A toy model to investigate stability of AI-based dynamical systems

B. Balogh[1], **D. Saint-Martin**[1], **A. Ribes**[1]

[1]CNRM, Université de Toulouse, Météo-France, CNRS, Toulouse, France

Key Points:

- A higher dimensional extension of the Lorenz'63 model is introduced to investigate instability issues.
- When trained on a single orbit, neural networks can generate unstable trajectories.
- Instability issues can be solved by using a well-designed learning sample.

Corresponding author: Blanka Balogh, `blanka.balogh@meteo.fr`

Abstract

The development of atmospheric parameterizations based on neural networks is often hampered by numerical instability issues. Previous attempts to replicate these issues in a toy model have proven ineffective. We introduce a new toy model for atmospheric dynamics, which consists in an extension of the Lorenz'63 model to a higher dimension. While neural networks trained on a single orbit can easily reproduce the dynamics of the Lorenz'63 model, they fail to reproduce the dynamics of the new toy model, leading to unstable trajectories. Instabilities become more frequent as the dimension of the new model increases, but are found to occur even in very low dimension. Training the neural network on a different learning sample, based on Latin Hypercube Sampling, solves the instability issue. Our results suggest that the design of the learning sample can significantly influence the stability of dynamical systems driven by neural networks.

Plain Language Summary

Part of atmospheric models accounting for small-scale processes, called parameterizations, can be developed by using artificial intelligence techniques, such as neural networks. The development of such parameterizations is often hampered by numerical instability issues. Toy models commonly used in atmospheric research are not complex enough to allow the study of these instabilities, and are easily learnt by neural networks. Here, we introduce a new toy model for atmospheric dynamics, which consists in an extension of the famous Lorenz'63 model to a higher dimension. While neural networks trained on a single orbit can easily reproduce the Lorenz'63 model, they fail to replicate the new toy model, leading to unstable trajectories. Training the neural network on a specifically designed learning sample, which explores the full phase space in the neighborhood of a given trajectory, solves the instability issues. Our results suggest that the design of the learning sample can significantly influence the stability of dynamical systems driven by neural networks.

1 Introduction

Many efforts have been made over the last few years to develop new, data-driven parameterization schemes using artificial intelligence (AI) for use in atmospheric models (e.g. Gentine et al., 2018; O'Gorman & Dwyer, 2018; Brenowitz & Bretherton, 2018). Even though they promise numerically affordable yet accurate physics for low resolution

atmospheric models (e.g. climate models), current state-of-the-art AI parameterizations are still biased and, more importantly, they face numerical instability issues. As reported by Rasp (2020), neural networks (NNs) are often numerically unstable, when coupled to the large-scale atmospheric fluid mechanics solver (e.g. Rasp et al., 2018; Brenowitz & Bretherton, 2019). Parameterizations based on random forests (RF) have been reported to be stable (Yuval & O'Gorman, 2020). But, when compared offline, NN-based parameterizations seem to outperfom RF-based parameterizations (Brenowitz, Henn, et al., 2020).

Stability issues can be interpreted as a result of breaking physical laws, and some authors propose techniques to ensure compliance with these laws (e.g. Beucler et al., 2019). Brenowitz, Beucler, et al. (2020) shows that instabilities are related to the linearized behavior of NNs when coupled to idealized wave dynamics. To fix numerical instabilities, Rasp (2020) proposes a coupled online learning method where a NN model, which was first trained offline, is continuously corrected according to online prediction errors. This concept is illustrated using the Lorenz'96 model (Lorenz, 1996).

Lorenz'96 (hereafter L96) and Lorenz'63 (Lorenz, 1963, hereafter L63) models often serve as toy models for atmospheric modeling, drawing simplified versions of the atmospheric flow. They have been extensively used as a test bed for assessing the use of data-driven methods in atmospheric models. L63 model has been accurately learnt by feedforward neural networks (Scher & Messori, 2019) or reservoir networks (Pathak et al., 2017). L96 dynamics has also been accurately learnt by feedforward NNs, recurrent NNs or generative adversarial networks (Dueben & Bauer, 2018; Chattopadhyay et al., 2020; Gagne et al., 2020), without reporting stability issues. Both L63 and L96 toy models appears to be insufficiently complex to study numerical instabilities encountered with more complex systems.

Understanding the instability encountered in the development of data-driven parameterizations is of outmost importance for any climate application. However, no toy model for atmospheric dynamics is currently available to investigate such instability issues. The first aim of this paper is to introduce a toy model for atmospheric dynamics, consisting in a higher dimensional version of L63, which exhibits instability when learnt by a NN. The second objective is to propose a possible method to ensure stability of the NN-generated trajectories, which in our case involves using a specific learning sample. Both objectives can be seen as steps towards solving numerical instabilities – an issue

which remains challenging to develop suitable NN-based parameterizations for atmospheric models.

Section 2 describes the Lorenz'63 model and illustrates that this model is not complex enough to challenge NNs. In Section 3, we introduce the embedded Lorenz'63 model, and show that learning of this simple model with NNs leads to unstable trajectories. In Section 4, a method to address these instability issues is proposed. Finally, conclusions and discussions will be drawn in Section 5.

2 Learning the Lorenz'63 model

By expanding the set of partial differential equations of Rayleigh-Bénard convection into Fourier series and then truncating, Lorenz (1963) derives a finite system of ordinary differential equations, given by:

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= x_1(\rho - x_3) - x_2, \\ \dot{x}_3 &= x_1 x_2 - \beta x_3 + x_1. \end{aligned} \tag{1}$$

The evolution of the state variable $\mathbf{x} = (x_1, x_2, x_3)$ is governed by the set of parameters (σ, ρ, β). L63 admits a chaotic solution for some values of these parameters, in particular $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$. We use these values in the following. Although computationally cheap, L63 is able to mimic some properties of the atmospheric flow. Hence, it is a suitable toy model to perform proof-of-concept experiments.

