Predicting Ski Jumps Using State-Space Model

Neca Camlek*
Univerza v Ljubljani
Ljubljana, Slovenia

Živa Hegler*
Univerza v Ljubljani
Ljubljana, Slovenia

Jakob Jelenčič
Jožef Stefan Institute
Ljubljana, Slovenia
jakob.jelencic@ijs.si

Marko Grobelnik
Jožef Stefan Institute
Ljubljana, Slovenia
marko.grobelnik@ijs.si

Dunja Mladenić
Jožef Stefan Institute
Ljubljana, Slovenia
dunja.mladenic@ijs.si

Abstract

Ski jumping performance is shaped by both athlete technique and environmental conditions, with factors such as wind speed, wind direction, and ski orientation playing a critical role in determining jump trajectories. Accurate modeling of these trajectories is challenging due to dynamic and time-dependent nature of the system. In this work, we introduce a dataset of measured ski jumps and present a state-space modeling framework that captures the evolution of jumps under varying conditions. The model parameters are estimated using a ridge regression approach, enabling us to predict trajectories from initial states and wind sensor inputs. We evaluated the predictive performance of the model through leave-one-out cross-validation and analyzed its stability, showing that the approach can generate realistic trajectories with reasonable accuracy. To complement the modeling results, we developed an interactive web application that allows users to explore both recorded and simulated jumps, adjust environmental factors, and visualize their effects through animations. Together, the dataset, modeling framework, and the application offer a foundation for further research in ski jump analysis and provide an accessible tool for exploring the influence of external conditions on performance.

Keywords

datasets, state-space model, ski jumping, simulations, least squares

1 Introduction

Ski jumping is a sport strongly influenced by both athletic technique and environmental conditions. Factors such as wind speed, wind direction, and different ski angles affect the trajectory and final distance of a jump, making accurate prediction a challenging problem. While statistical models and simulations have been applied in sports research for some time, many approaches simplify the problem and do not fully capture the dynamic evolution of the jump over time [11].

Recent advances in machine learning have introduced methods capable of modeling temporal systems with greater fidelity. In particular, state-space models provide a mathematical framework for representing hidden internal states that evolve over time in response to external input. This makes them well-suited for modeling ski jumps, where environmental factors determine performance [9].

In this paper, we present a ski jump dataset together with a state-space model trained to predict jump trajectories based on changing environmental conditions. The model is estimated using a least squares approach and demonstrates how inputs such as wind and ramp adjustments influence the resulting jump. Beyond the modeling framework, we also developed an application that allows general users to interact with the data, run simulations, and visualize jump trajectories through animations.

Beyond methodological interest, accurate prediction of ski jumps can improve athlete safety by anticipating risky conditions, support planning of hill design or enlargement, and contribute to fairer competitions through a better understanding of environmental effects.

The remainder of the paper is as follows. Section 2 presents the handling of received data. Next, the proposed methodology is described in Section 3. The project results are presented in Section 4. We discuss the results in Section 5 and conclude the paper in Section 6.

*Both authors contributed equally to this research.

Information Society 2025, Ljubljana, Slovenia
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2025.sikdd.30

2 Modeling Framework and Dataset

This section describes the handling of data, focusing on state-space models and our data processing.

2.1 State-Space Model

State-Space Models (SSMs) are a family of machine learning algorithms designed to capture and predict the behavior of dynamic systems by describing how their inner states change over time. Instead of only looking at past inputs and outputs, SSMs explicitly model the underlying dynamics, making them well-suited for sequential data. In state-space modeling, the objective is to identify the minimal set of system variables required to completely describe the system. These fundamental variables are referred to as the state variables. At any given time, the state of the system can be represented by a state vector, whose components correspond to the values of the respective state variables. SSMs are designed to predict both the manner in which inputs are reflected in the system's outputs and the evolution of a system's internal state over time and in response to specific inputs [2].

2.2 Least squares method

The least squares method is a regression technique that is used to determine the line that best fits a given set of data. It minimizes the sum of the squared differences between the observed data and the corresponding values implied by the regression function. Each data point reflects the relationship between a known independent variable and an unknown dependent variable [7].

To enhance the model, we incorporated ridge regression (L2 regularization), which helps to reduce overfitting during model training [12].

