Project Report for 10-707: Advanced Deep Learning "Deep Equilibrium Models for Safe Control"

Ritabrata Ray, Pharuj Rajborirug
Department of Electrical & Computer Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213
`ritabrar, prajbori @andrew.cmu.edu`

Abstract

In this project, we study Deep Equilibrium Models for risk estimation in safe control. In particular, we use weight tied deep equilibrium models to learn to predict the long term safety probability of a stochastic dynamical system with known dynamics. Even with known dynamics, this probability is computationally intractable to compute exactly but it is known to satify a convection-diffusion PDE. We use physics informed neural networks to approximate a solution to this PDE. To this end, we use a deep implicit layer to model the solution instead of a vanilla fully connected network as typically used in PINN literature and numerically show better risk estimation using almost half the number of trainable parameters.

1 Introduction to Deep Equilibrium Models

The notion of a layer is central to any deep-learning architecture. Deep Equilibrium Models are a novel architecture introduced by Bai et al. [2019]. Deep Equilibrium Models are based on the concept of deep implicit layers which emulate neural networks of infinite depth by implicitly defining its feed-forward output as the fixed point of a suitable function. Specifically, consider a deep feedforward network with L layers:

$$z^{[i+1]} = f_\theta^{[i]}(z^{[i]}; x) \quad \text{for } i = 0, \dots, L-1, \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the input, $z^{[i]} \in \mathbb{R}^{n_z}$ is the i^{th} layer's hidden state with $z^{[0]} = \mathbf{0}$, and $f_\theta^{[i]} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \mapsto \mathbb{R}^{n_z}$ is the feature map transformation of the i^{th} layer, parameterized by θ. Now if the model is *weight-tied* i.e., $f_\theta^{[i]} = f_\theta,\ \forall i$, and $\lim_{i\to\infty} f_\theta(z^{[i]}; x)$ exists and its value is z^*. Further, if the above z^* satisfies the fixed point equation $f_\theta(z^*; x) = z^*$, then the equilibrium modles can be interpreted as the infinite-depth limit of the above weight-tied network such that $f_\theta^\infty(z^*; x) = z^*$.[1] If the fixed point uniquely exists, then one common way to solve it is through fixed point iterations i.e., repeatedly apply the update: $z^{[t+1]} = f_\theta(z^{[t]}; x)$ until convergence. Other sophisticated method like Newton's method (to solve for the root z^* of $g(z^*; x) := z^* - f_\theta(z^*; x)$), or Anderson acceleration Anderson [1965] can be used to implement the forward function of such deep implicit models. However, naive use of automatic differentiation packages for computing the backpropagation could be computationally expensive (almost intractable for all practically useful models). This is because the iterative nature of the forward pass builds an enormously large computation graph and the use of second order methods like Newton, Anderson to reduce the iterations increases the complexity of the computation graph as well. However, deep equilibrium models Bai et al. [2019] avoid this problem by using the implicit function theorem to directly differentiate

[1] By Banach's fixed point theorem, the fixed point uniquely exists if f_θ is contractive over its input domain. Bai et al. [2019] states that f_θ needs to be stable and constrained.

through the fixed point z^* at equilibrium, thus requiring constant memory to backpropagate through an infinite-depth network. This also decouples the forward and backward pass through the model and several advanced iterative procedures can be flexibly used to compute the forward pass without having to change the backward pass method. In particular, the following equation is used to compute the backward pass gradient of the fixed-point solution:

$$\frac{\partial z^*}{\partial \theta} = \left(I - \frac{\partial f_\theta(z^*; x)}{\partial z^*}\right)^{-1} \frac{f_\theta(z^*; x)}{\partial \theta}. \tag{2}$$

This can be integrated into a standard automatic differentiation library based code by a couple of extra lines. We can take the forward pass outside the auto-differentiation tape for computing the fixed point root z^* of $g(z^*; x) = 0$ using our choice of the iterative solver. After this, we can reengage the automatic differentiation tape by computing the update:

$$z = z^* - g(z^*; x),$$

within the forward pass. At the root z^*, this has no effect on the output of the forward pass but it reinserts the gradient $-\frac{\partial g(z^*;x)}{\partial \theta} = \frac{\partial f_\theta(z^*;x)}{\partial \theta}$ into the autograd tape, when z is used as the final forward pass output. So, the backpropagated gradient received is $\nabla_z l(z^*)^\intercal \frac{\partial f_\theta(z^*;x)}{\partial \theta}$. The only extra line (the last line of forward pass) uses the "register hook" function of the auto-differentiation package and goes into obtaining the desired gradient $\nabla_\theta l = \nabla_z l(z^*)^\intercal \frac{\partial z^*}{\partial \theta}$. Thus, we have:

$$(\nabla_\theta l)^\intercal = \left(\frac{\partial f_\theta(z^*; x)}{\partial \theta}\right)^\intercal \left(I - \frac{\partial f_\theta(z^*; x)}{\partial z^*}\right)^{-\intercal} \nabla_z l(z^*).$$