Recently, the L63 model has been used as a test bed for assessing the use of data-driven methods in atmospheric models. The system of equations (1) can be formally rewritten:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)). \tag{2}$$

Data-driven methods provide an approximate function, $\widehat{f}$, to replace the 'true' function f, and thus they replace the initial dynamical system by:

$$\dot{\mathbf{x}}(t) = \widehat{f}(\mathbf{x}(t)). \tag{3}$$

That is, learning a dynamical system usually involves learning its derivative. For the L63 model specifically, various functions $\widehat{f}$ have been proposed recently, e.g., using sparse linear regression (Brunton et al., 2016), feedforward neural networks (Scher & Messori, 2019), or reservoir networks (Pathak et al., 2017). Fitting $\widehat{f}$ always involves an optimization

over a set of free parameters θ. This optimization is done for a given loss function, and a given learning sample. In the case of dynamical systems, the learning sample is typically a time series, also called an *orbit*, obtained by numerical integration of the system, e.g., Eq. (2). We note $[\mathbf{x}^{orb}]$ this learning sample:

$$[\mathbf{x}^{orb}] = \{(\mathbf{x}^n, f(\mathbf{x}^n))\}_{n=1..N}, \tag{4}$$

where $\mathbf{x}^n$ denotes the state of the system, $\mathbf{x}(t_n)$, at time t_n.

To learn the L63 model, we use one single orbit as a learning sample. This orbit is obtained by integrating Eqs. (1) with a time-step of $\Delta t = 0.05$ using a fourth-order Runge-Kutta time stepping scheme. The numerical integration is performed over 500 model time units (MTU, where 1 MTU = 20 Δt). Thus, the learning sample contains $N = 10000$ individuals. A simple, eight-layer feedforward artificial neural network is trained to derive $\widehat{f}$, using the mean squared error as the loss function. The integration of both the L63 model (Eq. 2) and its NN-approximation (Eq. 3), starting from the same initial condition, shows a good agreement (Fig. 1). In particular, the NN-based model does not manifest any unstable behavior, i.e., its orbits always lay on the Lorenz 'butterfly' attractor. Repeating this analysis with different initial conditions and/or different learning samples leads to the same conclusion. This result suggests that the L63 model is not challenging enough to data-driven methods and does not allow to study instability issues, as previously shown in similar studies.

L96 provides another simple framework to study data-driven methods in the context of atmospheric modeling. L96 is designed to be more complex than L63, owing to its larger number of state variables. However, reservoir computers and neural networks have succeeded in learning this system without reporting instability issues. Hence, both L63 and L96 appears to be too simple to encounter the typical instability issues that still hamper the development of AI parameterizations.

3 The embedded Lorenz'63 model

Building a higher dimensional model on the basis of L63 has already been proposed. Musielak and Musielak (2009) added a fourth spatial variable to extend L63 equations. Champion et al. (2019) created a high-dimensional model on the basis of L63 system by using Legendre polynomials. Here, we propose a different extension of this model specifically designed to investigate instability issues.

The key idea is as follows: the standard L63 model, which is of dimension 3, is embedded into a larger space of dimension $d > 3$. In the selected 3-D subspace, the chaotic dynamics of L63 is kept unchanged. Any deviation from this 3-D subspace is brought back by a restoring force. We call 'embedded' L63 model (hereafter, eL63) this new simple model. A formal definition of eL63 is in two steps (see Figure 2).

(1) Embedding: we construct a vector $\mathbf{z} = (z_1, z_2, ..., z_d) \in \mathbb{R}^d$, and define its dynamics by:

$$\begin{aligned} \dot{z}_1 &= \sigma(z_2 - z_1), \\ \dot{z}_2 &= z_1(\rho - z_3) - z_2, \\ \dot{z}_3 &= z_1 z_2 - \beta z_3 + z_1, \\ \dot{z}_j &= -\kappa z_j, \quad \forall j > 3. \end{aligned} \tag{5}$$

The first three equations are identical to L63, while the $d - 3$ additional equations are simple restoring forces. For the sake of simplicity, the relaxation coefficient, κ, is unique. In the following, we use $\kappa = 1$. We denote $\mathcal{B}_z$ the basis of vector $\mathbf{z}$.

(2) Random rotation: We apply a random rotation to derive the state vector of the eL63 system, $\mathbf{x} = (x_1, x_2, ..., x_d)$:

$$\mathbf{x}(t) = P\mathbf{z}(t), \tag{6}$$

where $P \in \mathbb{R}^{d \times d}$ is the rotation matrix between $\mathcal{B}_z$ and $\mathcal{B}_x$, the basis of $\mathbf{x}$. Note that the rotation matrix P does not depend on time.

A key property of eL63 is that $z_{j,j>3}$ exponentially decays towards zero and so $\mathbf{x}(t)$ is confined within a subspace of dimension 3. This subspace can be interpreted as resulting, e.g., from physical but unknown constraints. The difficulty for a data-driven method to fully capture the dynamics of eL63 comes from the fact that any orbit is very thin – almost all points are located in a subspace of dimension 3. If such an orbit is used as a learning sample, any deviation from this specific subspace in a predicted trajectory can lead to an out-of-sample issue.

We now consider the problem of learning eL63, in the same way as done for L63 in the previous section. A learning sample $[\mathbf{x}^{orb}]$ is built by integrating eL63 equations over 500 MTUs, with a timestep of $\Delta t = 0.05$ using a fourth-order Runge-Kutta scheme. The initial condition is sampled from an eL63 orbit, which is equivalent to removing the

model spin-up from the learning dataset, ensuring that $z_{j,j>3} \approx 0$. Therefore, the learning dataset contains $N = 10000$ individuals. Both the input and the target variables are then normalized so that their mean value equals 0 and their standard deviation equals 1. They are randomly partitioned into training and testing datasets, with proportions 80% and 20%, respectively. A simple feedforward NN is trained to best approximate f. The NN is eight layers deep and of similar complexity to the NN that has been utilized to learn the L63 dynamics in Section 2. All layers are activated by Rectified Linear Unit (ReLU) function. With an embedding dimension $d = 4$, this neural network has 176458 free parameters to fit. Learning is performed over 50 epochs, with Keras implementation of Adam optimizer, parameterized with a initial learning rate of 0.001. During training, the determination coefficient R^2 over the test subset is monitored. The training loss function $\mathcal{L}$ is the mean squared error:

$$\mathcal{L}(\theta, [\mathbf{x}^{\text{train}}]) = \frac{1}{N} \sum_{n=1}^{N} \left\| \widehat{f}_\theta(\mathbf{x}^n) - f(\mathbf{x}^n) \right\|^2, \tag{7}$$

where $[\mathbf{x}^{\text{train}}]$ denotes the training dataset, N the size of the training dataset, $\mathbf{x}^n \in [\mathbf{x}^{\text{train}}]$ for $1 \leq n \leq N$, and θ denotes the set of parameters over which the optimization is made. As the learning sample consists of a single orbit, we note $\widehat{f}_{orb}$ the estimated function, i.e.:

$$\widehat{f}_{orb} = \widehat{f}_{\theta^*} \text{ with } \theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, [\mathbf{x}^{orb}]). \tag{8}$$