2.3 Data Processing

For our project, we used 223 CSV files, each containing the data of a jump, measured on the flying hill of Gorišek brothers in Planica, Slovenia. Each contains 17 columns (`'Position'`, `'Height above ground'`, `'Time'`, `'X'`, `'Y'`, `'Z'`, `'Opening Angle'`, `'Stalling Angle Left'`, `'Stalling Angle Right'`, `'Roll Angle Left'`, `'Roll Angle Right'`, `'Yaw Angle Left'`, `'Yaw Angle Right'`, `'Speed hor.'`, `'Speed vert.'`, `'Speed resulting'`, `'WindTime|WindName|WindSpeed|Wind...'`) and the number of rows corresponding to the length of the jump. Data are recorded for every meter of air distance from the take-off point.

The data required some pre-processing before it could be used for training the model.
The column `WindTime|WindName|WindSpeed|...` combined multiple attributes separated by '|'. Data from 12 sensors, each measuring six wind characteristics, were expanded into $12 \times 6 = 72$ columns, one per sensor–feature pair (`sensor_feature`).

Position - air distance from the take-off point in meters. Begins with a negative value, which represents the distance from the starting point to the take-off point. In ski jumping, the starting point is adjusted according to the wind conditions, so this value is not constant.

Height above ground - height above ground in meters.

Time - time of the jump in seconds from the start of the jump.

X, Y, Z - coordinates of the jumper in a 3D space in meters. The X axis is aligned with the hill direction, the Y axis is across the hill, and the Z axis is vertical. The take-off point is $(0, 0, 0)$ as shown in Figure 1

Opening Angle - angle between the skis in degrees.

Stalling Angle Left, Stalling Angle Right - angle between the chord line of the left/right ski and the horizontal plane in degrees.

Roll Angle Left, Roll Angle Right - angle of the left/right ski around its longitudinal axis relative to the horizontal plane in degrees.

Yaw Angle Left, Yaw Angle Right - angle between the left/right ski and the horizontal plane in degrees.
(angles are shown in Figure 2)

Speed hor., Speed vert., Speed resulting - horizontal, vertical, and the resulting speed of the athlete in km/h [13].

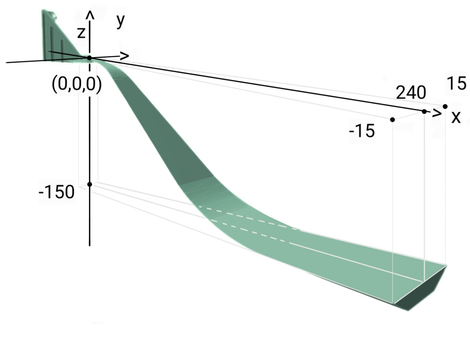

Figure 1: 3D model of Ski jump in Planica with added coordinates [10, 1]

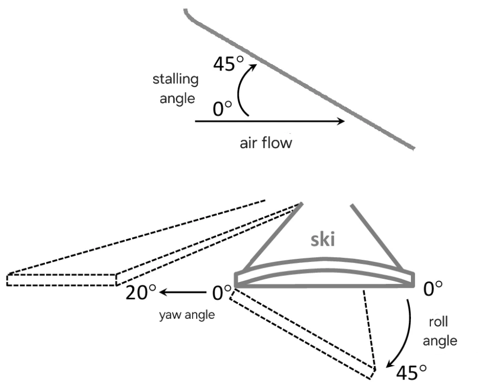

Figure 2: Different angles affecting the jump

The wind features are as follows:

WindTime - time of the wind measurement in the same format as the Time column itself. Since wind measurements are recorded less often, the wind values are applied to the most recent jump measurement and then just repeated until a new wind measurement is available. Since the wind is represented by a nonlinear function, it would be hard to capture its movements with interpolation, so we decided to drop this column.

WindName - name of the sensor (Wi for $i = 1, \dots, 12$)

WindSpeed - resulting speed of the wind in km/h

WindSpeedTangent - speed of the wind tangent measured along the x axis (hill direction) in km/h

WindTurbulence - vertical speed of the wind turbulence in km/h

WindSpeedCleanTan - wind speed tangent with turbulence removed in km/h

WindSpeedCross - speed of the wind measurement along the y axis across the hill in km/h

There are 12 wind sensors spread across the ski jump hill. To help with the analysis, we separated the jump section of the hill into 3 zones. The first zone contains wind sensors 1 to 4, the second zone contains sensors 5 to 8, and the third zone contains sensors 9 to 12 [11].

