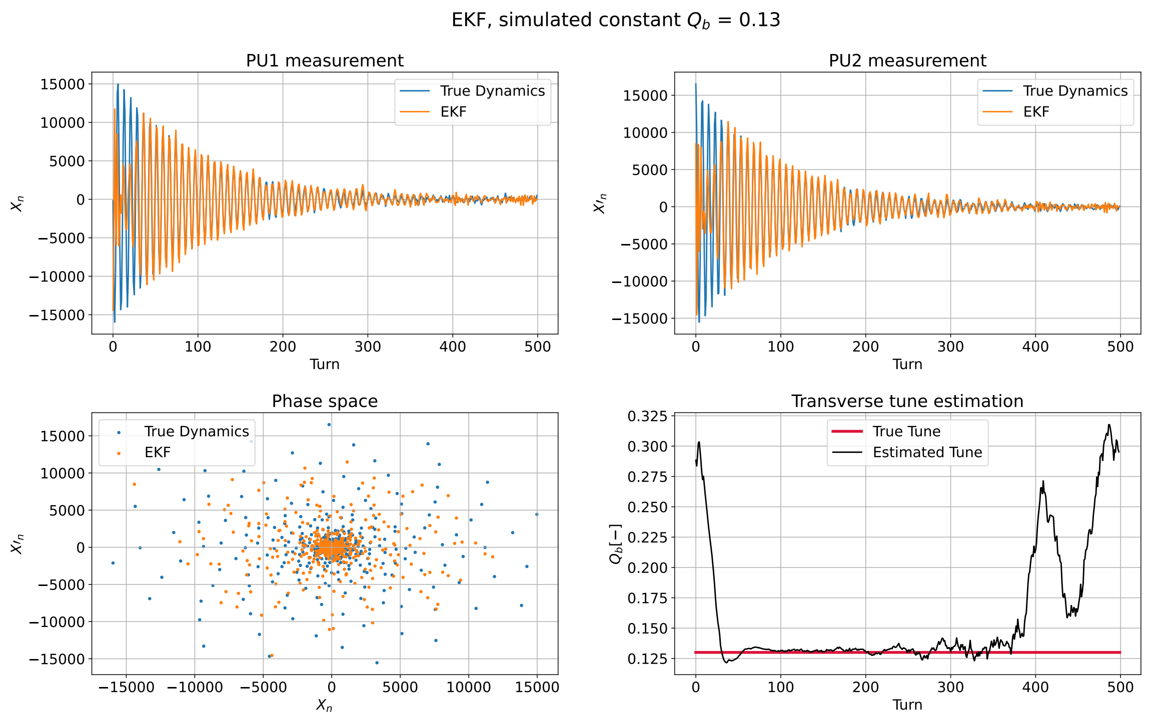


Figure 6: EKF results with simulated constant tune.

Figure 7 shows a similar simulation, but with a changing tune. Here, it is observed that the EKF successfully tracks changes in tune. The time delay when adjusting to the new tune value occurs due to the window size used to average the tune. Reducing the window size decreases the time delay, with the trade-off of increasing the variance in the prediction of the tune.

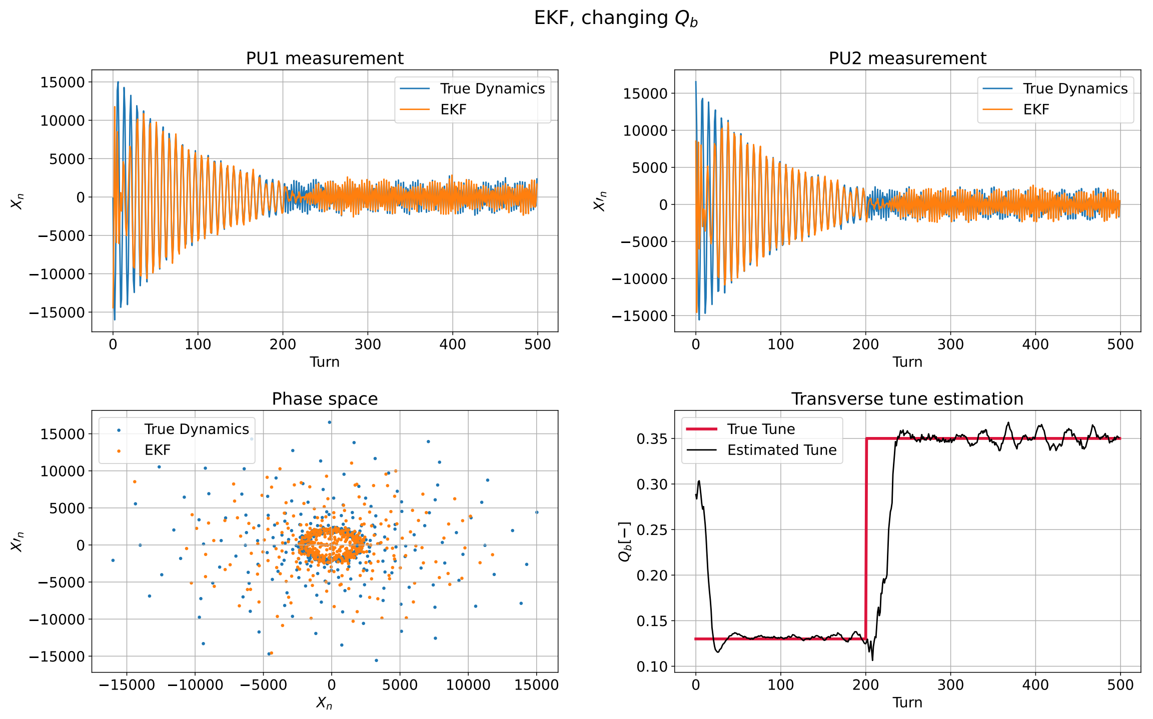


Figure 7: EKF results with simulated changing tune.

3.3.2 EKF Validation

To validate the tune algorithm, real transverse oscillation data has been used from the SPS accelerator. This is shown in Figure 8, where the PU data is compared with the EKF estimation, and the average tune of the beam is set against the EFK. Here, it is observed that the tune prediction rapidly goes from the initial guess, to values close to the actual average tune, to then correctly estimate the tune for roughly 150 turns. After 300 turns, the EKF tune estimation increasingly diverges from the true tune once the oscillation data is damped.

This occurs for several reasons. By inspecting the data, it is clear that the model of the accelerator dynamics is not accurate enough. This is specially evident in the PU1 data, where the EKF wrongly estimates the true oscillation amplitude, sometimes with errors larger than 100%. Since the model of the accelerator only included the phase shifts due to the tune, not including a complete model of the accelerator accounting for the succession of dipole and quadrupole magnets (and other effects, such as chromaticity), this is expected. Furthermore, the divergence of the tune prediction after 300 turns is caused by the aforementioned inaccuracy in the modelling of the accelerator, combined to the AM to PM induced error discussed in subsubsection 3.3.1. The AM to PM error source was considered a hard limit for this algorithm in the current set up, and thus a shift to Machine Learning methods was investigated (see section 4). Nevertheless, possible algorithm modifications are mentioned in section 9.

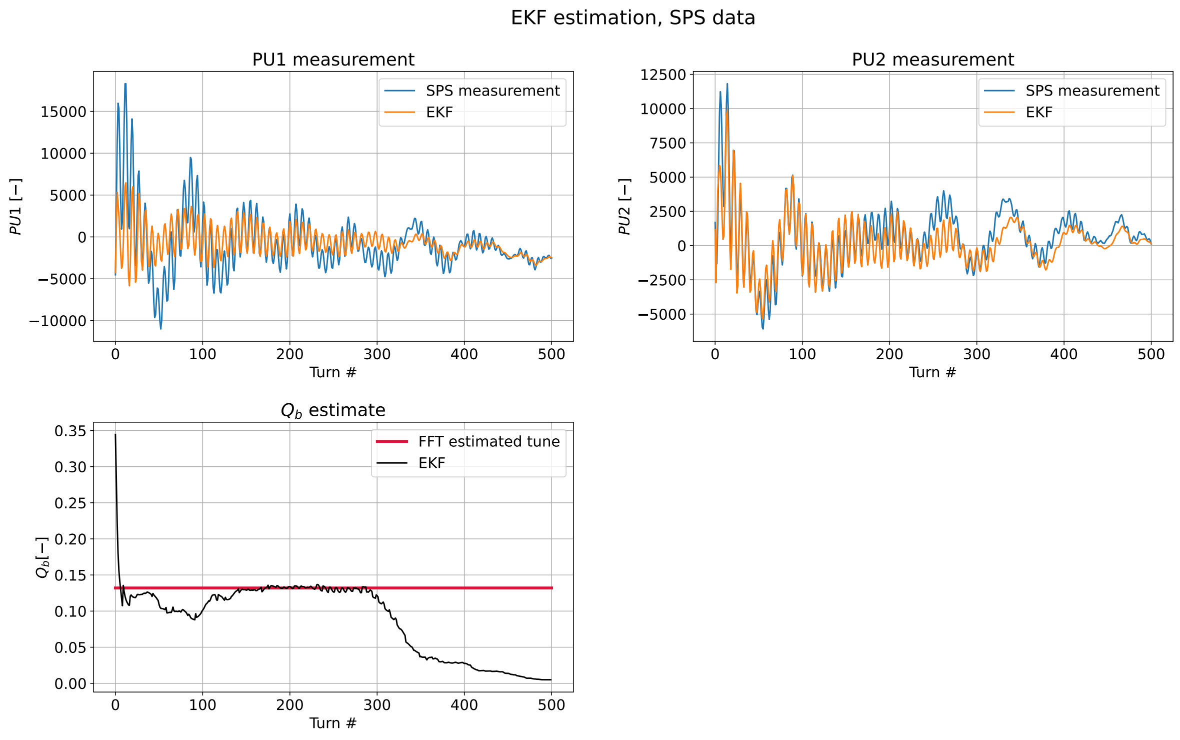


Figure 8: EKF applied to SPS one-bunch data.