Learning is very efficient, as we get $\mathrm{R}^2 = 0.9998$, i.e., an overall performance consistent with L63. To assess the stability of the trajectories generated by using $\widehat{f}_{orb}$ instead of f, we take an initial condition within the original learning sample $[\mathbf{x}^{orb}]$. Then, we generate the ANN-based trajectories starting from these points over 1000 MTUs, and compared them with the true eL63 orbits (i.e., integrating eL63 equations from the same initial condition). This seems long enough for a model to manifest numerical instabilities. Stability of the resulting orbit is assessed using a 'stability criterion' defined as follows: for each $i = 1..d$, we compute the minimum (m_i) and maximum (M_i) values of x_i over the training orbit $[\mathbf{x}^{orb}]$. A N-step trajectory is considered as stable if it remains within 7 times this range of values, i.e.:

$$m_i - 3(M_i - m_i) \leq x_i^n \leq M_i + 3(M_i - m_i),\ \forall\ 1 \leq i \leq d,\ \forall\ 1 \leq n \leq N. \tag{9}$$

The choice of this stability criterion is partly arbitrary, but it is motivated by its simplicity.

The above described validation strategy is iterated over 100 different $\widehat{f}_{orb}$, each being trained with a different rotation matrix P and a corresponding learning sample $\left[\mathbf{x}^{orb}\right]$. The resulting functions $\widehat{f}_{orb}$ are then tested by generating orbits of length 1000 MTU from 30 different initial conditions sampled randomly from their respective training orbit. As a result, stability can be assessed over 3000 simulated orbits of length 1000 MTU each. Figure 3 shows the percentage of stable trajectories generated with $\widehat{f}_{orb}$ for different values of d, the dimension of eL63. Even with minimal embedding (i.e., $d = 4$), 40% of the NN-generated orbits are unstable. With $d \geq 7$, all generated orbits are unstable regarding the stability criterion defined by Eq. (9). The accumulation of small prediction errors gradually leads the NN-generated orbit away from the learning sample, in a region where $\widehat{f}_{orb}$ is not accurate. Hence, many NN-generated orbits are unstable, proving that eL63 is a very simple model (as its dimension remains very low) which is able to reproduce instability issues. Lastly, we notice that the same recipe (i.e., embedding) can be applied to even simpler non-chaotic dynamical system, and leads to similarly unstable trajectories when learnt by NNs (see Supplementary Information).

4 Stabilizing the NN-based embedded Lorenz'63 model

We now propose to illustrate a possible method to solve instability issues encountered by simple NN models in the case of the eL63 model. We generate a new learning sample by taking points away from a typical orbit, taking advantage of the fact that the value of f can be sampled at any location. In this way, we better approximate the function f on regions away from the eL63 attractors. This method can be thought as a data augmentation technique. Pan and Duraisamy (2018) yet proposes data augmentation as a remedy to tackle instability issues encountered in the dynamical system they study.

Here, we use a Latin Hypercube Sampling (LHS) (McKay, 1992) to generate a new learning sample of size $N = 10000$ (i.e., N is kept unchanged). LHS generates an optimal near-random sample in a high-dimensional (here, dimension d) hypercube. The boundaries of the LHS are set to 1.5 times the range of an orbit $[\mathbf{x}^{orb}]$. This calculation is done for each $i = \{1, ..., d\}$, leading to an hypercube in the basis $\mathcal{B}_x$. Finally, we generate the target variables by simply applying f to the selected points, and obtain a new learning sample $[\mathbf{x}^{\mathrm{LHS}}]$. The next step is to estimate f from $[\mathbf{x}^{\mathrm{LHS}}]$. Consistent with the previous section, train and test datasets are obtained by randomly partitioning the learning sample in proportions of 80% and 20%, respectively. A feedforward neural network

is trained, and we denote

$$\widehat{f}_{\text{LHS}} = \widehat{f}_{\theta^*} \text{ with } \theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, [\mathbf{x}^{\text{LHS}}]). \tag{10}$$

the new estimate of f. Learning over the LHS sample is slightly less efficient than learning from a single orbit dataset, as we find $R^2 = 0.9975$. This is an expected outcome, as the region covered by the learning sample is wider in the case of $[\mathbf{x}^{\text{LHS}}]$, resulting in more complex variations of f.

100 $\widehat{f}_{\text{LHS}}$ are trained over different learning datasets. The estimate functions then generate orbits of length 1000 MTU from 30 initial conditions that has been randomly sampled from $\left[\mathbf{x}^{orb}\right]$. The same stability criterion (Eq. 9) is applied to determine whether the $\widehat{f}_{\text{LHS}}$ orbit is stable. Contrary to orbits generated with $\widehat{f}_{orb}$, all orbits generated with $\widehat{f}_{\text{LHS}}$ remain stable over 1000 MTUs (see Fig. 3) with $d = 4, \ldots, 10$. This result suggests that stability can be a property of the learning sample – and not only of the type of learning algorithm or of intense tuning of hyperparameters. It also shows that extending the training beyond the thin phase space region explored by a single trajectory is important to improve the long-term performance (in particular stability skill) of a NN-based dynamical model.

However, one caveat is worth mentioning. Even though the generated trajectories are stable, in some cases, they collapse onto a fixed point near the center of one of the eL63 attractors. This seems to happen more frequently as the embedding dimension d increases. It can suggest that the size of our learning sample (i.e., 10000 individuals) may not be sufficient when the embedding dimension increases. This finding could motivate further research on how to improve the design of the learning sample in such problems.

5 Conclusion and discussion

We have developed an extended version of the Lorenz'63 model by embedding this model into higher dimension d. This new model is called the embedded Lorenz'63 model. Using a real trajectory (orbit) as a learning sample, simple artificial neural network can successfully learn the time derivatives $\dot{\mathbf{x}}$ of either L63 or eL63 models. However, unlike in L63, long trajectories generated by NNs are unstable in eL63. Instability is observed even with a minimal dimension increase, i.e., $d = 4$, and becomes more frequent as d increases. As a result, eL63 is a first example of low-dimension toy model for atmospheric dynamics that can be used to investigate the stability of NN-generated trajectories. In-

troduction of this new model is important, because previous attempts to construct a toy model able to replicate instability issues have proven ineffective.

