# Results for Learned Evolution Function on the Bloch Sphere

## I Methodology

The nonlinear network is a dense neural network, where each layer is activated by SeLUs and optimized with the Adam optimizer. There are six layers, with 64, 256, 512, 512, 256, and 64 neurons respectively. Each model is trained for 250 epochs with a learning rate of $10^{-3}$ followed by another 250 epochs with a learning rate of $10^{-4}$. The nonlinear network is put between the encoder and decoder of an autoencoder network which has already been trained on points on the Bloch sphere and is frozen.

The training is done by putting points on the Bloch sphere through the entire network and then comparing the output to the result from the Schrödinger equation. The loss used is the L2 distance between the output point and the point given by the Schrödinger equation. Initial conditions were sampled as a normal distribution and evolved on the Bloch sphere using the constant Hamiltonian

$$\mathcal{H} = \begin{pmatrix} 0.5567 + 0i & 0.9560 + 0.7846i \\ 0.9560 - 0.7846i & 0.0873 + 0i \end{pmatrix}$$

We set aside 10% of the available evolutions for validation.

## II Results

We have measured most of our results as functions of timestep, evolution time, and number of initial conditions.

### II.1 Linear Recovery

Our entire end-to-end network (encoding→ evolution→ decoding) is ideally equivalent to the Schrödinger equation, and thus should be linear. Then we should have

$$\phi^{-1} \circ F^t \circ \phi(\alpha(v + w)) = \alpha(\phi^{-1} \circ F^t \circ \phi(v) + \phi^{-1} \circ F^t \circ \phi(w)). \tag{II.1}$$

In order to test this, we select points $v$ and $w$ in our training set such that $\alpha(v + w)$ is approximately in our dataset as well (there is some $\widetilde{v+w}$ such that $\|\alpha(v + w) - \widetilde{v+w}\|_2 < \epsilon$). $\alpha$ is selected such that $v + w$ has a norm of 1. Since it is in the training set and thus a point on the Bloch sphere, $\widetilde{v+w}$ will have this $\alpha$ factor built in, so in code the $\alpha$ only appears on the right hand side of the equation.

We select the test points to be in our training set in order to ensure that any difference between the two sides of the equation is due to lost linearity, rather than poor predictions on the part of one of the models.

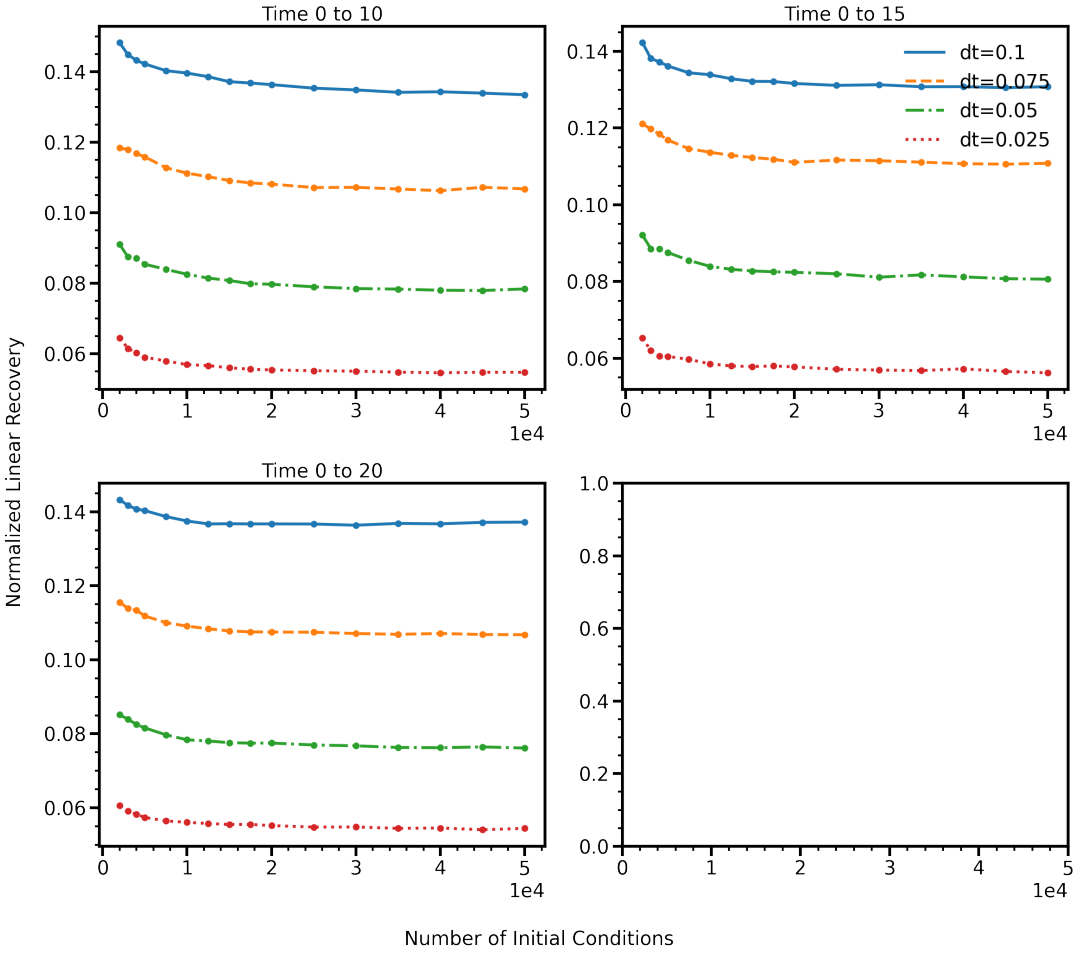


Figure 1: Linear recovery after timestep normalization.