During processing, we also removed some ski jumps that were incomplete or had corrupted data, so the final dataset contained around 200 ski jumps.

3 Methodology

This section describes our research methodology. We first present different variations of the SSM that we tested for the ski jump simulation, followed by describing our model and how it predicts the jumps. Finally, we present the description of our ski jump animation app.

3.1 Different modeling approaches

In addition to pure SSM, we considered different approaches for modeling ski jumps that included classical physics-based models, but the data are not sufficient to accurately capture all the forces acting on the jumper. We also tried a hybrid approach that combined SSM and Physics-informed Neural Networks (PINNs [14]), where the SSM would provide a baseline prediction and the PINN would learn to correct any discrepancies, taking into account physical properties of the system, such as the mass of the pilot, the properties of the wind, and gravitational force [4].

These parameters are included in the equations of motion and added to the total loss function. So, the model prefers solutions that are consistent with the laws of physics. This turned out to be less effective than a pure SSM approach, but the reason exceeds the purpose of this paper. More about errors and models' comparison is given in Section 4.1.

3.2 Ski jump prediction model

In order to fit our data to the SSM, we stored the data in each file in three vectors. The main vector contains states or state variables of the system, which in our case are the X, Y, and Z coordinates, jumper velocities, and all angles (opening, stalling, roll, and yaw) [6].

The observation vector contains the measured outputs of the system, which in our case are the X, Y, and Z coordinates and height above ground. The controls contain the external inputs to the system, which in our case are the wind measurements from all the sensors that are averaged over each zone and feature (speed, tangent, cross and turbulence).

We then used ridge regression to estimate the matrices A, B, C and D of the SSM, as shown in Figure 3, where we minimized the computed values from the current and previous values and the next time-stamped values. Thus, matrix A computes the next state from the current state, B computes the next state from the current control, C computes the next observation from the current state, and D computes the next observation from the current control. We then use recursion to predict the next state from the prediction of the previous state and the current control, to get the full simulated jump. This allows us to predict the jump trajectory based on the environmental conditions and the starting state of the jumper [9].

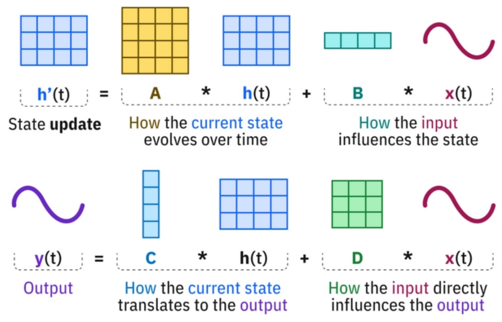
h'(t) = A * h(t) + B * x(t)
State update
How the current state evolves over time
How the input influences the state
y(t) = C * h(t) + D * x(t)
Output
How the current state translates to the output
How the input directly influences the output

Figure 3: Schema of SSM matrices [3]

3.3 Ski jump animation app

To make our results accessible beyond the research setting, we developed an interactive web application using Shiny for Python [8]. The application serves as a front-end to the trained state-space model and allows users to explore ski jump simulations under varying environmental conditions or just to observe different measured ski jumps. Firstly, through a set of input controls, users can adjust factors such as wind speed, wind directions, or different ski angles, and the application instantly updates the predicted jump trajectory. Secondly, users can simply explore random jumps from the provided dataset or upload their own CSV file of measured jumps, as long as it includes the columns described in Section 2.3.

The application presents the results as an animated visualization of the ski jump, showing the full trajectory and the final distance. In this way, the application functions both as an analytical tool, helping to test how different conditions affect performance, and as an educational resource that makes the mechanics of ski jumping easier to understand for a wider audience. It is available online. [1]

4 Main results

In this section, we present the results of our simulations. Firstly, we present a statistical comparison of all the models, followed by a precise analysis of our predictions.