Using NNs of similar complexity but with a different learning sample, specifically designed using a latin hypercube sample, solves the instability issue. We interpret this result as follows. A typical eL63 orbit converges towards the model's attractor very quickly, leading to degeneracy – the orbit stays confined into a subspace of dimension 3. Conversely, a LHS generates points in the full eL63 space of dimension d, which allows the NN to learn the restoring force playing away from the attractor. This result is important, as it suggests that the design of the learning sample can largely influence the stability of the NN-generated trajectories. This finding also suggests that the design of the learning sample might be an important and potentially overlooked step in IA-based atmospheric modelling. As opposed to this, much literature to date has focused on improving the learning technique in order to ensure stability.

An important further question is whether or not this new toy model can provide helpful guidance for real-world problems, i.e., developing full-complexity atmospheric parameterizations based on NNs. In our view, the response is unclear at this stage, as a few questions remain open. Is the real world as much degenerated as is eL63? The fact that any orbit gets confined into a low-dimension subspace can mimic unknown physical laws. There is a growing literature on this topic. Recent studies show that NNs cannot learn exact physical laws, typically inducing drifts in the generated trajectories (e.g. Greydanus et al., 2019). Various learning techniques have been proposed to account for such physical laws. However, the degeneracy could also result from a fast equilibrium in response to restoring forces, just as assumed in eL63 – an issue which could be more difficult to address, as physical knowledge is probably more difficult to incorporate in this case. So, if some degeneracy is likely to happen in the real-world, its strength is not well determined.

Could real-world application benefit from using designed learning samples rather than 'samples of opportunity' like single orbits? The idea of building a specific learning sample can be viewed as a data augmentation strategy. This technique first requires ability to calculate the value of f at any given location. Additionally, using a LHS strategy as suggested above can be computationally demanding, especially if the dimension of $\mathbf{x}$

is large. In particular, we note that real atmospheric parameterizations involve a dimension d much higher than those investigated with our toy model.

Acknowledgments

Code is made available at : `https://doi.org/10.5281/zenodo.4331711`.

References

Beucler, T., Pritchard, M., Rasp, S., Gentine, P., Ott, J., & Baldi, P. (2019). Enforcing analytic constraints in neural-networks emulating physical systems. *arXiv:1909.00912 [physics]*. (arXiv: 1909.00912)

Brenowitz, N. D., Beucler, T., Pritchard, M., & Bretherton, C. S. (2020). Interpreting and stabilizing machine-learning parametrizations of convection. *J. Atmos. Sci.*, 1–55. doi: 10.1175/JAS-D-20-0082.1

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, *45*(12), 6289–6298. doi: 10.1029/2018GL078510

Brenowitz, N. D., & Bretherton, C. S. (2019). Spatially extended tests of a neural network parametrization trained by coarse-graining. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2728–2744. doi: 10.1029/2019MS001711

Brenowitz, N. D., Henn, B., McGibbon, J., Clark, S. K., Kwa, A., Perkins, W. A., ... Bretherton, C. S. (2020). Machine learning climate model dynamics: offline versus online performance. *arXiv:2011.03081 [physics]*. (arXiv: 2011.03081)

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc Natl Acad Sci USA*, *113*(15), 3932–3937. doi: 10.1073/pnas.1517384113

Champion, K., Lusch, B., Kutz, J. N., & Brunton, S. L. (2019). Data-driven discovery of coordinates and governing equations. *Proc Natl Acad Sci USA*, *116*(45), 22445–22451. doi: 10.1073/pnas.1906995116

Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlin. Processes Geophys.*, *27*(3), 373–389. doi: 10.5194/npg-27-373-2020

Dueben, P. D., & Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geosci. Model Dev.*, *11*(10), 3999–4009. doi: 10.5194/gmd-11-3999-2018

Gagne, D. J., Christensen, H. M., Subramanian, A. C., & Monahan, A. H. (2020). Machine learning for stochastic parameterization: generative adversarial networks in the Lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, *12*(3), e2019MS001896. doi: 10.1029/2019MS001896

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., & Yacalis, G. (2018). Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, *45*(11), 5742–5751. doi: 10.1029/2018GL078202

Greydanus, S., Dzamba, M., & Yosinski, J. (2019). Hamiltonian neural networks. *arXiv:1906.01563 [cs]*. (arXiv: 1906.01563)

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*, 130–141. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2

Lorenz, E. N. (1996). Predictability: a problem partly solved. *Proc. ECMWF Seminar on Predictability*, *Vol. I, Reading, United Kingdom, ECMWF*, 1–18.

McKay, M. D. (1992). Latin Hypercube Sampling as a tool in uncertainty analysis of computer models. *Proceedings of the 24th Conference on Winter Simulation*, 557–564. doi: https://doi.org/10.1145/167293.167637

Musielak, Z. E., & Musielak, D. E. (2009). High-dimensional chaos in dissipative and driven dynamical systems. *Int. J. Bifurcation Chaos*, *19*(09), 2823–2869. doi: 10.1142/S0218127409024517

O'Gorman, P. A., & Dwyer, J. G. (2018). Using machine learning to parameterize moist convection: potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, *10*(10), 2548–2563. doi: 10.1029/2018MS001351

Pan, S., & Duraisamy, K. (2018). Long-Time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, *2018*, 1–26. doi: 10.1155/2018/4801012

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos*, *27*(12), 121102. doi: 10.1063/1.5010300

Rasp, S. (2020). Coupled online learning as a way to tackle instabilities

and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0). *Geosci. Model Dev.*, *13*(5), 2185–2196. doi: 10.5194/gmd-13-2185-2020

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *PNAS*, *115*(39), 9684–9689. doi: 10.1073/pnas.1810286115

Scher, S., & Messori, G. (2019). Generalization properties of feed-forward neural networks trained on Lorenz systems. *Nonlin. Processes Geophys.*, *26*(4), 381–399. doi: 10.5194/npg-26-381-2019

Yuval, J., & O'Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nat Commun*, *11*(1), 3295. doi: 10.1038/s41467-020-17142-3