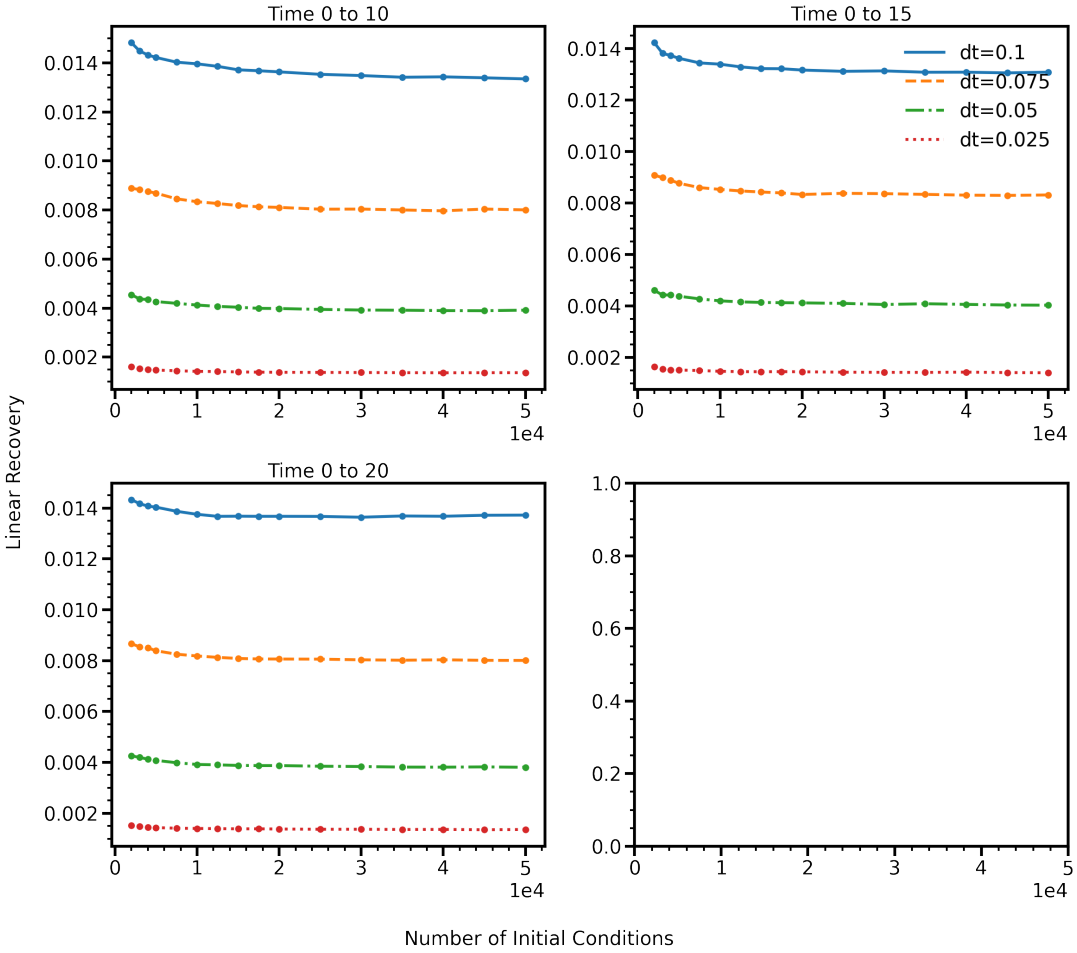


Figure 2: Linear recovery before timestep normalization.

Before normalization, linearity is recovered out to two decimal places, which is as far as the model is accurate to (see II.3, particularly figure 6). We note that the linear recovery is roughly constant across evolution time and has only a light dependence on the number of initial conditions.

Looking instead to the linear recovery plotted with a constant timestep as in figures 7 and 8, we can more easily see how the linear recovery changes with respect to the evolution time. Note that the differences in the evolution time primarily happen at the third decimal, beyond our model's maximum range of accuracy, hence some of the strange behavior.

## II.2 Nonlinearity

By our hypothesis, the learned evolution function function should be nonlinear. To measure this, we find the L2 distance between $F^t(\phi(v) + \phi(w))$ and $F^t(\phi(v)) + F^t(\phi(w))$. Unlike when checking for linearity, when checking for nonlinearity we must make sure that $\phi(v), \phi(w)$, and $\phi(v)+\phi(w)$ are all in the (compressed) dataset. This is due to checking for nonlinearity only in the compressed space which $F^t$ acts in (we already know that $\phi$ and $\phi^{-1}$ are nonlinear). The larger our nonlinearity value is, the more nonlinear our function.

We may first note that the nonlinearity values are all decidedly nonzero and reasonably larger than the linear recovery values, indicating that our model is indeed nonlinear in the compressed space.

## II.3 Accuracy

The primary accuracy metric which we have used is the validation loss normalized with the timestep. This metric can essentially be interpreted as the expected L2 distance between the predicted evolution at time $t = 1$ and the true evolution at $t = 1$. Smaller is better.

We note that upon normalization, shorter timesteps lead to a larger overall error despite the smaller local error. This implies that while the local error does decrease with the timestep as we would expect, it does not decrease as quickly as the number of timesteps required to evolve the same amount of time increases. As expected, evolving the system for longer yields better accuracy both globally and locally. We note however that this is not due simply to an increase in the number of training points; the system evolved for 10 units of time with a timestep of 0.05 has the same number of points as the system evolved for 20 units of time with a timestep of 0.1, yet the latter system preforms better.

As we would expect, the accuracy of our model increases as we give it more initial conditions to train on. We believe that there is a plateau beginning to form around 40000 initial conditions, however.