4 Supervised Learning Methods for Tune Estimation

After the implementation of model-based tune estimation algorithms in section 3, machine learning model are used to estimate the tune, this time by analysing the frequency data of the BBQ spectrum. This section first describes the data processing required to make the raw data from the accelerators suitable for the machine learning models (subsection 4.1). Secondly, the different model architectures are described in subsection 4.2. Finally, the results obtained from implementing the models are shown in subsection 4.4.

4.1 Data Processing

In order to implement machine learning (ML) models that are supervised, the learning data must come with a label for the ground truth. Since the BBQ tune estimation is prone to identify other peaks in the spectrum as the tune, as shown in Figure 9, where two peaks can be found in the BBQ spectrum (right image), human labels are chosen instead. To achieve this, a data labelling tool was developed, with an example of the interface shown in Figure 10. This tool allows for labelling tune spectra with human accuracy. Here, an `argmax` tune function is interpolated over the spectrum according to some "tune checkpoints" (marked with red crosses) labelled by a human. In this manner, large amounts of data can be labelled with minimum human input. The data was collected directly from PS, PSB, and SPS accelerators, both from operational beams and machine development beams. Furthermore, the data is normalised, since the activation functions of neural networks work in normalised space [8]. The data is constructed by measuring during **512 turns**, and using a rectangular window for the FFT. The acquisition interval was set to 10 ms.

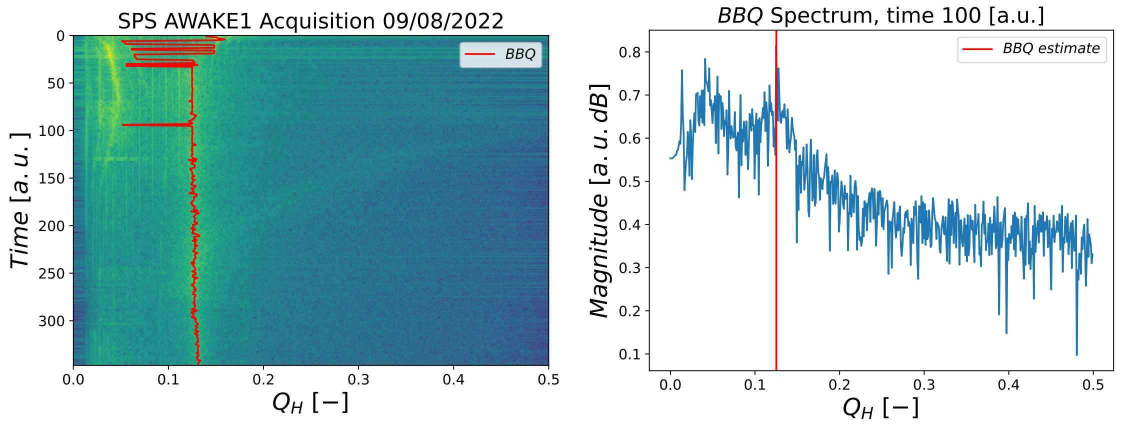


Figure 9: Left: BBQ spectra acquisition example, with the tune prediction shown. Right: Spectrum "slice" obtained from BBQ spectra shown in left image, with tune prediction shown.

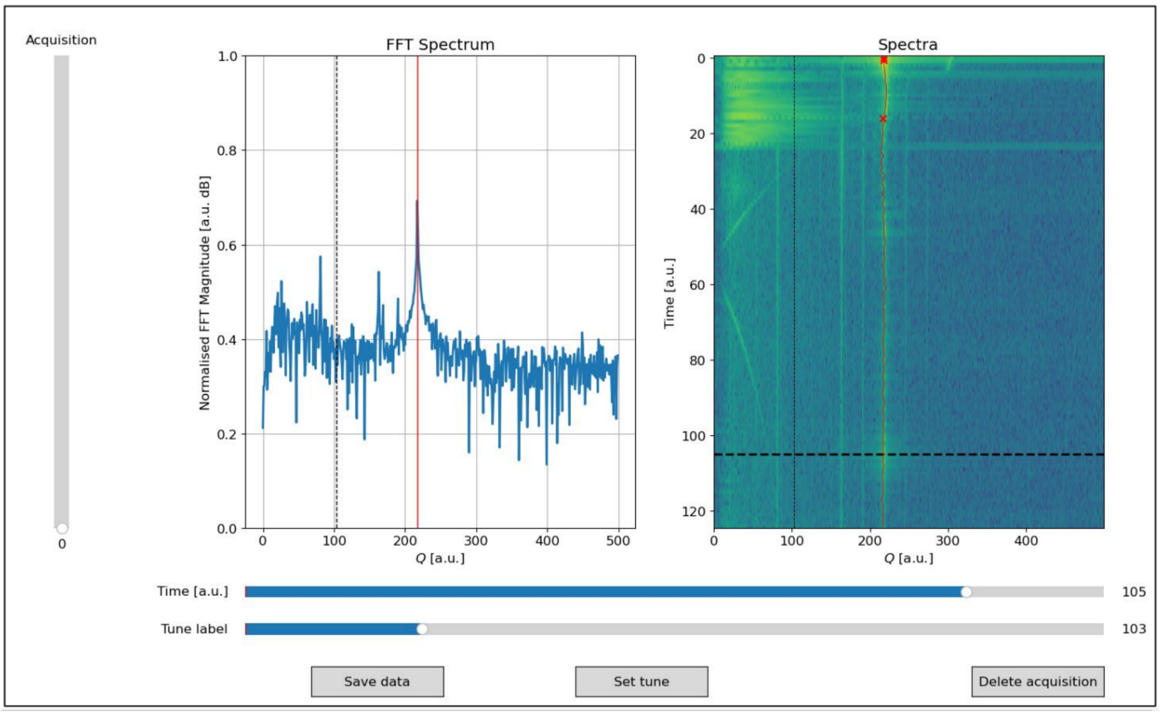


Figure 10: Data labelling tool.

4.2 Machine Learning Models

Several ML models have been used to estimate the tune and assess an optimal ML architecture; Feed forward (FF) models, convolutional (CNN) and recurrent (RNN) neural networks, and a combined convolutional-recurrent neural network structure CNN-RNN [4]. These models/architectures have different characteristics that can help improve the tune estimation accuracy. All models are implemented using TensorFlow in Python.

Feed forward neural networks (see Figure 11) are used to map an input to an output. This mapping is achieved by using three main elements; weights w and biases b (the trainable parameters of the network), and the so-called activation functions (which need to be specified beforehand, and can be non-linear), shown as dots in Figure 11. Furthermore, a FF network can consist of several layers; each with its own sets of weights, biases, and activation functions. Each layer maps an input vector x into an output vector y by performing a dot product between the weights tensor W and the input x, and then adding the bias term: $y = W \cdot x + b$. This in itself induces a linear transformation of the input. In this context, the activation functions are used as operators that, if desired, can add non-linearities, or modify the mapping $x \rightarrow y$ in a certain way. Thus, for example, if the `ReLu` activation function is used, the mapping is as follows $y = \texttt{ReLu}(W \cdot x + b)$.
Different layers can be stacked (the output of one layer becomes the input of the next) in the network to achieve increasingly complex mappings. Furthermore, the "learning" procedure consists on finding the correct weights and biases to realise a certain mapping [3]. This is achieved by using the gradient of the loss (error), whose metric needs to be defined (mean squared error, for example). This loss gradient is then related to each of the trainable parameters (weights and biases) by using the derivatives chain rule [3]. FF networks are used in this project as the first attempt to correctly map the BBQ spectrum to the tune, and also serve as a benchmark for the other network architectures.

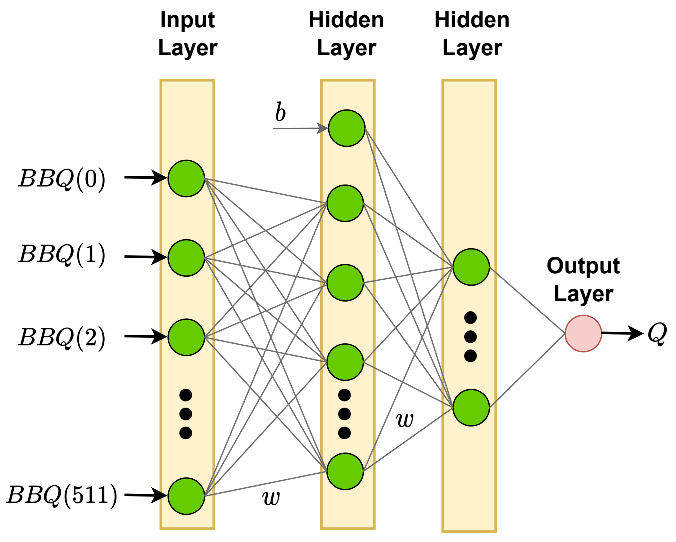


Figure 11: Feed forward network architecture.