(i) Lorenz '63 orbit (ii) NN-based Lorenz '63 orbit

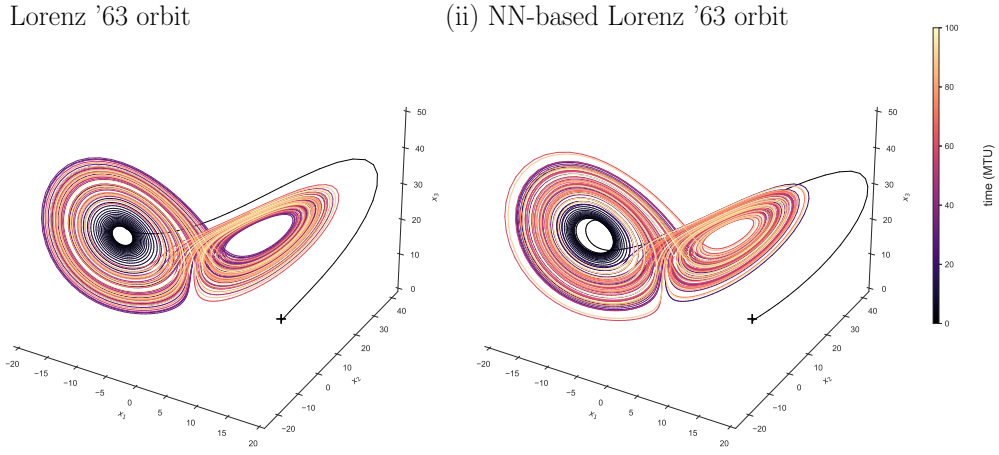
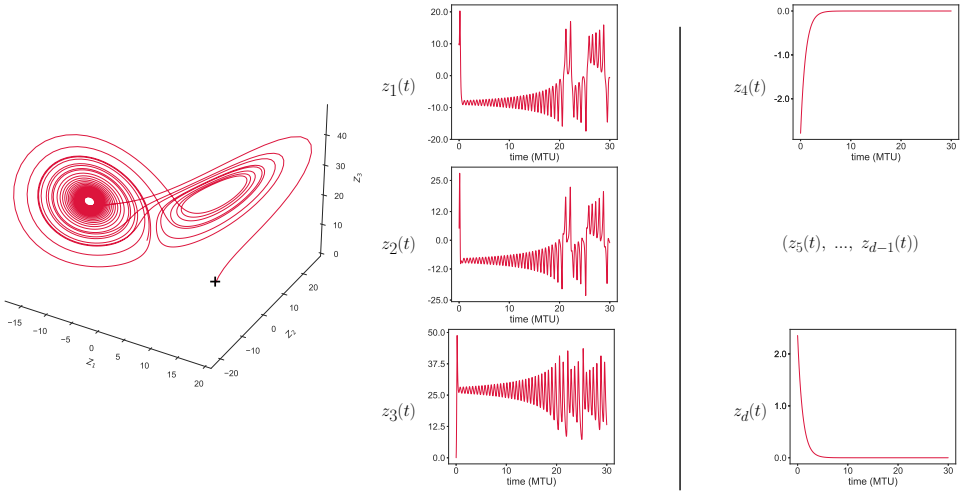


Figure 1. Examples of Lorenz'63 orbits. (i) The orbit is obtained by integration of Lorenz'63 system of equations (see Eq. 1 in the text) with σ=10, ρ=28 and β=8/3. (ii) The orbit results from the integration of neural network model $\widehat{f}$ (see Eq. 3). The numerical integration is performed over 100 model time units from initial condition $(10, 15, 0)$. The orbits are colored by the time variable.

(i) Embedding : $\mathbb{R}^3 \mapsto \mathbb{R}^d$

$$(z_1(t), z_2(t), z_3(t)) \mapsto \mathbf{Z}(t) = (z_1(t), z_2(t), z_3(t), ..., z_d(t))$$



(ii) Random rotation : $\mathbb{R}^d \mapsto \mathbb{R}^d$

$$\mathbf{X}(t) = (x_1(t), x_2(t), x_3(t), ..., x_d(t)) = P\mathbf{Z}(t),\ P \in \mathbb{R}^{d\times d}$$

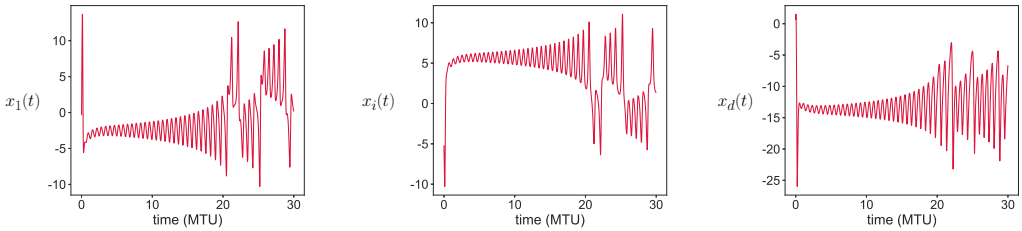


Figure 2. The embedded Lorenz'63 model (see text for details). (i) Representation of an eL63 orbit in $\mathcal{B}_z$ basis over 30 MTU : (left) tri-dimensional representation of $(z_1(t), z_2(t), z_3(t))$, (middle) time series of $z_1(t)$, $z_2(t)$, and $z_3(t)$, (right) time series of $z_4(t)$ and $z_d(t)$. (ii) Representation of the same eL63 orbit in $\mathcal{B}_x$ basis. Time series of $x_1(t)$, $x_i(t)$ and $x_d(t)$ are shown.