4.1 Models' error

In order to evaluate different models, we first had to define a metric to measure the prediction error. Since actual and simulated jumps are represented with x, y, and z coordinates but are measured at different time stamps and can contain a different number of measurements, we had to find a way to compare them. We first tried to project the shorter trajectory on to the other one and compute the distance between the original and the projection, but this method turned out to be computationally expensive. So we decided to compute the distance between the actual and the simulated jumps by interpolating both jumps. The new measurements contain the start and end point and all the ones, where x reaches a natural value. We then compute the error as the norm of the difference between the two jumps. And after one of the jumps ends, we just add the distance from the end of the shorter jump to the end of the longer jump to the error. In this way, we penalize the model for not being able to predict the correct length of the jump.

Since we had a limited number of jumps, we used leave-one-out cross-validation to evaluate the models. For each jump, we trained the model on all other jumps and then simulated the left-out jump. We then calculated the average error between the actual jump and the simulated jump for both the training set and the test set, as shown in Figure 4.

In the process of developing our ski jump prediction model, we evaluated several variations to determine the most effective approach. We compared the performance of a pure SSM with a hybrid model that combined SSM with PINN. The pure SSM demonstrated superior predictive accuracy, probably due to its ability to directly model the temporal dynamics of ski jumps without the added complexity of PINNs. We also experimented with different configurations of the SSM, including using all available wind sensor data versus an averaged value of the zone. When we used all sensors, the average error for each point (in the training data is 1.67 m and in the test data is 1.89 m), while when we averaged the sensors over the zones, the error (in the training data was 1.76 m and in the test data 1.82 m). This suggests that averaging the wind data helps with the simulation.

4.2 Analysis of our model

Wind is a critical factor in ski jumping, so we attempted to capture its nonlinear effects by including columns for the squared wind features. However, we found that adding these squared terms did not significantly reduce the prediction error.

Since the simulation still requires numerous inputs, we made it interactive, allowing users to adjust the wind conditions and observe their impact on the jump. In the ski jumping app, users

[1] https://camlekn.shinyapps.io/ski-jump/

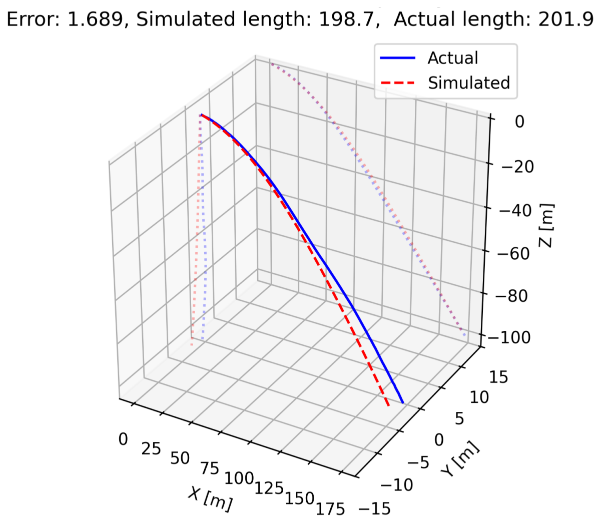

Figure 4: Actual vs. simulated ski jump trajectory

can manipulate sliders to set the wind speed, wind tangent, wind cross, and wind turbulence for each of the three zones. As a result, the wind loses its original movement function in the simulation. All other inputs are set to the average values computed from the dataset.

5 Discussion

This section examines the predictive performance of the trajectories, highlights the limitations of our current approach, and suggests directions for future improvements.

5.1 Limitations

Given the relatively small dataset of ski jumps, the main limitation of our project lies in the limited data available for training the model. After preprocessing, the dataset contained only about 200 jumps, which may limit the SSM's ability to represent the full range of trajectory variations under different circumstances. As a result, the model may struggle to accurately predict jumps under novel or extreme conditions.

Furthermore, the dataset lacks detailed information, or any information at all, about individual jumpers, such as body weight, sex, or other physiological characteristics that are known to influence jump performance. Incorporating these variables could improve model accuracy and provide more personalized predictions [5].

Lastly, due to limited computing power, only one CPU was available, restricting the use of possible better models. To address these challenges, using cloud-based resources could help run larger models and improve the prediction of trajectories.

5.2 Future work and potential improvements

Although the current approach shows promise, there are several avenues for future improvements. Some of which we are working on at the time of writing this paper.

Currently, we are working on improving the sliders' functions. Since the wind data determined by the user is static throughout the jump, this adds a lot of generalization. In reality, wind conditions can change rapidly during a jump. So we would like to add additional controls to the app that would allow the user to define how the wind changes during the jump. They could choose whether the wind would gain or lose a certain feature (such as speed or turbulence) during the jump.