While the FF network learns global patterns in the input spectrum, the CNN learns **local** patterns [3]. These learned patterns are translation invariant [3]. Furthermore, the spatial hierarchies of patterns are also learned [3]. This allow CNNs to understand spectrum features to identify the tune, possibly learning to differentiate spectrum traits to correctly assess the tune maxima. The different CNN elements are shown in Figure 12. First, a convolution operation is done over the data. Here, a sliding window (kernel) of predefined size stops at every possible location in the data and extracts a 2D patch (*window length*, *input depth*) of the surrounding features [3]. The number of features (the filters in Figure 12) is also predefined. After this, a max-pooling operation is used to *"to aggressively down sample feature maps, much like strided convolutions. Max pooling consists of extracting windows from the input feature maps and outputting the* `max` *value of each channel. It's conceptually similar to convolution, except that instead of transforming local patches via a learned linear transformation (the convolution kernel), they're transformed via a hard-coded* `max` *tensor operation"* [3, p. 209]. Convolution and max-pooling layers can be stacked to extract higher level features from the data. Finally, the data from the last max-pooling layer is flattened into a a 1D array which can then be connected to dense layers (much like the FF model layers) to finally output the tune.

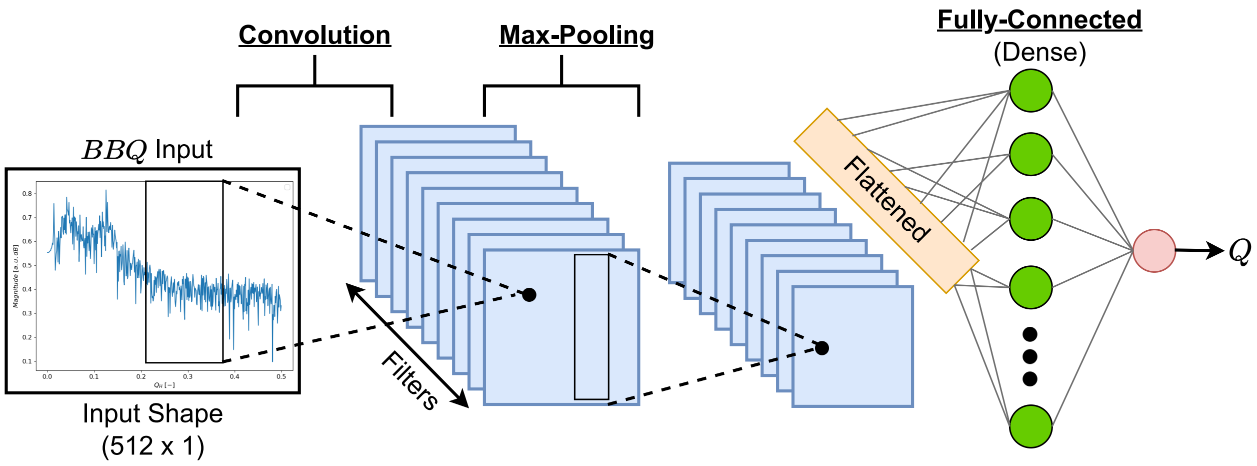


Figure 12: Convolutional neural network architecture.

When looking at Figure 9, it can be observed that knowing past tune values can help better predict the current tune. In this context, the RNN model is implemented since it can estimate present tunes values also considering the time-series evolution of the tune. Particularly, the implemented RNN model has a many-to-one (many inputs, one output) configuration, see Figure 13. Moreover, classic RNNs have the problem of catastrophically forgetting information from the past, as sometimes the gradient used to update the network (i.e. learning) vanishes or increases to a very large number, over time. To overcome this issue, Long Short Term Memory cells (LSTM) are used [11]. LSTM cells work by recursively transmitting two channels of information over the input; the cell state C and the the self state H, shown in Figure 13. The cell state C can be seen as the long term memory component of the cell, while H, the self state, is the short term memory component. The cell state C transmits information down the entire time-series input chain, having only some minor linear interactions, and short-term memory modifications are possible, but controlled by information gates who forget, store, and update information. Furthermore, the H cells carry the more recent information, and directly concatenate with the input at each time step.

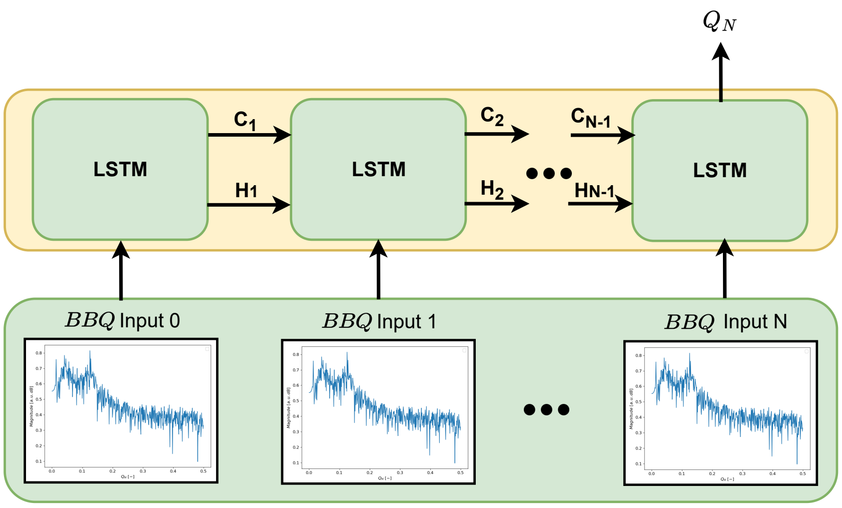


Figure 13: Recurrent neural network architecture (N tune spectra are inputted, and the current tune prediction is the output).

Finally, it is interesting to combine the powerful spatial feature extraction capabilities of CNNs with the temporal feature extraction of RNNs (see Figure 14). Thus CNN-RNN networks are also considered to estimate the tune. Such networks are currently used in science, giving positive results in fields such as glaucoma detection [7]. For the tune estimation problem, the network architecture consists of a CNN layer that extracts the features from the BBQ spectrum which are then connected to a LSTM layer. Furthermore, a many-to-many configuration, meaning that the network receives many temporal inputs and also has several temporal outputs (predicting the tune at each time step), is used.

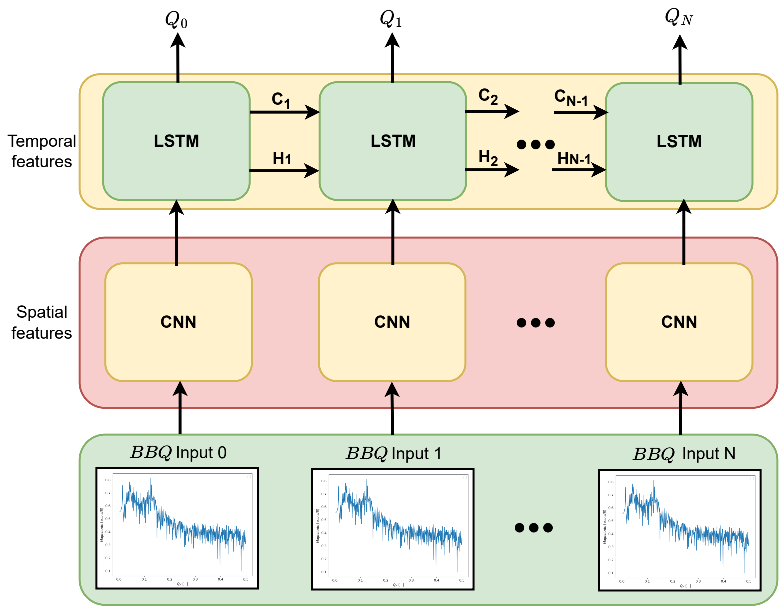


Figure 14: CNN-RNN neural network architecture.

4.3 Model Selection

When implementing the different ML models, several modifications where tried to optimise the network; stacking layers, modifying the layers sizes, activation functions, loss, and optimiser. Importance was given to reduce the amount of parameters of each model in order to decrease the computation complexity. Furthermore, in the final configuration, the `Adam` optimiser was used with a mean squared error loss. For all models, the batch size was set to 10 *BBQ* spectra, and 50 epochs were used. All the other learning parameters are the TensorFlow 2.3.0 defaults. The final architectures are shown in Tables 1, 2, 3, and 4.

Table 1: Feed Forward neural network architecture.

Layer (type)	**Output Shape**	**Parameters**	**Activation**
Dense	(None, 400)	205200	ReLu
Dense	(None, 300)	120300	Linear
Dense	(None, 70)	21070	Linear
Dense	(None, 1)	71	Linear
Total parameters: 346,641			

Table 2: CNN network architecture.

Layer (type)	**Output Shape**	**Parameters**	**Activation**	**Kernel Size**	**Filters**
Conv1D	(None, 493, 32)	672	Linear	20	32
MaxPooling1D	(None, 246, 32)	0	-	-	-
Flatten	(None, 7872)	0	-	-	-
Dense	(None, 1)	7873	Linear	-	-
Total parameters: 8,545					

Table 3: RNN network architecture.