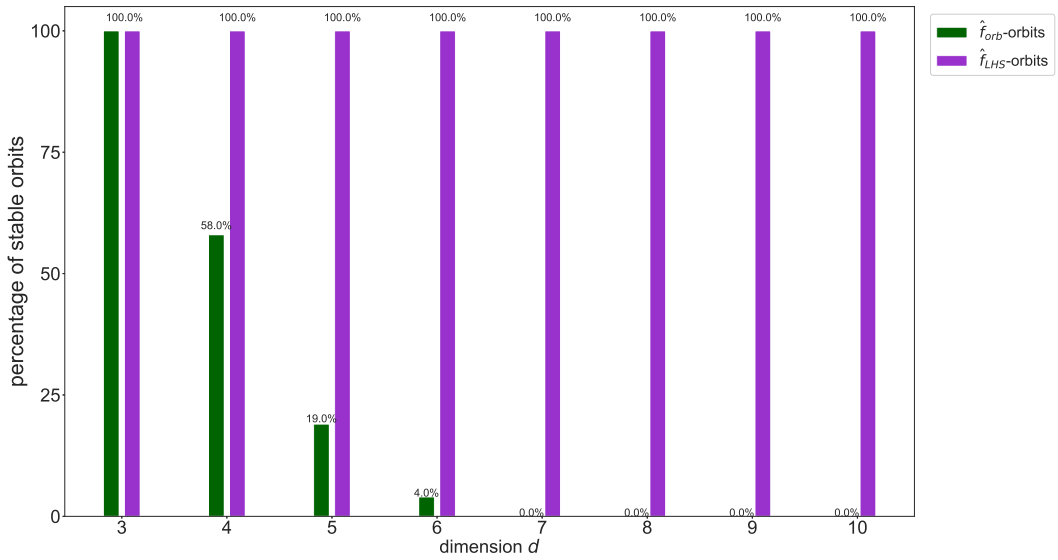


Figure 3. Percentage of stable orbits generated with $\widehat{f}$ when trained with orbital ($\widehat{f}_{orb}$, green) or LHS ($\widehat{f}_{\mathrm{LHS}}$, purple) learning samples, as a function of embedding dimension d. Initial conditions are randomly sampled either in an eL63 orbit or in a region around the eL63 attractor. Stability is assessed with 100 different $\widehat{f}$ and 30 initial conditions for $d \in \{3, 4, 5, 6, 7, 8, 9, 10\}$, using the stability criterion defined in Eq. (9).

Figure 1.

(i) Lorenz '63 orbit

(ii) NN-based Lorenz '63 orbit

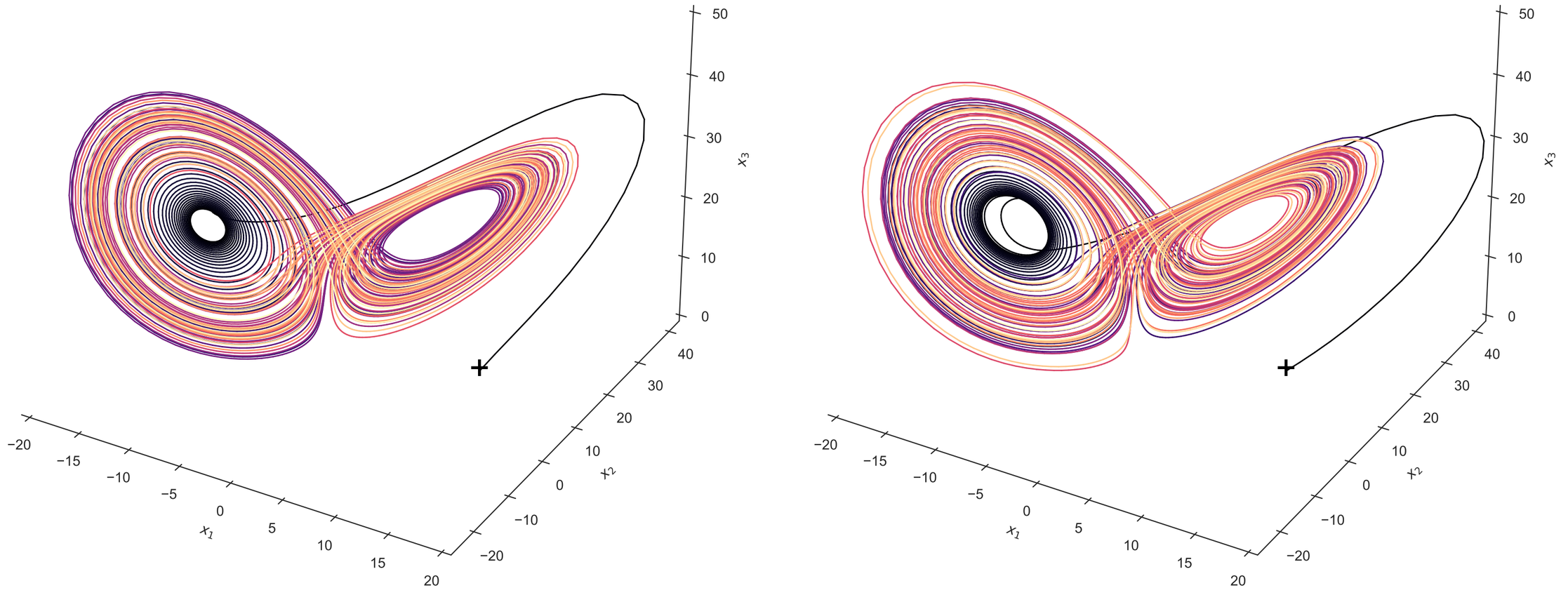
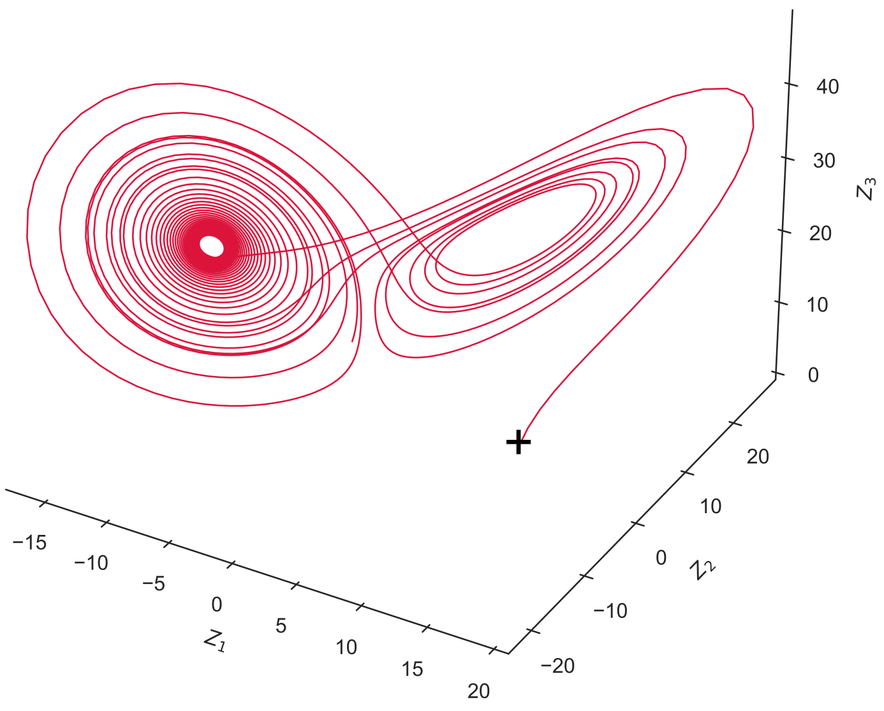
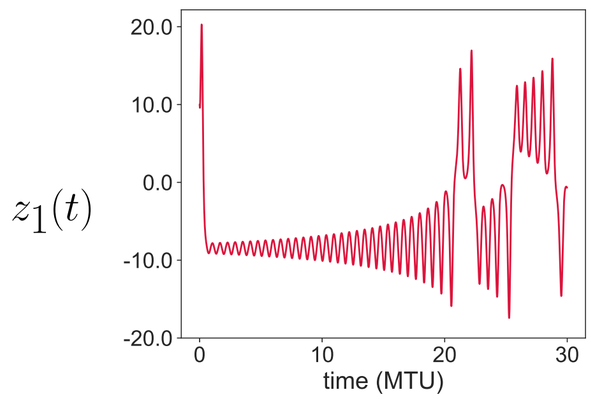