Now we can reuse the Jacobian $J = \frac{\partial f_\theta(z^*;x)}{\partial \theta}$ computed at z^* using solvers like Newton's method and use the register hook to solve linear equations with matrix $(I - J)^\intercal$ and the incoming backpropagated gradient, so we have the the new gradient $(I - J)^{-\intercal} \nabla_z l(z^*)^\intercal$. This pre-multiplied by the matrix $\frac{\partial f_\theta(z^*;x)}{\partial \theta}^\intercal$ (the effect of reengaging the autodiff tape) yields the desired gradient with no extra computation other than the equation solver at the hook. [2]

Deep Equilibrium Models have performed significantly well towards solving ordinary differential equations. Recently, Marwah et al. [2023] showed the application of these models towards solving partial differential equations and their performance compares against the erstwhile state-of-the-art Fourier Neural Operators Li et al. [2021] for solving steady state PDEs.

2 PDE for long term safety

We consider a stochastic dynamical system with the dynamics given by the SDE below:

$$dX_t = f\left(X_t, U_t\right) dt + \sigma\left(X_t\right) dW_t, \tag{3}$$

where $X_t \in \mathbb{R}^n$ is the state variable at time $t \in \mathbb{R}_+$, $U_t \in \mathbb{R}^m$ is the control-variable at time t, W_t is the k-dimensional Brownian motion starting at $W_0 = 0$, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the deterministic part of the system dynamics, and $\sigma : \mathbb{R}^n \to \mathbb{R}^{n \times k}$ determines the magnitude of noise in the dynamics stemming from the Brownian motion disturbance W_t, which represents the uncertainty due to unmodeled dynamics, and prediction error of environment variables. $f, \sigma, (U_t)_{t \in \mathbb{R}_+}$ are chosen in such a way that (3) admits a unique strong solution[3].

Definition 1 (Safe-set and barrier function)**.** *The safe region is defined as the set $\mathcal{S}$ which is characterized by a super-level set of some function $\phi(x)$, i.e.*

$$\mathcal{S} = \left\{x \in \mathbb{R}^n | \phi(x) \geq 0\right\}. \tag{4}$$

We call $\phi(x)$ the barrier function and $\mathcal{S}$ the safe set. Additionally, we denote the boundary of the safe set and the unsafe set respectively as follows:

$$\partial\mathcal{S} = \left\{x \in \mathbb{R}^n | \phi(x) = 0\right\}, \tag{5}$$

$$\mathcal{S}^c = \left\{x \in \mathbb{R}^n | \phi(x) < 0\right\}. \tag{6}$$

[2]See the Chapter 1 of Deep Implicit Layers tutorial: `https://implicit-layers-tutorial.org/`.

[3]See Chapter 5 of Moustris et al. [2011], Chapter II.7 of Borodin [2017] and Chapter 1 of Øksendal [2014].

We assume that the barrier function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a second-order differentiable function whose gradient does not vanish at $\partial\mathcal{S}$.

Definition 2 (Forward invariant set)**.** *A set M is said to be a forward invariant set with respect to the dynamics given by* (3) *if*

$$X_0 \in M \implies X_t \in M, \ \forall t \in \mathbb{R}_+. \tag{7}$$

We further define the following two functions to characterize forward invariance:

$$\Phi_x(T) := \inf \left\{ \phi(X_t) \in \mathbb{R} | t \in [0,T], X_0 = x \right\}, \tag{8}$$

$$\Gamma_x(l) := \inf \left\{ t \in \mathbb{R}_+ | \phi(X_t) < l, X_0 = x \right\}, \tag{9}$$

where $x \in \mathcal{S}$.

Definition 3 (Forward convergent set)**.** *A set M is said to be a forward convergent set with respect to the dynamics given by* (3) *if*

$$X_0 \notin M \implies \exists t \in \mathbb{R}_+ \text{ such that } X_t \in M. \tag{10}$$

We further define the following two functions to characterize forward convergence:

$$\Theta_x(T) := \sup \left\{ \phi(X_t) \in \mathbb{R} | t \in [0,T], X_0 = x \right\}, \tag{11}$$

$$\Psi_x(l) := \inf \left\{ t \in \mathbb{R}_+ | \phi(X_t) \geq l, X_0 = x \right\}, \tag{12}$$

where $x \in \mathcal{S}$.