Layer (type)	**Output Shape**	**Parameters**	**Activation**	**Rec. Activation**
LSTM	(None, 1, 32)	69760	Tanh	Sigmoid
Time Distributed Dense	(None, 1, 1)	33	Linear	-
Total parameters: 69,793				

Table 4: CNN-RNN network architecture.

Layer (type)	**Output Shape**	**Parameters**	**Activation**	**Kernel Size**	**Filters**
Time Distributed Conv1D	(None, 10, 512, 10)	110	Linear	10	10
Time Distributed MaxPooling1D	(None, 10, 256, 10)	0	-	-	-
Time Distributed Flatten	(None, 10, 2560)	0	-	-	-
LSTM	(None, 10, 32)	331904	Tanh	-	-
Time Distributed Dense	(None, 10,1)	33	Linear	-	-
Total parameters: 332,047					

Once trained on several operational beams, it was found that the CNN-RNN model showed better performance when compared to the other models, as well as showing more robustness to over-fitting, which was found to be a weakness of the other models, which showed worst generalisations capabilities. Thus, in the end the CNN-RNN model was chosen. Figure 15 shows the different tune lines found by the model for an SPS SFTPRO1 beam. Although all models show good performance (low variance in this case, for a constant tune line), all models except the CNN-RNN struggled to find the tune for spectra with heavily changing or severe noise, as is sometimes the case in the PSB and PS beams.

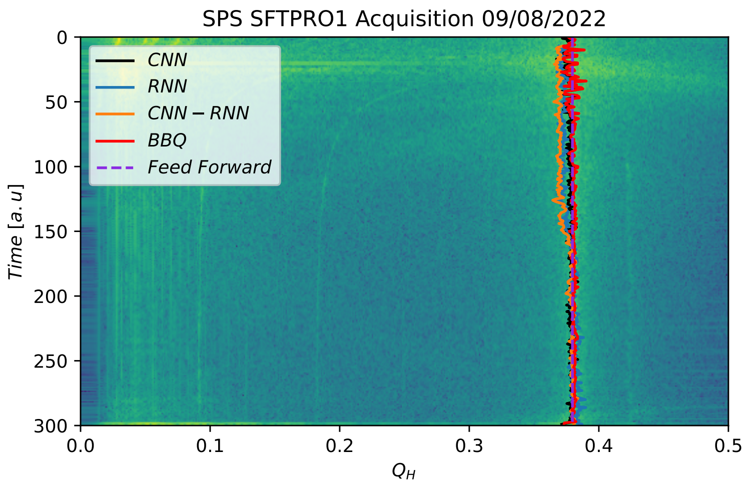


Figure 15: *BBQ* spectra with different machine learning model tune estimates.

4.4 Results

Once trained with human-labelled data, the CNN-RNN model was validated with unseen *BBQ* spectra from the PSB, PS and SPS beams. This is shown in Figure 16, where it can be observed (top right plot) that, except for some outliers, the CNN-RNN model closely matches the human labels. It is also worth noting that, since the output of the CNN-RNN is continuous, a discrete peak finder which looked 5 FFT bins away from the model prediction was used to correctly identify the discrete FFT bin corresponding with the tune.

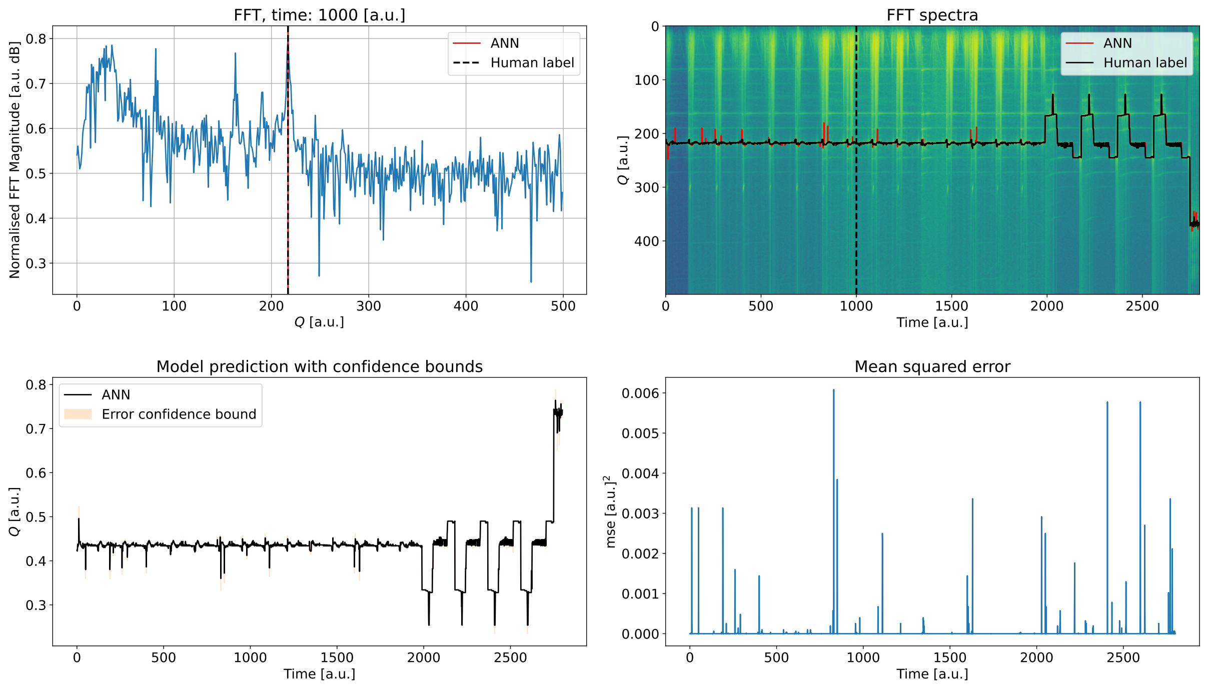


Figure 16: Validation data with the CNN-RNN model tune estimates. Top left: example of model output for a single *BBQ* spectrum, top right: comparison between model output and human label over several *BBQ* spectra, bottom left: model prediction with error confidence bound, bottom right: MSE for the different tunes shown in the top right figure.

4.5 Miscellaneous

As a side-study, a LSTM model was used to predict the next pick up reading in the accelerator, using the previous three readings. Such a model is shown in Figure 17, where it is observed, that, for the small dataset shown, the model correctly predicts the next measuring of the pick up signal, as well as decreasing the learning and validation losses over training epochs.

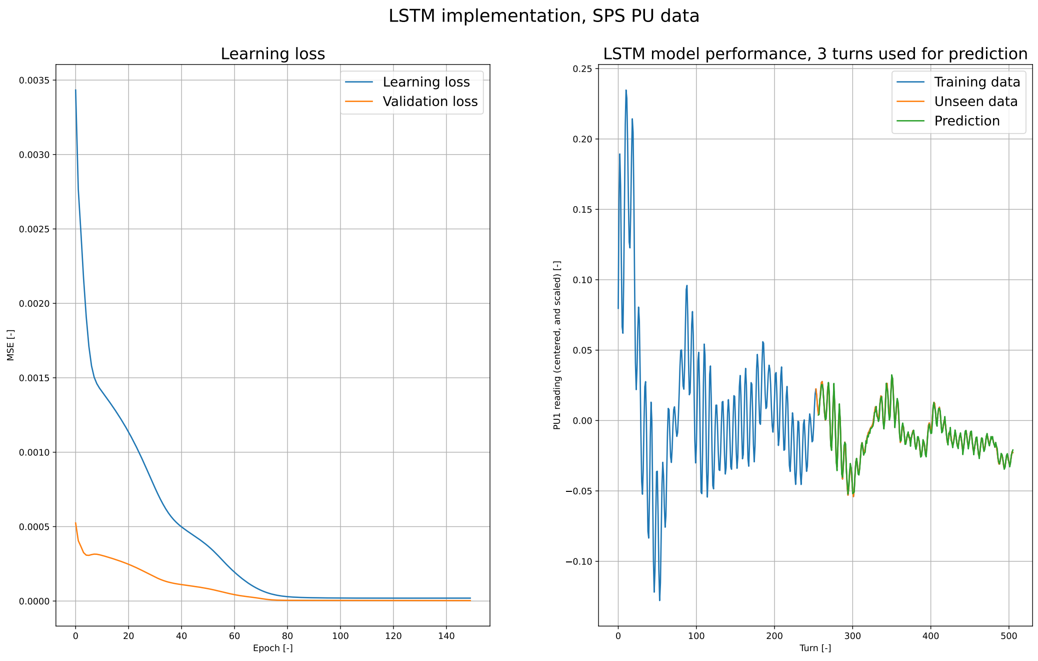


Figure 17: LSTM model used to predict the next pick up reading.

5 Fourier Synchrosqueezed Transform

Since the tune spectra generated by the classical FFT algorithms explored so far contain artefacts that can mislead the tune-finding algorithms, a new algorithm was explored to produce tune spectra with sharper, more protruding tune lines. The algorithm is described in subsection 5.1, and its performance is described in subsection 5.2.