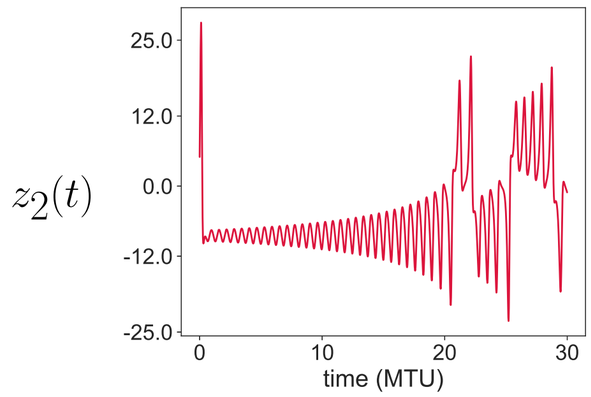
Figure 2.

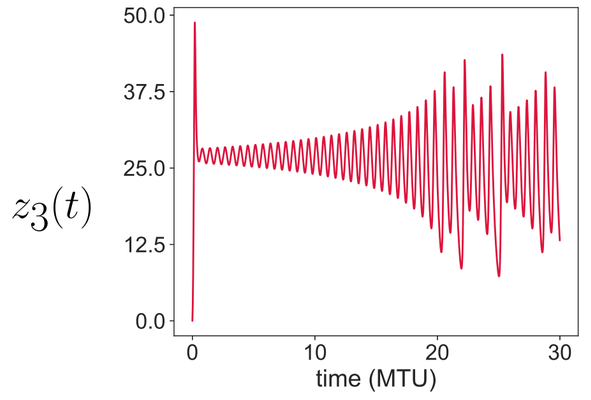
(i) Embedding : $\mathbb{R}^3 \mapsto \mathbb{R}^d$

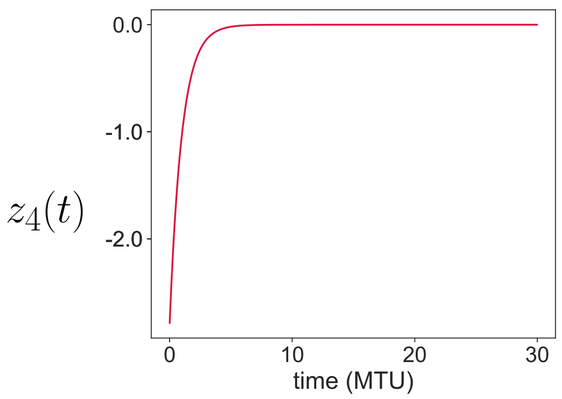
$$(z_1(t), z_2(t), z_3(t)) \mapsto \mathbf{Z}(t) = (z_1(t), z_2(t), z_3(t), ..., z_d(t))$$



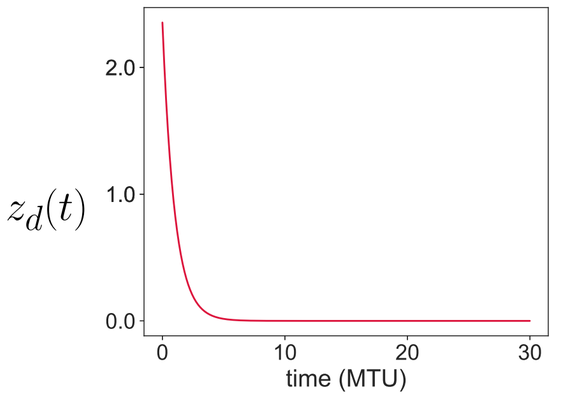






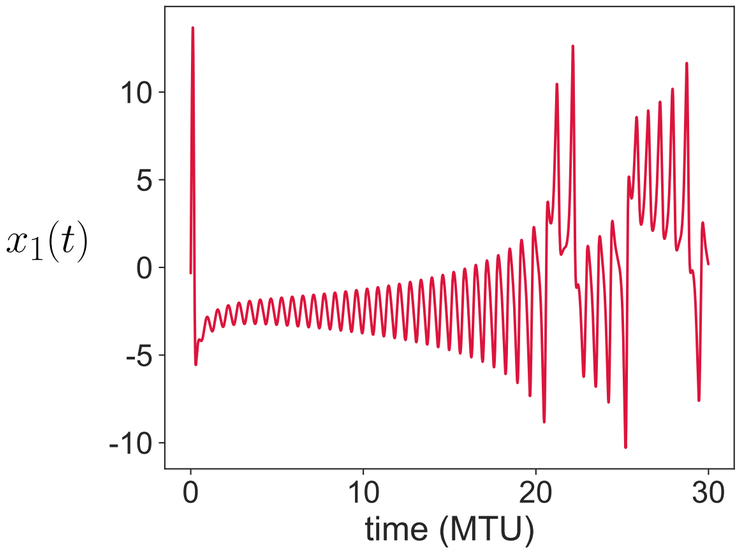


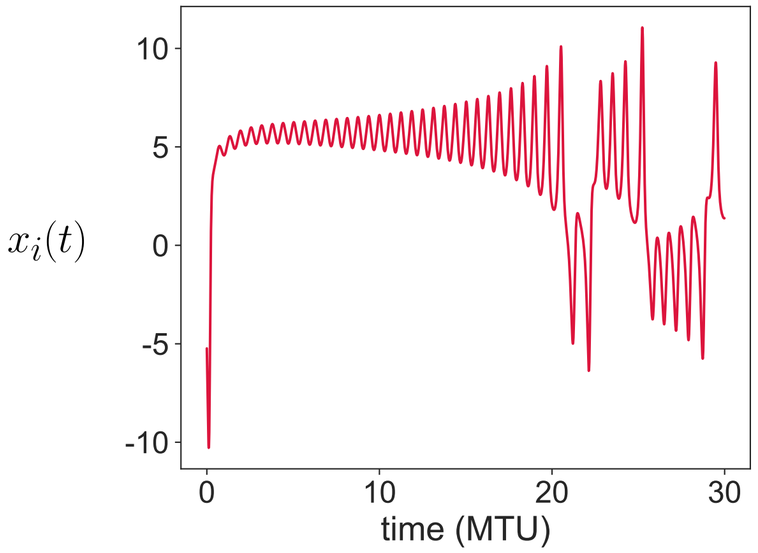
$(z_5(t),\ ...,\ z_{d-1}(t))$



(ii) Random rotation : $\mathbb{R}^d \mapsto \mathbb{R}^d$

$$\mathbf{X}(t) = (x_1(t), x_2(t), x_3(t), ..., x_d(t)) = P\mathbf{Z}(t),\ \ P \in \mathbb{R}^{d \times d}$$





