Supplemental Material: Solving Differential Equations with Generative Adversarial Networks

A. Hyperparameters

We performed grid searches for each problem to find the appropriate set of hyperparameters for our DEQGAN method, and then used the same parameters (where applicable) with the L_2-based method to ensure accurate comparisons. Specifically, we ensured the L_2-based neural network had the exact same hyperparameters as the generator in DEQGAN. The grid searches iterated over tens to hundreds of settings, depending on the problem[1], to find the settings used in the paper. Exact hyperparameters used for each method are detailed in Table 1.

In addition to this supplementary material, we have also provided our code. When attempting to reproduce our results, it is important to follow the README exactly, and to set the correct hyperparameters in denn/config.py. Most notable is the fact that different operating systems (OS) may produce significantly different results. The README details which OS to use for each problem.

B. Example of Non-Convergence

We mentioned, but did not have the space to elaborate on, the fact that our DEQGAN method is prone to divergences in training. In Figure 1 we present an example where we used the exact same hyperparameters as presented earlier in the paper, for the nonlinear oscillator, but with a different random seed[2].

We see that the mean squared error initially decreases, then quickly increases and flattens around 10^0 (very poor). We note that the generator's loss fluctuates wildly and diverges away from the discriminator's loss. This is what we mean by non-convergence, and is the reason that we take the top k results from n trials (to remove some of these non-converged examples). While it may not be necessary, in general, to have the losses converge to the same value, it appears that at least converging towards stable values for each loss can produce good results. In Figure 1 we see that the generator's loss is actually drifting away from the discriminator's loss at the end of training.

[1]The exponential problem required the least tuning (~ 10 settings) while the nonlinear oscillator required the most (~ 100 settings).

[2]This leads to different initialization of the weights of the two models, and produces different realizations for the grid sampling procedure.

Table 1. Hyperparameter settings for each experiment in the main paper.

HYPERPARAMETER	EXPONENTIAL	SIMPLE OSCILLATOR	NONLINEAR OSCILLATOR	2-D POISSON
NUMBER OF ITERATIONS	500	10000	50000	500
GENERATOR NODES	20	64	64	64
GENERATOR LAYERS	2	2	12	4
DISCRIMINATOR NODES	20	64	64	64
DISCRIMINATOR LAYERS	2	10	16	4
GENERATOR STEP SIZE	10^{-2}	10^{-3}	10^{-3}	10^{-3}
DISCRIMINATOR STEP SIZE	10^{-2}	10^{-3}	10^{-3}	10^{-3}
ADAM MOMENTS (β_1, β_2)	(0.9, 0.999)	(0., 0.9)	(0., 0.9)	(0., 0.9)
LEARNING RATE DECAY (γ)	1.0	0.999	0.9999	0.99
ACTIVATION FUNCTION	LEAKYRELU	TANH	TANH	TANH
GAN LOSS	CROSS-ENTROPY	WASSERSTEIN	WASSERSTEIN	WASSERSTEIN
GRADIENT PENALTY	N/A	0.1	0.1	0.1
CONDITIONAL GAN (D ONLY)	FALSE	TRUE	TRUE	TRUE

C. Examples of L_2-based Training

While our paper referenced the results of the classical L_2-based training method throughout, we did not present any figures to visualize training due to space constraints. Figures 2 and 3 present these visualizations for the simple and nonlinear oscillator problems, respectively.

For both problems, the L_2-based method does pretty well. Particularly noteworthy is the behavior we observer in Figure 3, where we see the L_2 loss (middle plot) decreasing roughly monotonically, but the mean squared error experiencing a sharp increase around step $10,000$.

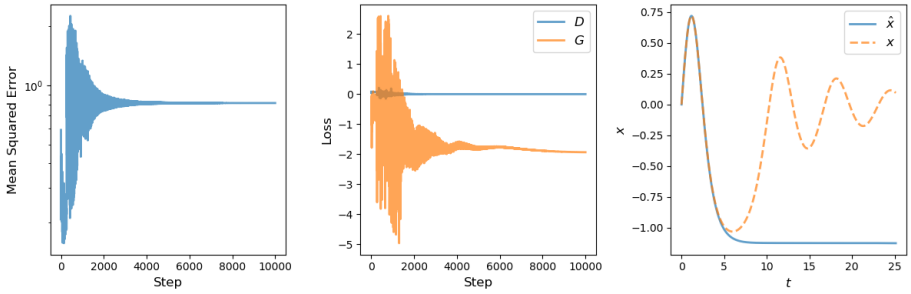

Figure 1. Non-convergence of DEQGAN training for the non-linear oscillator problem. The left-most figure plots the mean squared error vs. step (iteration) count. To the right of this, we plot the value of the generator (G) and discriminator (D) losses for each step. The right-most figure plots the prediction of the generator $\hat{x}$ and the ground-truth solution x as functions of time t.

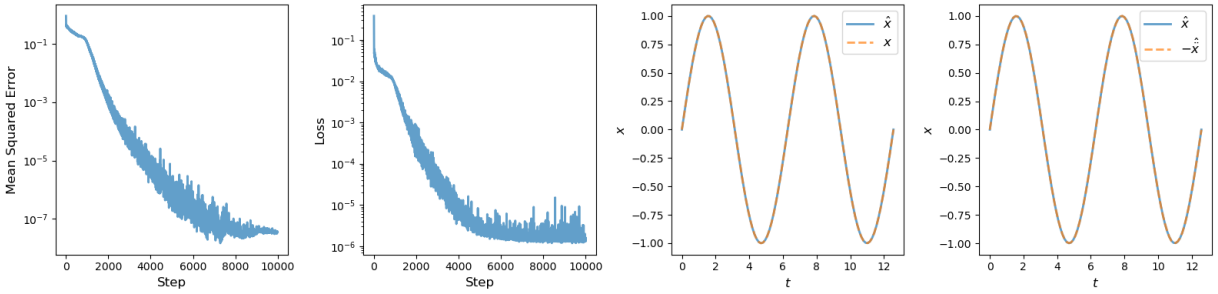

Figure 2. Visualization of L_2-based training for the simple oscillator problem. The left-most figure plots the mean squared error vs. step (iteration) count. To the right of this, we plot the value of the L_2 loss. Next to this we plot the prediction of the model $\hat{x}$ and the ground-truth solution x as functions of time t. The far-right plot shows the value of the second derivative of the solution $\hat{\ddot{x}}$ which indicates the solution properly obeys the equation.

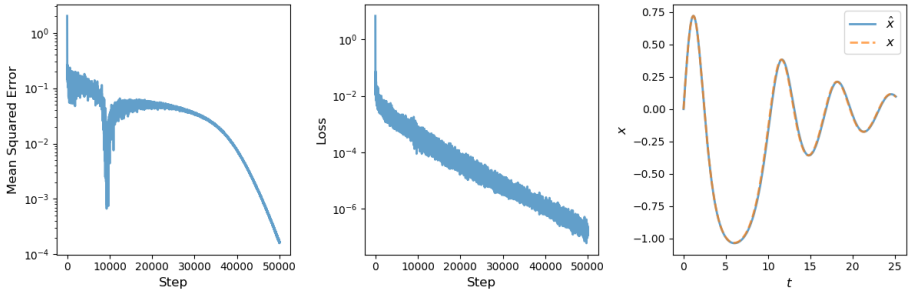

Figure 3. Visualization of L_2-based training for the nonlinear oscillator problem. The left-most figure plots the mean squared error vs. step (iteration) count. To the right of this, we plot the value of the L_2 loss. The far-right plot shows the prediction of the model $\hat{x}$ and the ground-truth solution x as functions of time t.