5.1 Algorithm description

To analyse the time-frequency evolution (TF) of the tune, the synchrosqueezing transform was used (ST) [15]. The ST belongs to the class of time-frequency reassignment (TFR) algorithms, and allows for the extraction of oscillatory signal components that change in amplitude and frequency over time [15]. This algorithm post-processes the Pick Up (PU) signal's spectrogram and *"focuses the spectrograms's energy towards the instantaneous frequencies (IF) curves and results in a sharpened TF plot"*[15].

Particularly, **the synchrosqueezed short-time Fourier transform (FSST) [6] was used to extract the tune from the PU signal**. The PU signal $f(t)$ was assumed to have the form of equation 4, where each component $f_k(t)$ is an oscillating function with amplitude $A_k(t)$ and phase $\phi_k(t)$. The IF is then $d\phi_k(t)/dt$. Obtaining the IF of the tune over turns is the objective.

$$f(t) = \sum_{k=1}^{K} f_k(t) = \sum_{k=1}^{K} A_k(t) e^{2\pi i \phi_k(t)} \tag{4}$$

To estimate the tune, first, a modified short-time Fourier transform (STFT) of the PU signal $f(t)$ can be computed (referred to as V_g), using a spectral window g and adding a modulation factor with parameter η, as described in [6]. The IF information is then defined as equation 5 [6, p. 2].

$$\omega f(t,\eta) = \frac{\partial_t V_g f(t,\eta)}{2\pi i V_g f(t,\eta)} \tag{5}$$

Furthermore, FSST is benchmarked against the STFT, and the Hilbert transform IF (see Equation 6), where P^+ is the single-sideband modulation or the analytic signal. This definition is further described in [6, p. 3].

$$\mathrm{IF}_H f(t) = \frac{1}{2\pi} \operatorname{Im}\left(\frac{\partial_t P^+ f(t)}{P^+ f(t)}\right) \tag{6}$$

5.2 Results

To analyse the FSST performance, a chirped PSB signal was measured, and the instantaneous frequency (tune) of the signal was computed using the Hilbert, STFT, and FSST algorithms, as shown in Figure 18. Here, it is observed that both the STFT and FSST compute more robust tune lines when compared against the Hilbert transform, which is susceptible to noise, specially when the signal amplitude decreases. Furthermore, it can be observed that the FSST tune line is sharper than the STFT one, with the tune line squeezed into fewer frequency bins. Also, the FSST heavily reduces the noise in the spectrogram, which is still observable in the SFTF spectrogram. This results in a spectrogram where extracting the tune line is facilitated by the fact that there is less noise information. Furthermore, depending on the signal characteristics, it was found that using windowing techniques can help further sharpening the tune line.

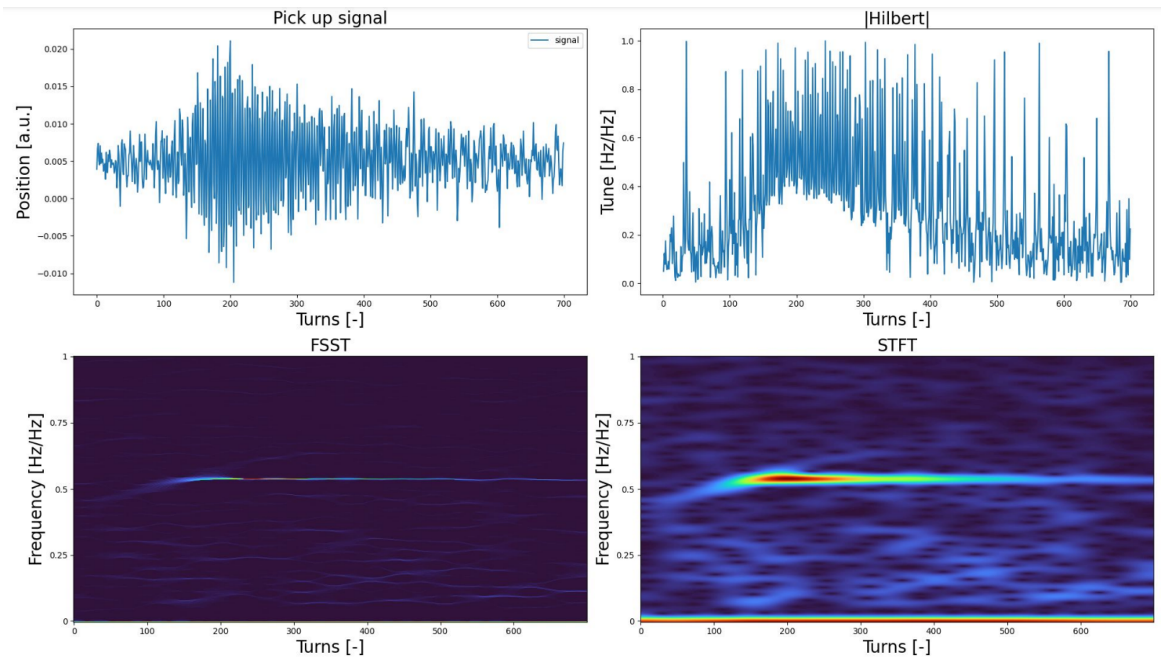


Figure 18: Top left: measured pick up signal, top right: IF Hilbert transform of the signal, bottom left: FSST spectrogram, bottom right: STFT spectrogram.

6 Reinforcement Learning for Transverse Damping

Regarding the control of the transverse dynamics in the accelerator, currently, a proportional control strategy is used. With the emergence of machine learning, the study of other control algorithms is of interest, since they have the potential to allow for a faster damping of the bunches' transverse dynamics, or reducing the power/energy needed for damping. More specifically, reinforcement learning is investigated. Subsection 6.1 describes the reinforcement learning implementation, and subsection 6.2 the obtained results.

6.1 The reinforcement learning scenario

The reinforcement learning RL scheme (shown in Figure 19) is as follows; an agent (which can be imagined as the control law) interacts with an environment by taking actions. Also, the agent observes the environment, and gets a reward based on the state of the environment and the action it took. In this context, the environment is the particle accelerator, the observations are made via the PUs, and the action is the kicker input on the bunch. **The goal of the agent is to damp the transverse motion of the particles**. The agent does this by maximising the total cumulative reward after an episode is finished (in this case, after the particle bunch completes a set amount of turns in the accelerator). In this manner, via trial and error (updating the agent's parameters) over multiple episodes, the agent **learns** to control the accelerator. The implementation of the three main elements of RL (**environment, agent** and **reward function**) is described in subsections 6.1.1, 6.1.2, and 6.1.3.

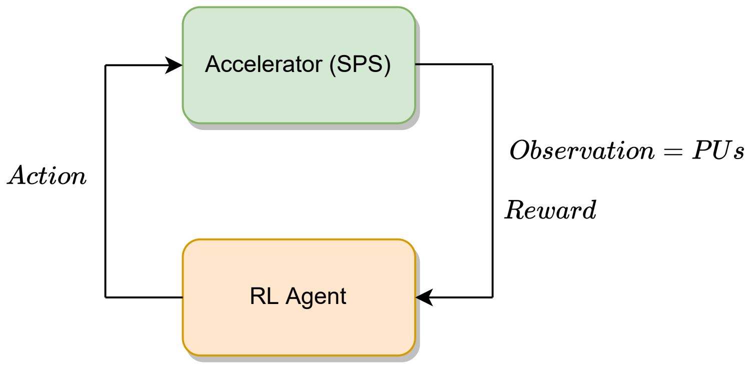


Figure 19: Reinforcement learning model.

6.1.1 The environment

To correctly deploy the RL agent, it is important to have a defined environment. In this study, the environment is a particle accelerator, which is simulated for the purpose of this study. Specifically, the transverse beam dynamics are simulated, accounting for tune effects, octupole magnets, the transverse contribution of the longitudinal dynamics, and the effect of the feedback system, where the agent takes actions. Furthermore, for each RL episode, the initial states of the accelerator are generated from a uniform distribution bounded to the desired phase space limits. It is worth mentioning that the environment has four states embedded in two complex numbers; the horizontal and vertical phase space components of the transverse and longitudinal directions, B and B_L. Only the transverse states are visible for the agent (via the PUs in the real accelerator). The simulated particle accelerator environment scheme is shown in Algorithm 2.

Algorithm 2 Accelerator Environment (tailored for **Scenario 1**)

```
1: if Initialising = True then
2:     $\Re(B), \Re(B_L) = U(-10,10)$                          ▷ Initialising real states
3:     $\Im(B), \Im(B_L) = U(-10j,10j)$                        ▷ Initialising imaginary states
4: else
5:     B $\leftarrow B \cdot \exp(2\pi j Q)$                    ▷ Phase advance due to tune
6:     B $\leftarrow B \cdot \exp(2\pi j \mu |B|^2)$           ▷ Phase advance due octupole magnets
7:     B$_L \leftarrow B_L \cdot \exp(2\pi j Q_L)$              ▷ Longitudinal phase advance due to synchrotron tune
8:     $\delta = \Re(B_L)$                                      ▷ Energy offset
9:     B $\leftarrow B \cdot \exp(2\pi j \xi \delta)$          ▷ Chromaticity
10:    B $\leftarrow B - G \cdot action \cdot j \cdot \overline{\Im(B)}$   ▷ Feedback contribution
```

In this study, two RL scenarios are treated; **Scenario 1**, where the agent scales a maximum allowed feedback gain on the kicker (the control input to the particle beam), and **Scenario 2**, where the agent directly kicks the beam with a maximum allowed strength. Furthermore, **Scenario 1** uses the complete simulated model (Algorithm 2), while **Scenario 2** only takes into account the tune effects, but also initialises each episode with a new, randomly generated tune. These two scenarios are used to investigate RL agents that could be deployed in the current accelerator setup (for SPS) in the case of **Scenario 1**, while also giving a less restricted control problem (**Scenario 2**) to the agent to observe whether new control policies emerge.