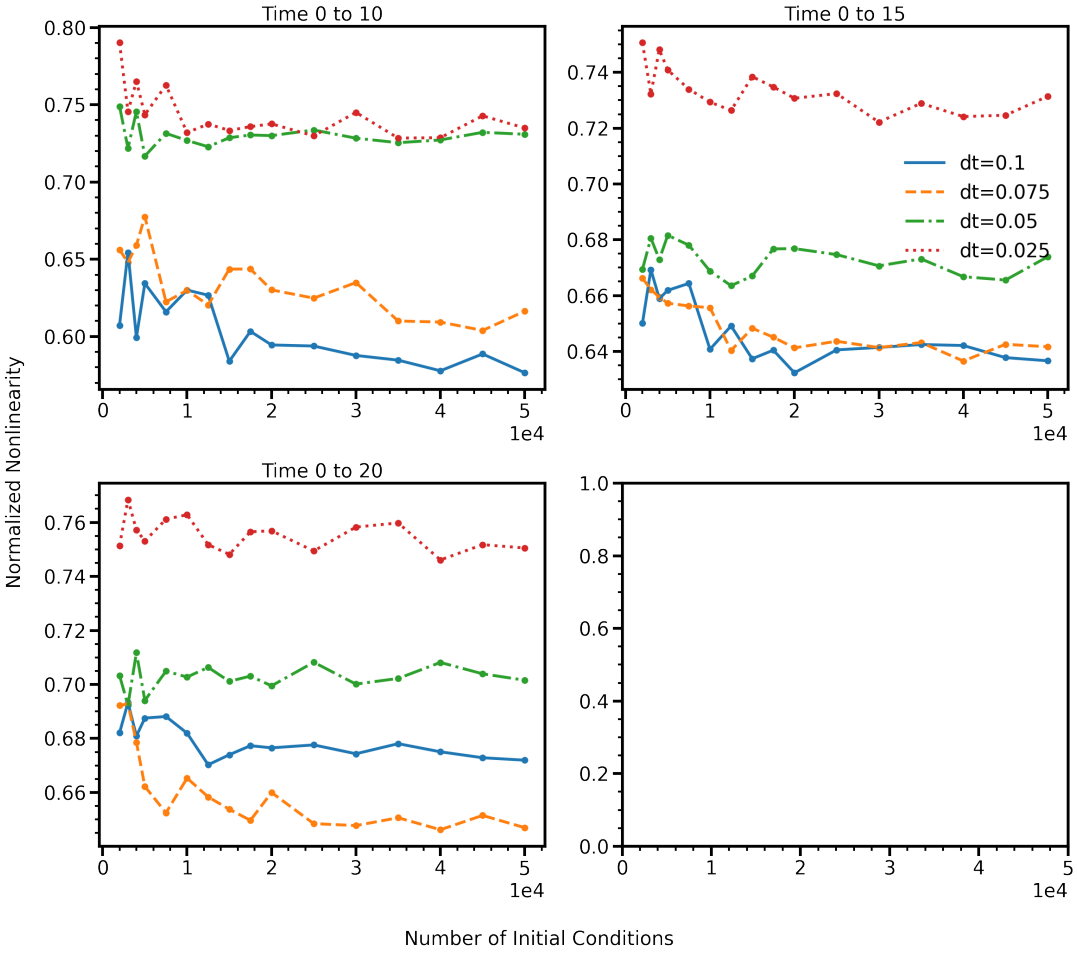


Figure 3: Nonlinearities after normalization.

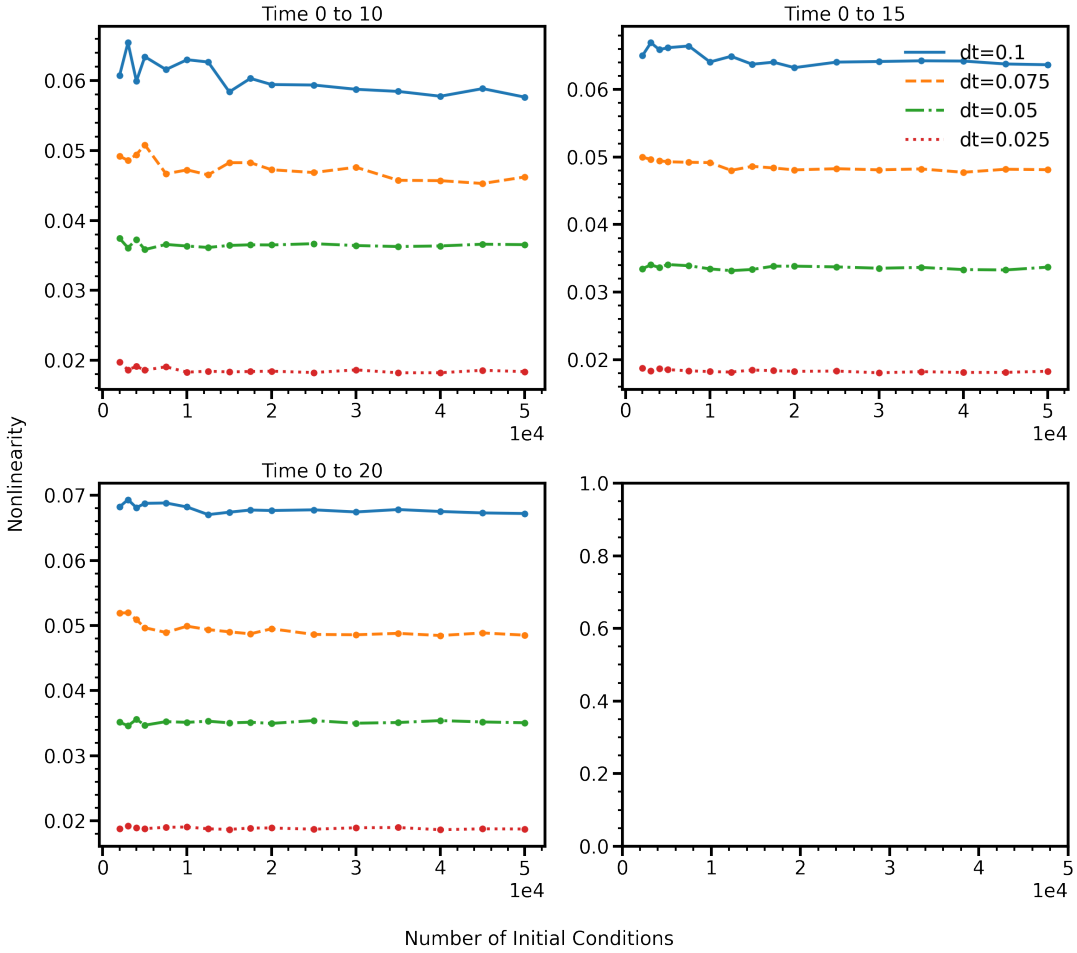


Figure 4: Nonlinearities before normalization.

