# Experimental study of Neural ODE training with adaptive solver for dynamical systems modeling

Alexandre Allauzen
Thiago Petrilli Maffei Dardis
Hannah Plath

## Objectives

The goal of this paper is to explore the issue of increasing computational cost to evaluate a model according to the increasing complexity of the problem, in the case of modern ODE solvers.

## Neural ODE Model

The Lorenz'63 system is described by an ODE which makes a Neural ODE a good alternative of framework to be considered for learning its generative model.

The goal of the network it's to learn the $\dot{x} = f_\theta(x)$ where $\theta$ is the set of parameters that define the arbitrary architecture $f$. This means that, in order for the Neural Network to make an inference and compute the output, it is necessary to apply a numerical solver (ODE_Solver).

For the presented work, the value of $x$ at the moment $i$ is computed taken into account the value of $x_{i-1}$, that is:

$$x_i = ODE\_Solve(f_\theta, \tilde{x}_i)$$

The minimised loss function used was the Mean-Squared-Error:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \sum_{i=1}^{N} ||\tilde{x}_i - x_i||^2$$

## Fehlberg's method

Our method requires three evaluations of $f_\theta$ defined as follows for the step-size denoted by $h$:

$$f_1 = f_\theta(x_i), \qquad f_2 = f_\theta(x_i + hf_1), \qquad f_3 = f_\theta(x_i + \frac{h}{4}[f_1 + f_2])$$

With these three evaluations, two approximations to the next point can be obtained:

$$A_1 = x_i + \frac{h}{2}[f_1 + f_2] \text{ (RK2 method), and } A_2 = x_i + \frac{h}{6}[f_1 + f_2 + 4f_3] \text{ (RK3).}$$

Based on the approximation of these two algorithms, we can estimate the following error:

$$r = \frac{|A_1 - A_2|}{h} \approx Kh^2$$

If this error exceeds a $\epsilon$ tolerance, hypothesis $A_2$ is rejected and we recompute it with a new step-size $h' = S \times h\sqrt{\epsilon/r}$ with $S$ a safety factor. In our case, initial $h$ was 1, $S$ = 0.9 and $\epsilon$ = 0.1.

## Experiments

For the firsts experiments, an ODE Model was trained with L-BFGS wrapped in the Fehlberg's model. For simplicity, a two layer with size 50 feed-forward network was used. The chosen activation function was ReLu.

The image below shows the results for different time windows. As it can be seen, although the model partially captures the "butterfly" effect of the dynamical system, it is not successful in reproducing the generated data.

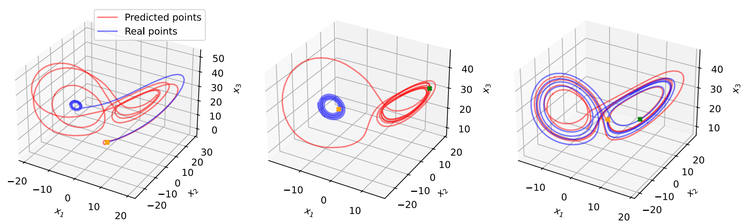


Figure 1: The trajectories predicted after a regular training, using the Fehlberg's solver as a black box.

Further investigating these results, we found out that while the MSE error at the beginning of the time series is small, it introduces time distortions as times goes on. From figure 2, it is also clear that the step adaptation tool is not being well explored as the step is equal to one most of the time.

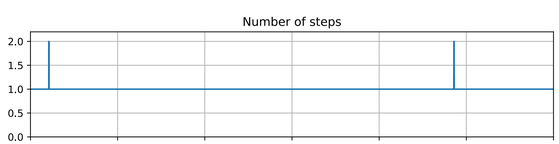


Figure 2: Time evolution of the number of steps.

From figure 3, we can see that this is consequence of the Fehlberg's method high acceptance of the RK3 predictions with one step. This is connected to the fact that the random initialisation starts of the values around zero, making the initial estimated error smaller than $\epsilon$ most of the time.

## Fehlberg's training

As an alternative solution for the training procedure, we changed the way the truncated error is calculated in the Fehlberg's method. Instead of using the raw estimated value $A_1$ to predict $x_i$ from $\tilde{x}_{i-1}$, we used directly the target value $\tilde{x}_i$. As shown in the second column of figure 3, this change impacts greatly the training process: the proportion of accepted hypothesis increases with time, reaching a maximum of 98%. The number of steps this time starts at 7 and decreases slowly until it reaches a value below two.

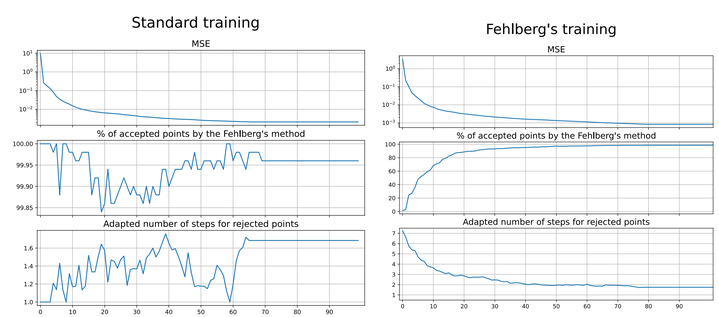


Figure 3: Time evolution for two training conditions of the MSE loss; the percentage of accepted hypotheses $A_2$; the new number of steps for the rejected hypotheses (before rounding).

## Conclusion

Through this work, we investigated that the use of a solver as a black box hinders the advantages of the adaptive step tool. We showed that, by increasing the interaction between the solver and the training process, the efficiency of the adaptive step is increased.

### References

R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. K. Duvenaud (2018). "Neural Ordinary Differential Equations". In: In Advances in Neural Information Processing Systems 31, pages 6571–6583, 2018.

Erwin. Fehlberg and NASA (1968). "Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control". In: NASA Technical Report. NASA, Washington, D.C.

D. C. Liu and J. Nocedal (1989). "On the limited memory BFGS method for large scale optimization". In: 149 Math. Programming, 45(3, (Ser. B)):503–528.