6.1.2 The agent

Several RL agents have been considered: A2C, DDPG, DQN, PPO, TD3, and SAC (these algorithms are implemented with Stable Baselines3 [13]).
It is important to realise that the RL problem deals with both continuous action and observation spaces. Hence DQN, although showing good performance in several control tasks [16], is not suited for continuous action spaces, and is thus discarded. When inspecting the other algorithms, SAC is chosen because of its robustness regarding convergence during learning [10], as well as in the face of model and estimation errors [18]. SAC also outperforms the other algorithms in several control tasks, showing more stability and homogeneous performance when different random seeds are used during training [10]. SAC is a maximum entropy RL algorithm, which aims at succeeding at a certain control task while maximising its entropy [10]. One important aspect of SAC is its sensitiveness to the scaling of the reward function, which is regarded as the main hyper parameter that requires tuning [10].

6.1.3 The reward function

The objective of the implemented RL agent is to damp the transverse bunch oscillations as fast as possible. Thus, the reward function is shaped to be inversely proportional to the norm of the PU readings. Initially, sparse reward functions have been considered; providing positive rewards only after a very small norm in the phase space is achieved. Such reward functions allow the agent to develop creative control strategies, since they don't contain previous knowledge on the control task, only the desired final state. However, when implemented, this sparse function led to erratic learning, providing poor damping times and even making the phase space norm increase over time. This is because it takes a great amount of turns (assuming good damping control) for the phase space norm to be small enough such that a positive reward is obtained, thus, it is challenging for the agent to converge to a specific control strategy.

To overcome this issue, a denser reward function is used which increases in reward (or decreases in penalty) as the norm of B, the transverse states, is reduced.

6.2 Results

As discussed in subsubsection 6.1.1, each scenario has a specific setup for the environment. The environments' settings are shown in Table 5. Here, the first SAC agent (**Scenario 1**) explores an environment where all parameters are constant with the exception of the initial states of the beam, while the second agent explores a simpler environment (see subsubsection 6.1.1), but also having a randomly varying tune at each episode. Also, the first agent can take actions between 0 and 1, while the second agent takes actions between -1 and 1. Both the observation and action spaces are continuous.

Table 5: Environment parameters.

	Initial States	Q	Q_L	μ	ξ	**Gain**	**Max Turns**
Scenario 1	*U(-10,10)*	0.18	5.5e-3	2e-4	2	0.01	2000
Scenario 2	*U(-1,1)*	*U(0,0.5)*	-	-	-	-	200

When deployed, both scenarios achieve transverse damping, although in different ways, as shown in Figure 20 and 21. In the case of **Scenario 1**, the agent first applies an action close to 1 (100% of the allowed damping strength) until the norm decreases to a small value in 730 turns. Then, the action strength is decreased, and the variance in the selected action increases (see top-right image of Figure 20). This is expected, because for small norms, changing the scaling of the gain has very little impact, given that the control input of the kicker is very small due to the nature of proportional controllers. Hence, the SAC agent receives very small changes in reward for the different actions it takes when the norm is small, and thus it doesn't correctly learn the optimum policy. This consists in using 100% of the kick strength, an action of 1 in Figure 20, since the maximum allowed gain of 0.01 results in a very under-actuated control problem. Furthermore, it can be observed that the norm damping curve of the SAC agent follows closely the proportional controller curve. This shows that the agent learnt the optimal control strategy.

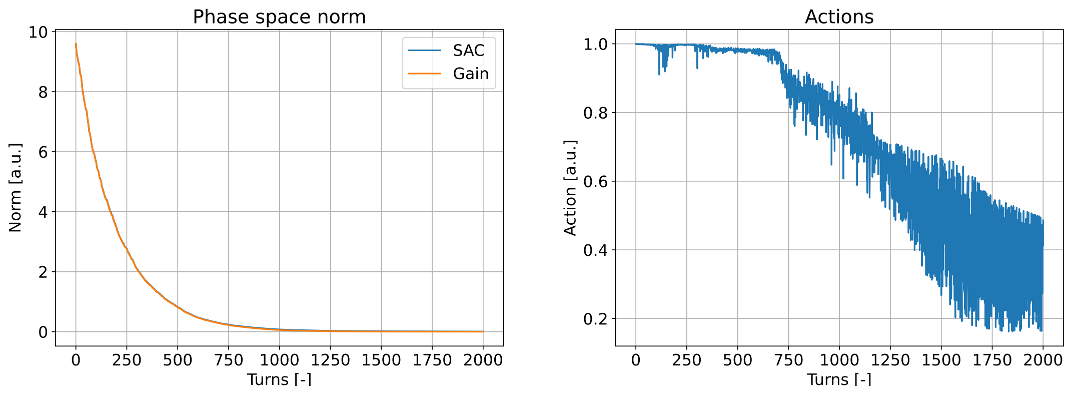


Figure 20: **Scenario 1** damping evolution for a given initial state (left), and the actions taken (right).

The damping curve in the phase space norm plot of **Scenario 2**, Figure 21, also shows that the SAC performance is similar to the proportional controller. This is not expected. Since the SAC agent was given a larger control authority, there is the possibility of outperforming the proportional controller. This way, a bang-bang control strategy was expected; placing the beam close to the phase-space vertical axis (see bottom plot of Figure 21), and then kicking with the required strength to damp the norm, reaching the point $(0, 0)$ in the phase space. Instead, the SAC agent damps the strength of its actions, much like a proportional controller, to damp the particles. This could occur because, since the tune was also randomised for each learning episode, the agent needed to find the bang-bang control approach for different starting positions in phase space, and different tunes. This results in a large search space, and it is probable that with larger training times (the agent learnt during several days), the bang-bang control strategy is found.

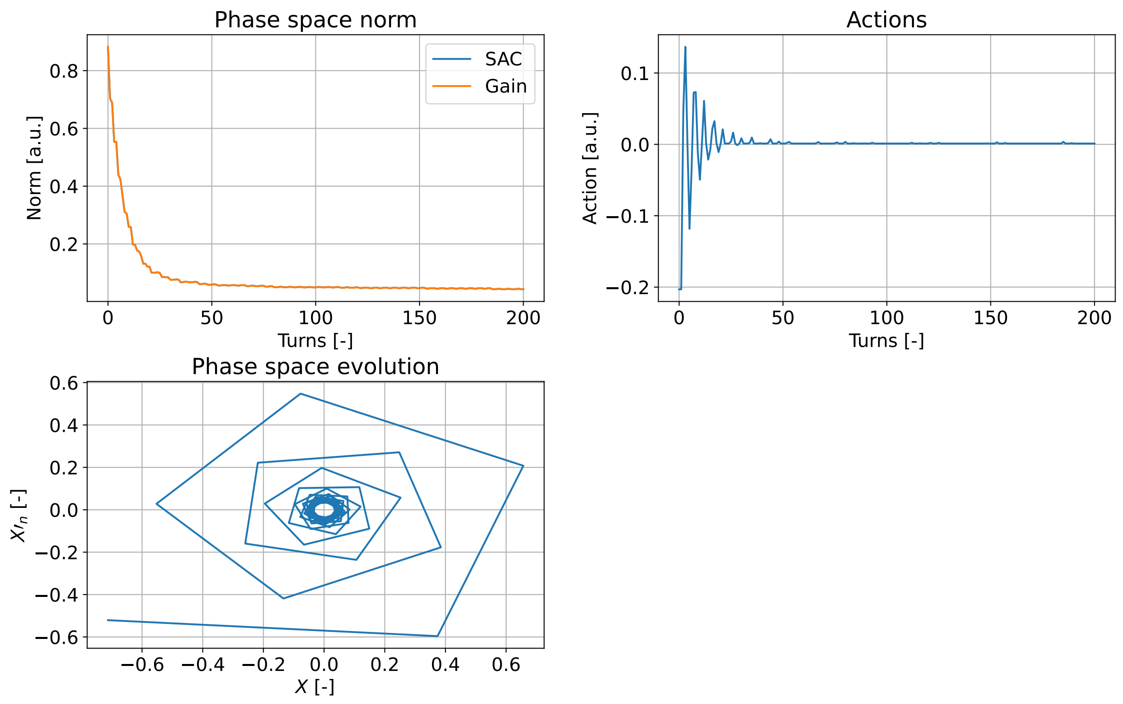
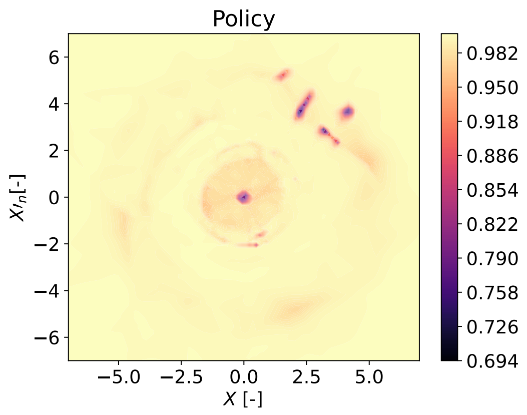
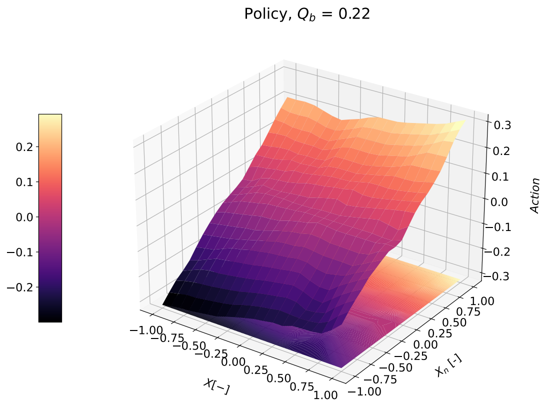


Figure 21: **Scenario 2** damping evolution for a given initial state (left), the actions taken (right), and the phase space evolution (bottom).