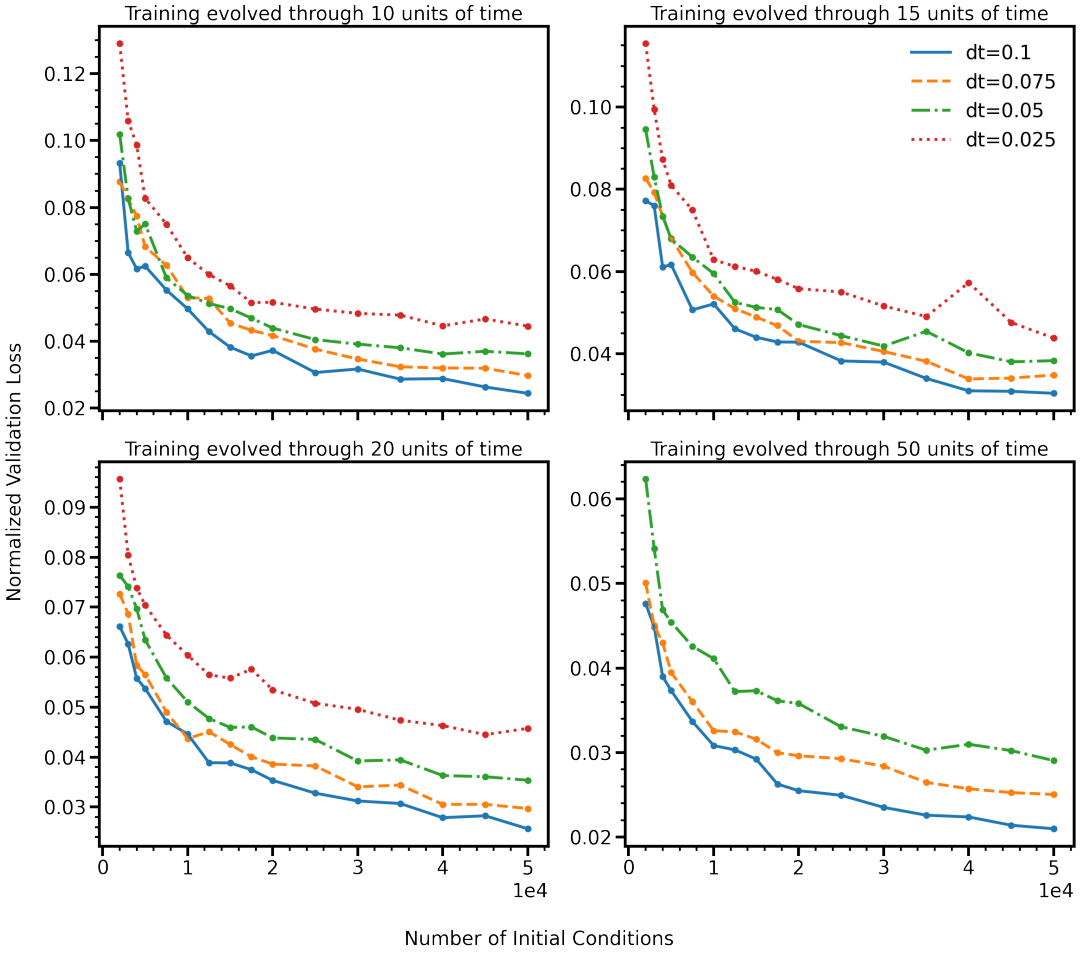


Figure 5: Plots of the normalized validation loss. Note the difference in scale of each plot.