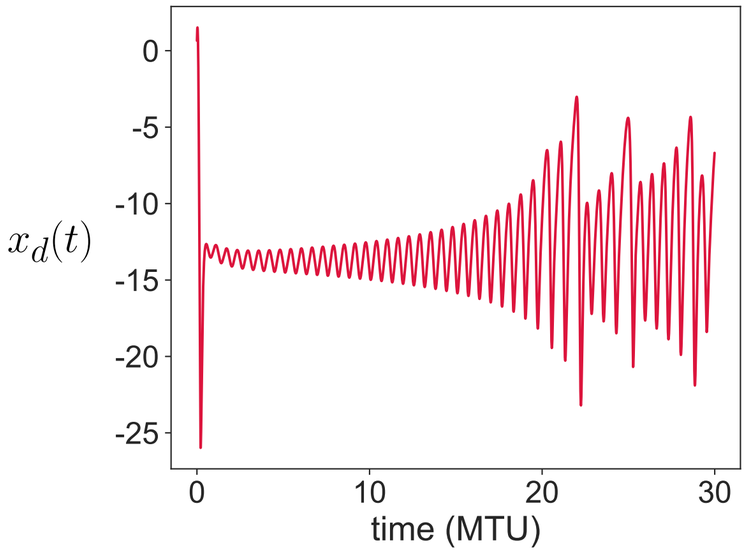
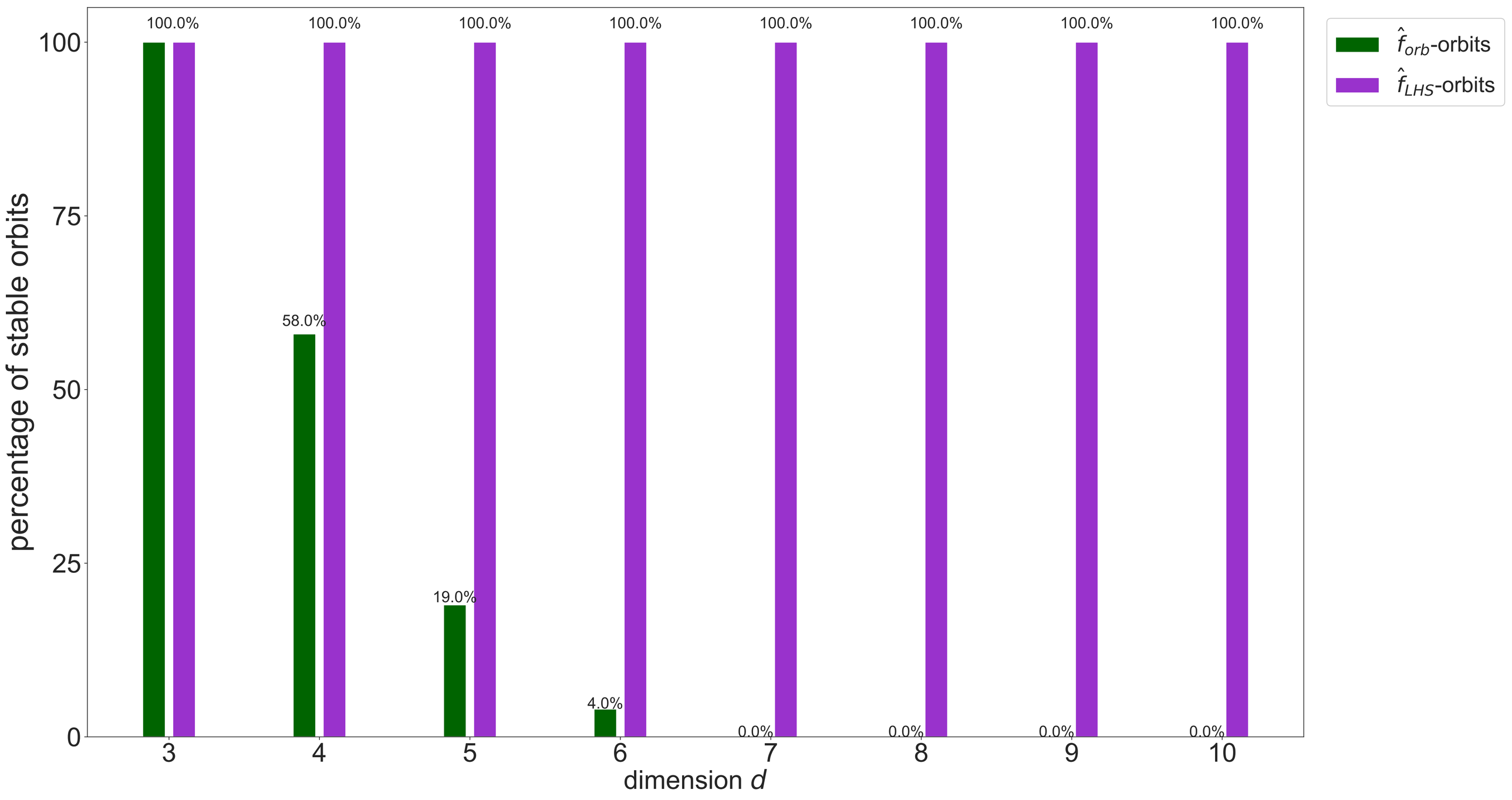


Figure 3.


100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
58.0%
19.0%
4.0%
0.0%
0.0%
0.0%
0.0%
100
75
50
25
0
3
4
5
6
7
8
9
10
percentage of stable orbits
dimension d
$\hat{f}_{orb}$-orbits
$\hat{f}_{LHS}$-orbits