When inspecting the learnt control policies, Figure 22, it is observed that for **Scenario 1**, most of the phase space actions correspond to a 100% kick, as expected. Also, as previously discussed, the action strength reduces closer to the point $(0, 0)$. Furthermore, some weak action traces can be seen in the top right of the phase space. This traces lie very close to the tune line. This way, if the particles are in that region, they will land very close to the vertical axis in the plot in the next turn, hence, specially when the norm is large (as is the case in the region of the traces), the vertical kicking in the particles becomes more effective. This might have been learnt by the SAC agent.
Moreover, the **Scenario 2** policy shows a clear pattern of using positive actions in the positive X_n' regions of the phase space, and negative actions in the negative counterparts. This is expected since a positive action decreases the norm in the positive X_n' region, and viceversa.



(a) **Scenario 1** policy.



(b) **Scenario 2** policy, $Q = 0.22$.

Figure 22: SAC Agents' policies.

7 Model Predictive Control for Transverse Damping (MPC)

As discussed in section 6, machine learning algorithms are explored as an alternative to the proportional controller currently used in the damper. In this section, model predictive control, combined with online system identification, is investigated. subsection 7.1 describes the fundamentals of model predictive control, subsection 7.2 gives a brief overview of the online system identification algorithm, subsection 7.3 explains the implemented controller, and subsection 4.4 discusses the simulation results and how they compare against a proportional controller.

7.1 Model predictive control

Model predictive control is implemented with the idea of leveraging the fact that a model of the transverse dynamics is available (described in subsubsection 6.1.1). In this fashion, the accelerator model is used to propagate in a predefined horizon of turns the transverse position of the bunch (from the initial conditions defined by the pick-up readings), and optimise a set of control action that would lead to mininmise a predefined cost function. The control scheme is established to optimise a control input sequence that will minimise transverse emittance, turns to damp the bunch below a threshold, and the aggressiveness of the changes in the control actions, while respecting power constraints. The schematic of the control loop is shown in Figure 23, where is observed that the controller optimise a set of actions with the pick-up reading as the initial conditions, and then, the first action of the found sequence of optimum actions is applied on the accelerator.

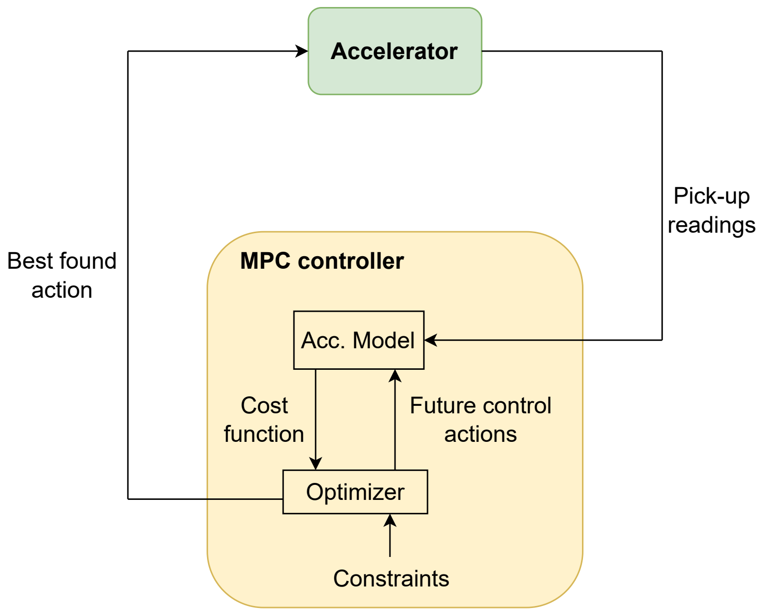


Figure 23: Model predictive control scheme.

7.2 Sparse identification of nonlinear dynamics (SINDy)

The model predictive control approach described in subsection 7.1 is limited by the accuracy of the model, which can be affected by noise, or changes of non-modelled parameters in the accelerator. To allow for a controller that is also flexible to changes in the dynamics of the accelerator, we use sparse identification of nonlinear dynamics (SINDy) [2], a machine learning approach to obtain the governing equations of a dynamical system from measured data. Here, the SINDy algorithm extracts from a predefined library of candidate functions that can describe the dynamics, the fewest combination of possible terms that will describe the measured data. In this way, a data-driven, flexible approach that results in a deterministic set of equations can be used to find the accelerator dynamics.

7.3 SINDy + MPC

By replacing the accelerator model of the MPC controller shown in Figure 23 by the set of equations discovered by the SINDy algorithm; an explainable, adaptive and powerful control algorithm can be designed [12]. This is schematised in Figure 24, where the SINDy model is constructed by a memory buffer containing past states and control actions up to a predefined horizon.

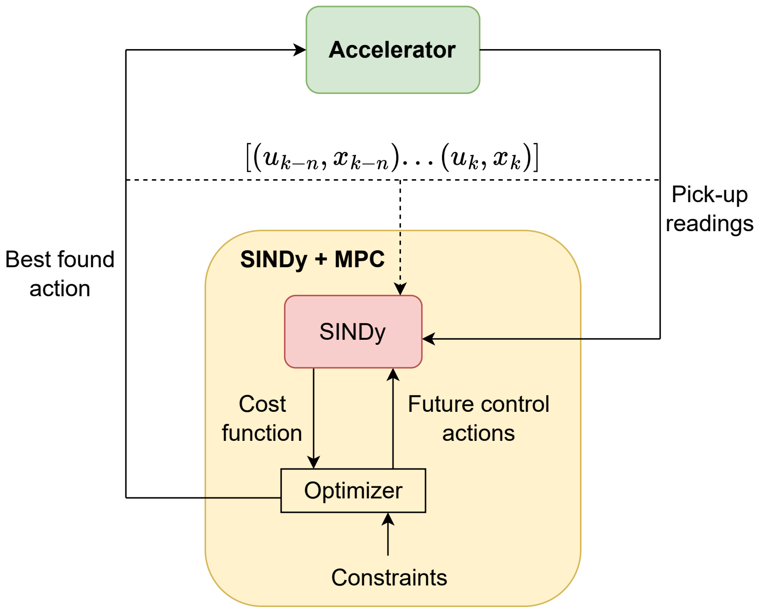


Figure 24: SINDY + MPC scheme.

7.4 Simulation results

The SINDy+MPC controller was bench-marked against a proportional controller in the task of damping the transverse dynamics of a simulated bunch. As shown in Figure 25, after a certain amount of turns (19 in this example), the SINDy+MPC controller achieves a better damping of the bunch, and reduces the bunch tail. The cause of this reduction in bunch tail needs to be further investigated, and is yet to be tested with a real beam (in this simulation, control actions could only be taken on the X'_n plane).

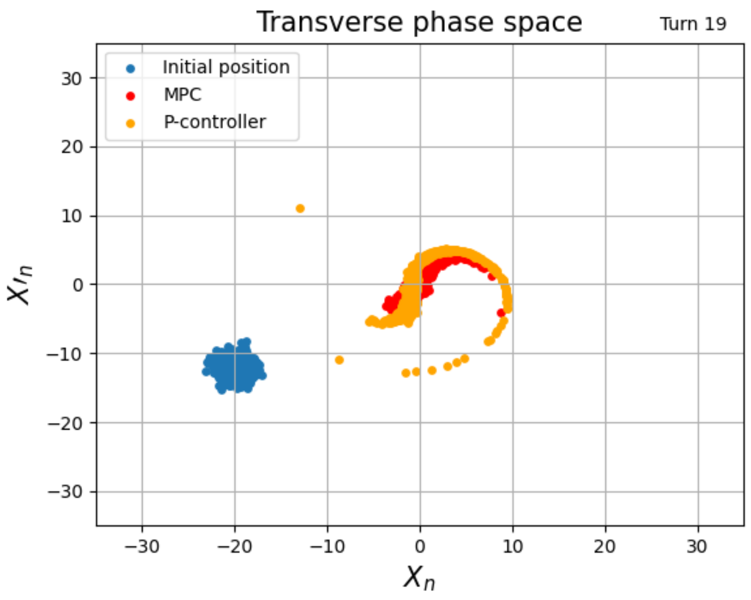


Figure 25: Transverse phase space after 19 turns.