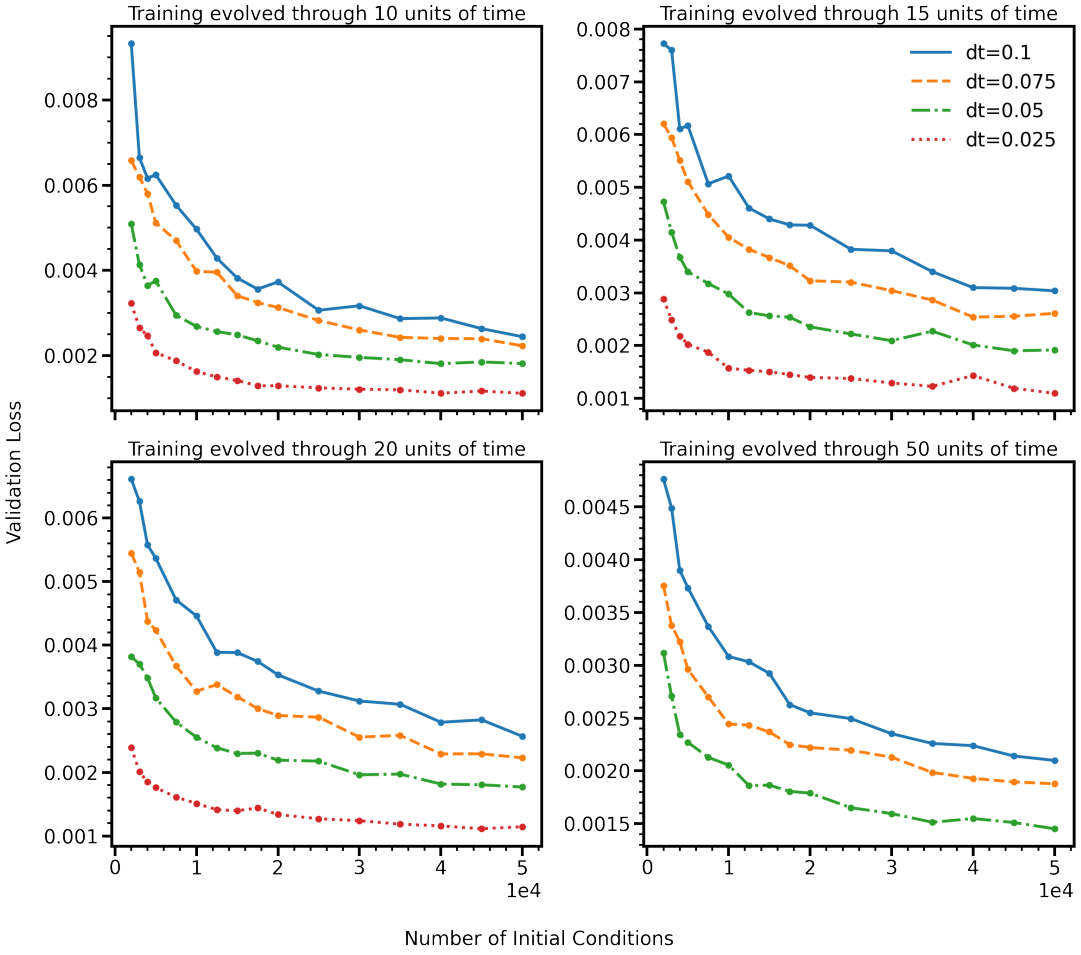


Figure 6: Validation loss before normalization. Note that all values are below $10^{-2}$, indicating that the model is accurate to two decimals step-to-step.

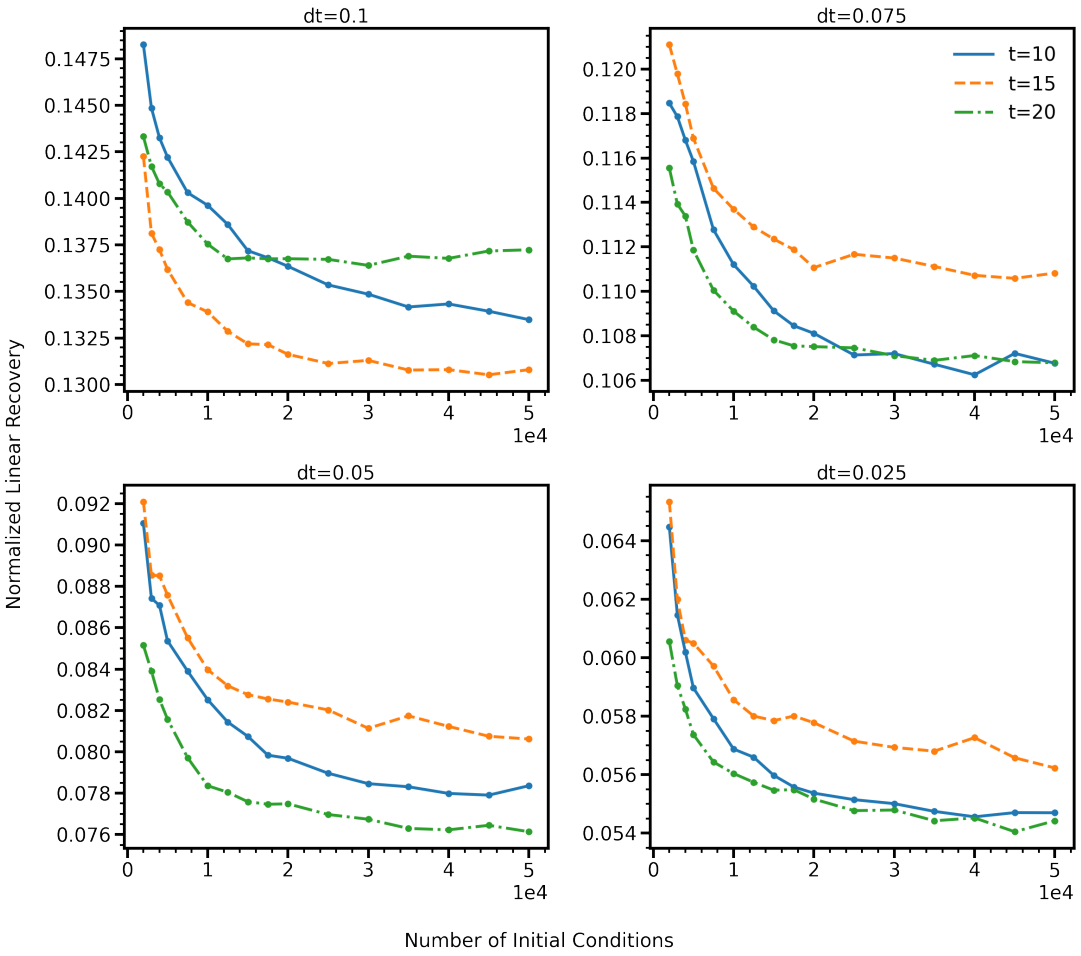


Figure 7: Normalized linear recovery, now with each plot having a constant timestep rather than a constant evolve time as in figure 1