Expanding the dataset to include more jumps and additional contextual information about individual jumpers could improve the accuracy of the model. We could try to generate more data by using data augmentation techniques, such as adding noise to the wind measurements or slightly modifying the angles. We could also try to find the nonlinear movements of the wind and interpolating the wind measurements by their original time stamps to better capture the wind dynamics.

6 Conclusion

This paper presents a method for predicting ski jump trajectories based on environmental conditions. By incorporating external factors into the modeling framework and applying least squares estimation, we demonstrated that the model is capable of capturing the dynamics of ski jumps and producing realistic trajectory predictions. In addition, we developed an interactive application that makes the results accessible to a broader audience through simulations and animations of predicted jumps. Although the current model is limited by the size of the dataset and the absence of certain athlete-specific variables, the results show that state-space models are a promising tool for analyzing ski jumping performance.

7 Acknowledgments

This work was supported by Smučarska Zveza Slovenije (Ski Association of Slovenia), whom we would like to thank for providing the ski jump data.

References

[1] 3DWarehouse via 3dmdb.com. 2025. "ski jumping planica" [3d model]. https://3dmdb.com/en/3d-model/ski-jumping-planica/8386000/?free=True&q=Ski+jump. Free model; accessed: 2025-08-26. (2025).

[2] Masanao Aoki. 1990. *State Space Modeling of Time Series.* (2nd, revised and enlarged ed.). *Universitext.* Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN: 978-3-642-75883-6. doi:10.1007/978-3-642-75883-6.

[3] Dave Bergmann. 2025. What is a state space model? Accessed: 2025-09-24. https://www.ibm.com/think/topics/state-space-model.

[4] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George E. Karniadakis. 2021. Physics-informed neural networks (pinns) for fluid mechanics: a review. *Acta Mechanica Sinica*, 37, 12, 1727–1738. doi:10.1007/s10409-021-01148-1.

[5] Wolfram Müller. 2008. Performance factors in ski jumping. In *Sport Aerodynamics.* CISM International Centre for Mechanical Sciences. Vol. 506. Helge Nørstrud, editor. Online ISBN: 978-3-211-89297-8. Springer, Vienna, 139–160. ISBN: 978-3-211-89296-1. doi:10.1007/978-3-211-89297-8_8.

[6] Wolfram Müller. 2006. The physics of ski jumping. Tech. rep. CERN report on the aerodynamics and physics of ski jumping. CERN. https://cds.cern.ch/record/1009275/files/p269.pdf.

[7] Bor Plestenjak. 2016. *Razširjen uvod v numerične metode.* Slovenian textbook on numerical methods. DMFA-založništvo.

[8] Posit Team. 2025. Shiny for python. Accessed: 2025-08-29. https://shiny.posit.co/py/.

[9] Serrano.Academy. 2025. State-space model (ssm) tutorial. https://youtu.be/g1AqUhP00Do. State-Space Model (SSM) video. (2025).

[10] Ski Jumping Hill Archive, skisprungschanzen.com. 2025. Letalnica (letalnica bratov gorišek), planica, slovenia — ski jumping hill archive. https://www.skisprungschanzen.com/EN/Ski+Jumps/SLO-Slovenia/Planica/0475-Letalnica/. Accessed: 2025-09-12. (2025).

[11] Ava Thompson, ed. 2025. *Ski Jumping.* Found via Google Books at https://www.google.si/books/edition/Ski_Jumping/G2pPEQAAQBAJ?hl=en&gbpv=0. Publifye AS.

[12] Wessel N. van Wieringen. 2015. Lecture notes on ridge regression. arXiv preprint arXiv:1509.09169. Revision v8, submitted 30 September 2015; revised 27 June 2023. (2015). doi:10.48550/arXiv.1509.09169.

[13] Mikko Virmavirta and Juha Kivekäs. 2019. Aerodynamics of an isolated ski jumping ski. *Sports Engineering*, 22, 1, 1–6. doi:10.1007/s12283-019-0298-1.

[14] StatQuest with Josh Starmer. 2025. Neural networks tutorial. https://youtu.be/CqOfi41LfDw. Neural networks introduction video. (2025).