Moreover, the SINDy algorithm was proven to be robust to noise, identifying the correct dynamics with noise levels tested up to $\mu = 0$, $\sigma = 0.35$. Figure 26 shows how the propagated dynamics identified by the SINDy model accurately match the true simulated dynamics (which where injected with noise during the system identification phase) up to a range of 20-25 turns in this scenario, and thus making it suitable for the combination with MPC.

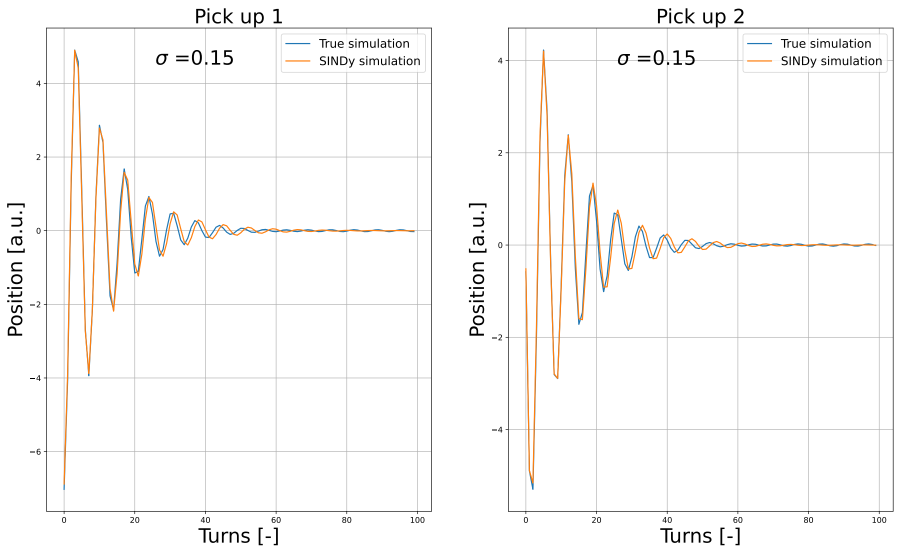


Figure 26: SINDy system identification with noisy measured dynamics ($\sigma = 0.15$), with the identified model propagated and compared with the true dynamics.

8 Conclusion

In this study, four main objectives have been established: implementing the Extended Kalman Filter and estimating the tune in time series data, implementing machine learning tune estimation algorithms and correctly identifying the tune in spectral data, investigating alternative methods for processing the BBQ spectrogram, and successfully damping beam oscillations with reinforcement learning and model predictive control.

Regarding the first objective, the Extended Kalman Filter was successfully implemented in simulated data, correctly estimating the tune, and tune changes. However, the algorithm's performance in real data was deteriorated due to the lack of accuracy in representing the real accelerator dynamics. Furthermore, the EKF method for tune estimation in time series data is fundamentally limited in its accuracy when the beam is well damped, given the error source caused from converting the amplitude domain data to the phase domain to extract the tune.

For the second objective, several machine learning algorithms were successfully implemented to identify the tune from real frequency data; feed forward neural network models, CNN, RNN, and CNN-RNN. However, the CNN-RNN model was more robust to tune changes and less prone to over-fitting, showing similar results to those of human labels. Moreover, the FSST algorithm successfully post-processed the BBQ spectrogram to sharpen the tune line and reduce the noise.

Finally, two reinforcement learning agents have been implemented to damp the beam in a higher fidelity accelerator environment, and an accelerator model only including the tune dynamics. In both cases, the agents successfully damp the beam, but fail to outperform the proportional controller strategy. Furthermore, the first agent exhibits a very homogeneous policy, almost always choosing large actions, while the second agent has a more varied policy, changing the magnitude and sign of the taken actions. As stated, both policies successfully damp the beam oscillations.
Regarding the model predictive controller with online system identification, the controller exhibits better damping performance when compared with the proportional controller, achieving a smaller bunch tail, and the SINDy algorithm can identify the system dynamics, even in the presence of noise.

9 Recommendations for Future Work

The objectives of this internship were to explore the potential of advanced algorithms such as machine learning in determining the transverse tune from recorded data, and investigating reinforcement learning techniques applied on transverse feedback systems, at the CERN particle accelerators. Both goals were achieved. However, to further continue with the examination of the different topics, several recommendations can be made.

Regarding the tune estimation, first, a model based tune estimation was implemented by using an Extended Kalman Filter EKF with simplified transition dynamics, which estimated the phase-space transition of the particle beam dynamics from amplitude data. This method obtained good performance with simulated dynamics, but limited accuracy with real data.
Also, machine learning methods to measure the tune were implemented using spectral BBQ data from the PSB, PS, and SPS accelerators. All methods successfully predict the tune but also tend to over-fit the training data (failing to extract the tune from unseen data, but having low prediction errors in the training dataset). However, it was found that the CNN-LSTM model is more robust to over-fitting than the other models. With this in mind, several modifications could be applied to both methods.

Firstly, when looking at the EKF problem, adding the full optics model of the accelerators, instead of the simplified model used in this report, might increase the validation accuracy, given that the EKF will have more accurate transition dynamics. Furthermore, the error caused by the AM to PM conversion will still remain in this set up, although new Kalman Filter approaches can be implemented examining the frequency spectra of the dynamics instead of the time series amplitude data, potentially unveiling new ways of circumventing this issue.

Secondly, in the machine learning tune estimation, the models should be trained and tested with more data, including that of LHC, to make them production-ready. Thus, next possible steps could include labelling data where the BBQ has poor performance, such as in injection and the RAMP LHC mode. Furthermore, training data including more varied tunes and tune changes over time is needed.

In the reinforcement learning investigation, two Soft Actor Critic reinforcement learning agents were implemented in two different scenarios; changing the gain of the accelerator damper, and directly kicking the beam. In both scenarios, the objective was to damp the particle oscillations as fast as possible.

Regarding the reinforcement learning control of the particle beam, it was observed that the reinforcement learning agents failed to outperform the proportional controller, only matching its performance. To overcome this issue, new control scenarios can be implemented; giving larger control authority to the agents (allowing for higher power inputs on the kicker). Also, new reward functions could be implemented to explore their effect on the agent's learning. Moreover, to further breach the reality gap with the real accelerator, a one-turn delay for the taken actions can be implemented.

Finally, the MPC+SINDy controller could further be implemented in a real accelerator and the causes of the differences between the transverse phase-space bunch profile achieved by the MPC+SINDy and the proportional controller, investigated.

References

[1] S. Baird. *Accelerators for pedestrians.* CERN-AB-Note-2007-014, CERN, Geneva, Switzerland, Feb. 2007. URL: https://cds.cern.ch/record/1017689.

[2] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[3] F. Chollet. *Deep Learning with Python.* Manning Publications, Nov. 2017. ISBN: 9781617294433.

[4] J. Donahue et al. "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 677–691. DOI: 10.1109/TPAMI.2016.2599174.

[5] F. Dubouchet et al. "Tune measurement from transverse feedback signals in LHC". In: (IBIC2013, Oxford (UK)). URL: https://cds.cern.ch/record/1711231.

[6] G.Thakur and H.-T. Wu. "Synchrosqueezing-Based Recovery of Instantaneous Frequency from Nonuniform Samples". In: *SIAM Journal on Mathematical Analysis* 43.5 (Jan. 2011), pp. 2078–2095. DOI: 10.1137/100798818.

[7] S. Gheisari et al. "A combined convolutional and recurrent neural network for enhanced glaucoma detection". In: *Sci Rep* 11, 1945 (2021). DOI: 10.1038/s41598-021-81554-4.

[8] L. Grech, G. Valentino, and D. Alves. "A Machine Learning Approach for the Tune Estimation in the LHC". In: *Information* 12 (Apr. 2021), p. 197. DOI: https://doi.org/10.3390/info12050197.

[9] L. Grech et al. "An Alternative Processing Algorithm for the Tune Measurement System in the LHC". In: (IBIC2020, Santos (Brazil)). DOI: 10.18429/JACoW-IBIC2020-WEPP27.

[10] T. Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290.

[11] S. Hochreiter and J. Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[12] E. Kaiser, J. N. Kutz, and S. L. Brunton. "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2219 (Nov. 2018), p. 20180335. DOI: 10.1098/rspa.2018.0335.

[13] A. Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[14] M. I. Ribeiro. "Kalman and extended kalman filters: Concept, derivation and properties". In: *Institute for Systems and Robotics* 43 (2004).

[15] G. Thakur et al. "The Synchrosqueezing algorithm for time-varying spectral analysis: Robustness properties and new paleoclimate applications". In: *Signal Processing* 93.5 (2013), pp. 1079–1094. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2012.11.029. URL: https://www.sciencedirect.com/science/article/pii/S0165168412004240.

[16] M. Volodymyr et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602.

[17] G. Welch and G. Bishop. "An Introduction to the Kalman Filter". In: Department of Computer Science University of North Carolina at Chapel Hill Chapel Hill, NC 27599-3175, 1994.

[18] B. D. Ziebart. "Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy". CMU-ML-10-110. Machine Learning Department, School of Computer Science, Carnegie Mellon University. PhD thesis. Pittsburgh, PA (USA), 2010. ISBN: 9781124414218.