# A Additional Plots

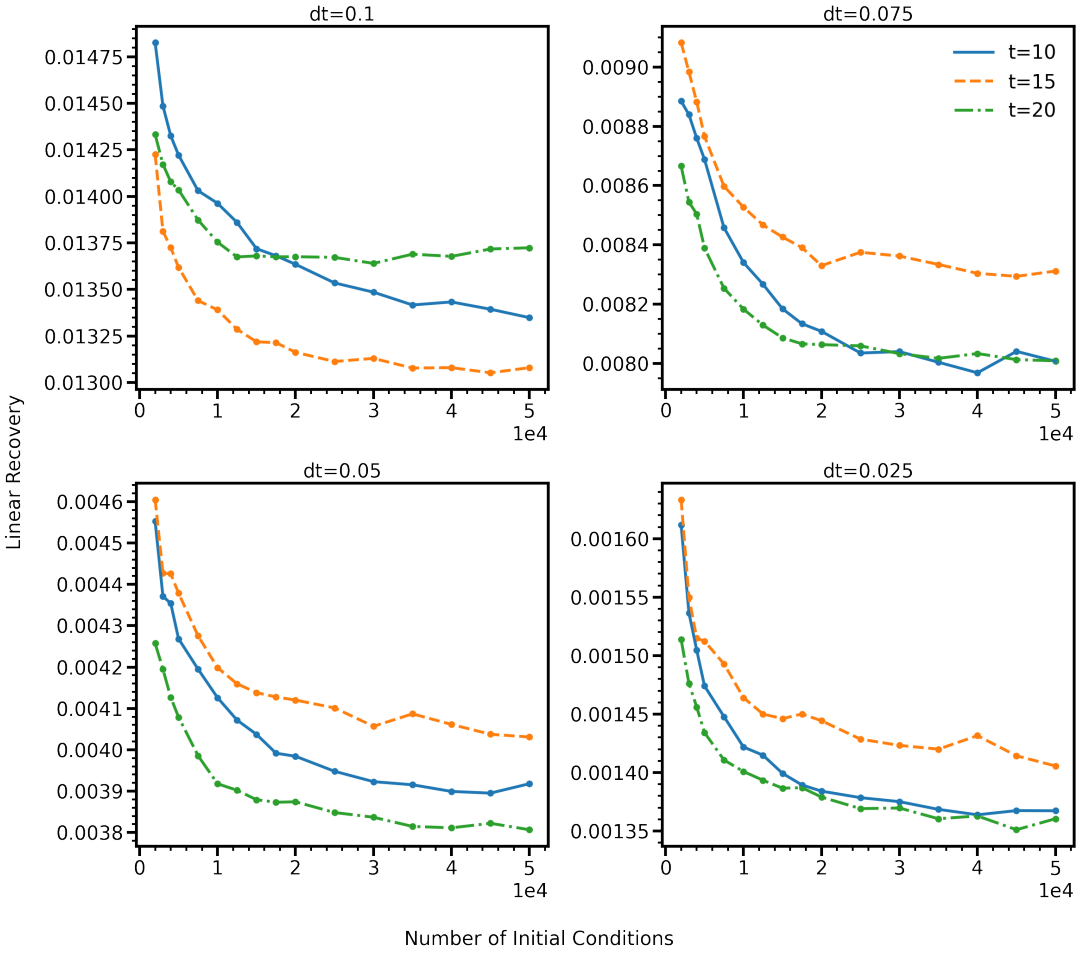


Figure 8: Non-normalized linear recovery, now with each plot having a constant timestep rather than a constant evolve time as in figure 2