The probability of staying within the safe set $\mathcal{S}$ during a time interval $[0, T]$ given $X_0 = x \in \mathcal{S}$ can be computed using the distributions of $\Phi_x(T)$ or $\Gamma_x(l)$ as:

$$\mathbf{P}_x(X_t \in \mathcal{S}, \ \forall t \in [0,T]) = \mathbf{P}(\Phi_x(T) \geq 0) \tag{13}$$

$$= 1 - \mathbf{P}(\Gamma_x(0) \leq T). \tag{14}$$

Similarly, the probability of having re-entered the safe set $\mathcal{S}$ by time T given $X_0 = x \notin \mathcal{S}$, can be computed using the distributions of $\Theta_x(T)$ and $\Psi_x(l)$ as:

$$\mathbf{P}_x(\exists t \in [0,T] \text{ such that } X_t \in \mathcal{S}) = 1 - \mathbf{P}(\Theta_x(T) \leq 0) \tag{15}$$

$$= \mathbf{P}(\Psi_x(0) \leq T) \tag{16}$$

2.1 Differential equation for long term safe probability

We restate Theorem 1 of Chern et al. [2021] stating the differential equation that the long term safe probability satisfies:

Theorem 2.1 (Theorem 1 of Chern et al. [2021])**.** *Let $z = \left[\phi(x), x^\intercal\right]^\intercal \in \mathbb{R}^{n+1}$, and let the system in* (3) *be initialized at state $X_0 = x$. Define the complimentary cumulative distribution function (Co-CDF) of $\Phi_x(T)$ as,*

$$F(z,T;l) := P(\Phi_x(T) \geq l), l \in \mathbb{R}. \tag{17}$$

Then, the augmented state has the closed loop dynamics:

$$dZ_t = \rho\left(Z_t\right) dt + \zeta\left(Z_t\right) dW_t, \tag{18}$$

with drift and diffusion parameters[4]*:*

$$\rho\left(z\right) := \begin{bmatrix} \mathcal{L}_{f(x,u)}\phi + \frac{1}{2}Tr\left[\sigma(x)\sigma(x)^\intercal \nabla^2\phi\right] \\ f(x,u) \end{bmatrix}, \tag{19}$$

$$\zeta\left(z\right) := \begin{bmatrix} \mathcal{L}_\sigma(x)\phi \\ \sigma(x) \end{bmatrix} \tag{20}$$

[4] $\mathcal{L}_v\phi := \nabla\phi^\intercal v$ denotes the Lie derivative of ϕ along the vector field v, and $\mathcal{L}_M\phi$ is the Lie derivative of ϕ along the matrix field M such that for any vector u, $\left(\mathcal{L}_M\phi\right)u := (\nabla\phi)^\intercal M u$.

respectively, and the long term safety probability $F(z, T; l)$ is the solution to the initial-boundary value problem of the convection-diffusion equation on the super-level set $\{z \in \mathbb{R}^{n+1} : z[1] \geq l\}$,

$$\frac{\partial F}{\partial T} = \frac{1}{2}\nabla \cdot (D\nabla F) + \mathcal{L}_{p-\frac{1}{2}\nabla\cdot D}F, \quad z[1] \geq l, T > 0, \tag{21}$$

$$F(z, T; l) = 0, \quad z[1] < l, T > 0, \tag{22}$$

$$F(z, 0; l) = \mathbb{1}_{\{z[1]\geq l\}}(z), \quad z \in \mathbb{R}^{n+1}, \tag{23}$$

where $D = \zeta\zeta^\intercal$.

Observe that $F(z, T; 0)$ is the probability of the system staying through time T starting at a safe augmented state z.

Similarly, we have the following PDE for the long term recovery probability:

Theorem 2.2 (Theorem 4 of Chern et al. [2021])**.** *Let $z = \left[\phi(x), x^\intercal\right]^\intercal \in \mathbb{R}^{n+1}$, and let the system in* (3) *be initialized at state $X_0 = x$. Define the cumulative distribution function (CDF) of $\Psi_x(T)$ as,*

$$F(z, T; l) := P(\Psi_x(T) \leq l), l \in \mathbb{R}. \tag{24}$$

Then, the augmented state has the closed loop dynamics:

$$dZ_t = \rho\left(Z_t\right)dt + \zeta\left(Z_t\right)dW_t, \tag{25}$$

with drift and diffusion parameters[5]*:*

$$\rho\left(z\right) := \begin{bmatrix} \mathcal{L}_{f(x,u)}\phi + \frac{1}{2}Tr\left[\sigma(x)\sigma(x)^\intercal\nabla^2\phi\right] \\ f(x, u) \end{bmatrix}, \tag{26}$$