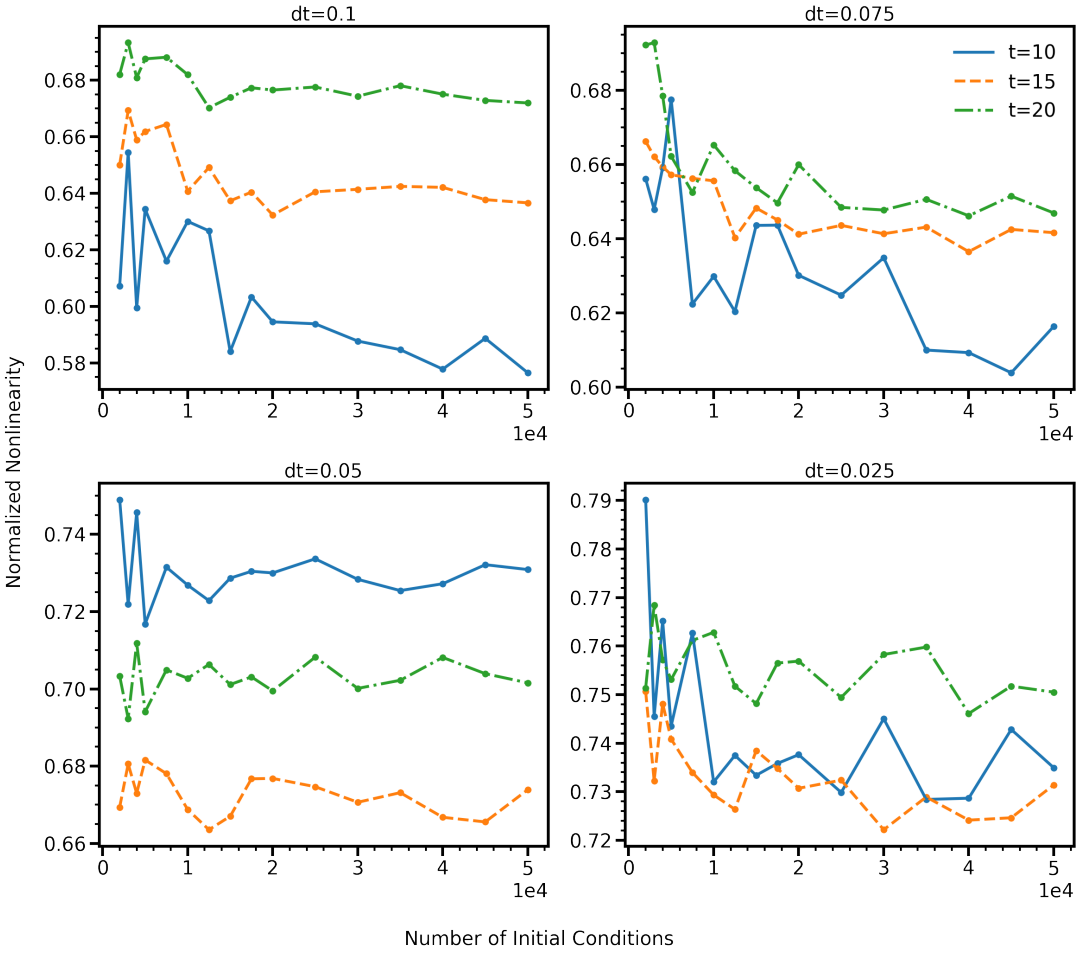


Figure 9: Normalized nonlinearity, now with each plot having a constant timestep rather than a constant evolve time as in figure 3

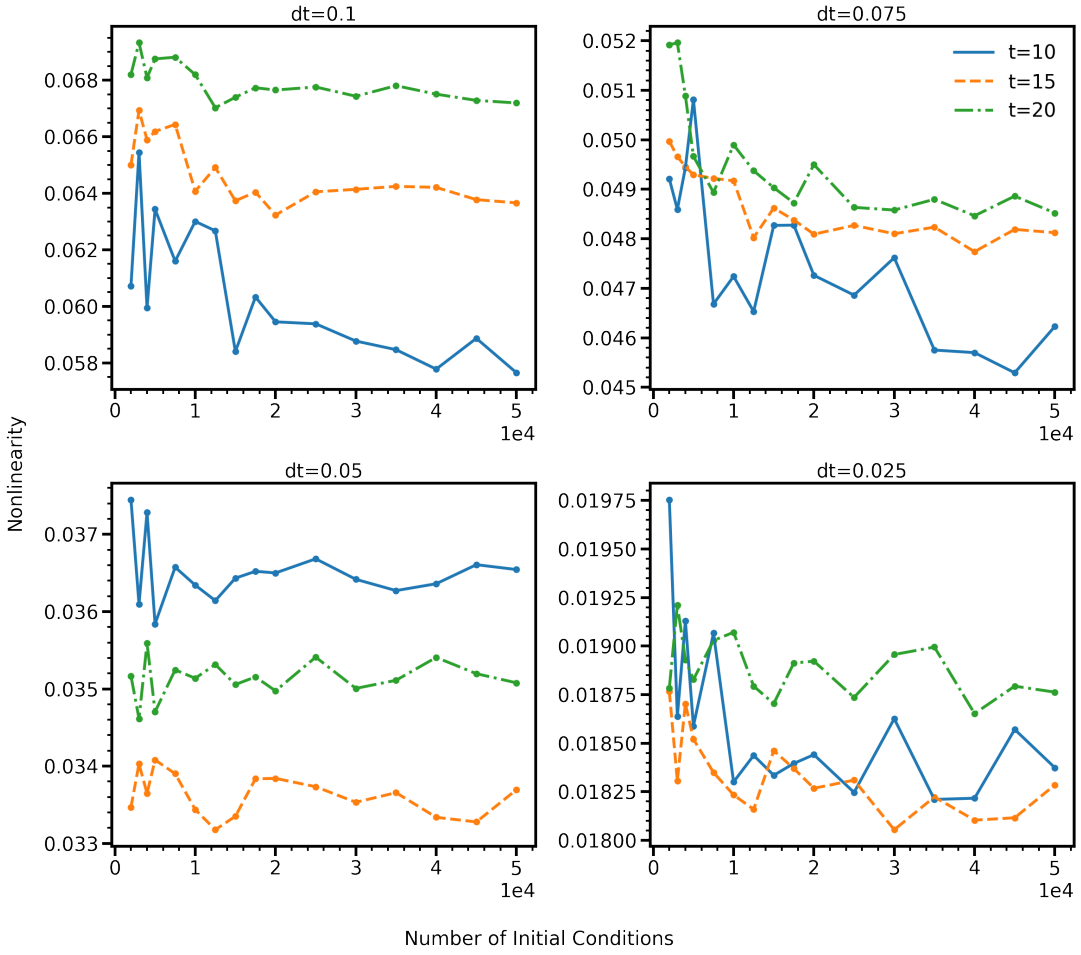


Figure 10: Normalized nonlinearity, now with each plot having a constant timestep rather than a constant evolve time as in figure **??**

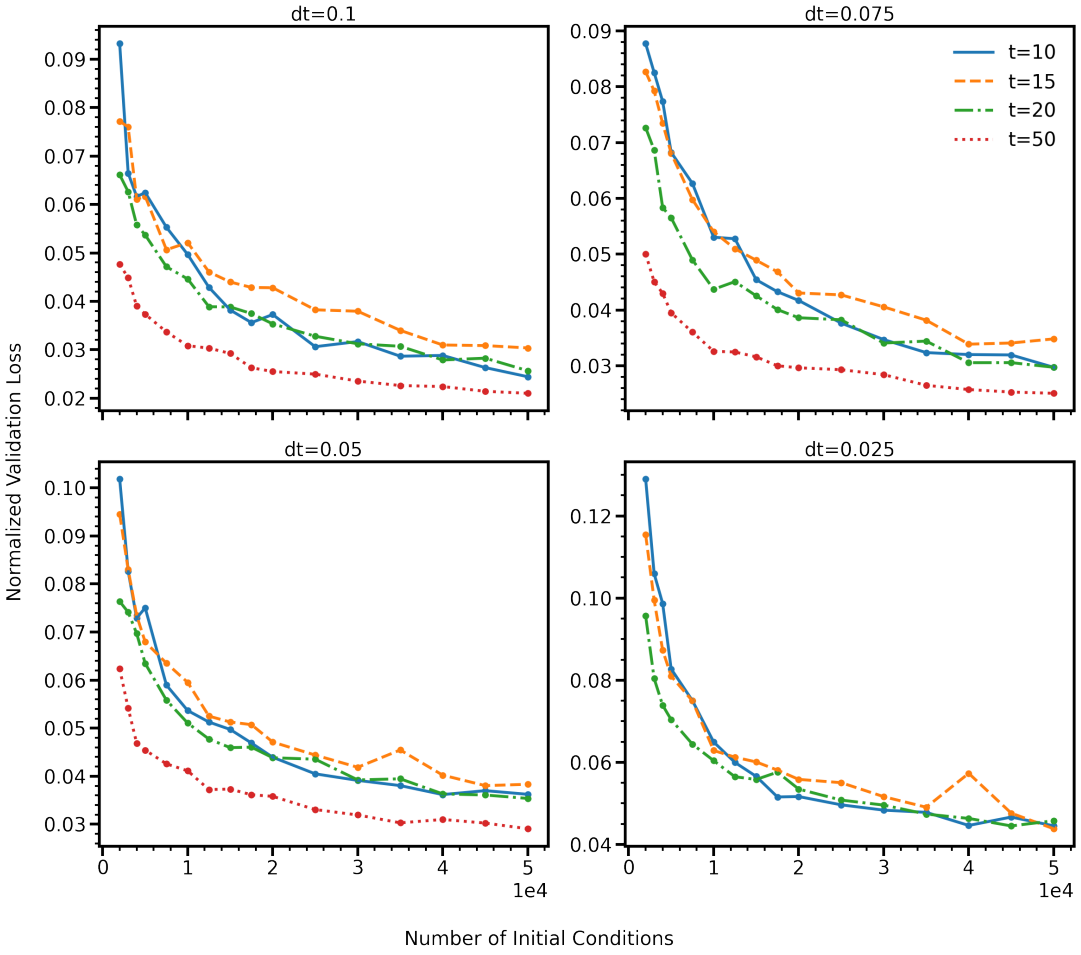


Figure 11: Normalized nonlinearity, now with each plot having a constant timestep rather than a constant evolve time as in figure 5

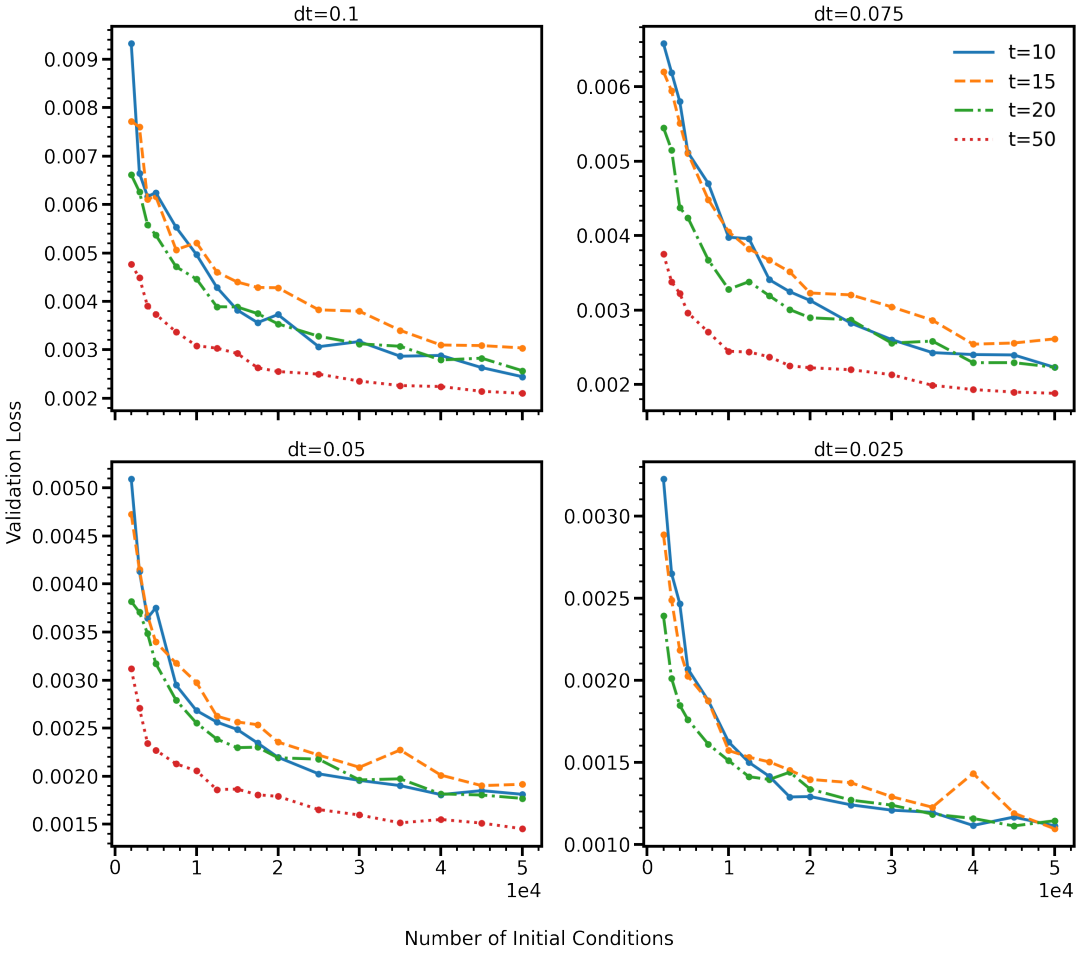


Figure 12: Normalized nonlinearity, now with each plot having a constant timestep rather than a constant evolve time as in figure 6