$$\zeta\left(z\right) := \begin{bmatrix} \mathcal{L}_\sigma(x)\phi \\ \sigma(x) \end{bmatrix} \tag{27}$$

respectively, and the long term safety probability $F(z, T; l)$ is the solution to the initial-boundary value problem of the convection-diffusion equation on the super-level set $\{z \in \mathbb{R}^{n+1} : z[1] \geq l\}$,

$$\frac{\partial F}{\partial T} = \frac{1}{2}\nabla \cdot (D\nabla F) + \mathcal{L}_{p-\frac{1}{2}\nabla\cdot D}F, \quad z[1] < l,\ T > 0, \tag{28}$$

$$F(z, T; l) = 1, \quad z[1] \geq l, T > 0, \tag{29}$$

$$F(z, 0; l) = \mathbb{1}_{\{z[1]\geq l\}}(z), \quad z \in \mathbb{R}^{n+1}, \tag{30}$$

where $D = \zeta\zeta^\intercal$.

Observe that $F(z, T; 0)$ is the probability of safe recovery within time T starting at augmented state z.

In this project we aim to learn the risk probability $F(z, T; l)$ for a general dynamical system with $f(X_t)$ in place of $f(X_t, U_t)$ using a data driven approach. Solving the above PDEs for general dynamics is intractable even for cases of known dynamics (known f and σ). Directly learning the safe probability by fitting a neural network on empirically obtained monte-carlo data could be very expensive since we will need a lot of samples and and collecting such samples would mean exaposing the dynamical system to unsafe states during our monte carlo data generation. Therefore, we aim to reduce the sample complexity requirement by using physics informed neural networks where we use our knowledge of the above PDEs.

3 Physics Informed Neural Networks

Let $D \subset \mathbb{R}^{d+1}$ (d dimensional space plus time variable) be a bounded domain with $\bar{D}$ being its closure, and Σ be its boundary. Let $L = -\frac{\partial}{\partial T} + \sum_{i,j} a_{ij}\frac{\partial^2}{\partial x_i \partial x_j} + \sum_i b_i \frac{\partial}{\partial x_i} + c$, be a parabolic operator and let A be the matrix with $[A]_{ij} = a_{ij}$ and let $\lambda = \lambda_{min}(A) > 0$. Further let $C^i(E)$

[5] $\mathcal{L}_v\phi := \nabla\phi^\intercal v$ denotes the Lie derivative of ϕ along the vector field v, and $\mathcal{L}_M\phi$ is the Lie derivative of ϕ along the matrix field M such that for any vector u, $\left(\mathcal{L}_M\phi\right)u := (\nabla\phi)^\intercal Mu$.

denote the class of functions which have continuous derivatives upto the i-th order in the domain E. Further for any solution u of the PDE, let $\tilde{u}$ denote the boundary values of u on Σ.

Let $D \subset \mathbb{R}^{d+1}$ be a domain. We define the *regualrity* of D as:

$$R_D := \inf_{(x,T)\in D, r>0} \frac{|B(x,T,r) \cap D|}{\min\{|D|, \frac{\pi^{(d+1)/2} r^{d+1}}{\Gamma((d+1)/2+1)}\}}, \tag{31}$$

where $B(x,T,r) := \{(y \in \mathbb{R}^{d+1}) : \|y - (x,T)\| \le r\}$, denotes the ball of radius r centered at (x,T).

Now a direct extension of Theorem 6 in Wang and Nakahira [2024] gives us the following guarantee:

Theorem 3.1. *Suppose $D \subset \mathbb{R}^{d+1}$ is a bounded domain with regularity $R_D > 0$, and $u \in C^0(\bar{D}) \cap C^2(D)$ be the solution to the parabolic PDE $Lu = 0$, with $\tilde{u}(x,T),\ (x,T) \in \Sigma$ being the boundary condition on the boundary set Σ of D which has regularity $R_\Sigma > 0$ and the parabolic operator L having $c \le 0$. Let F_θ denote a physics informed neural network parameterized by the network weights θ. Let $m_D = \frac{|D|}{(\epsilon/\ell)^{d+1}}$, and let $m_\Sigma = \frac{|\Sigma|}{(\epsilon/\ell)^{d}}$. If the points $T_D = \{(\mathbf{x}_i, T_i) : i = 1, \ldots, m_D\}$ from D, and points $T_B = \{(\mathbf{y}_i, T_i) : i = 1, \ldots, m_\Sigma\}$ from Σ are sampled from a uniform discretization of the domain and if the following holds:*

1. $\frac{1}{m_\Sigma}\sum_{(\mathbf{y}_i,T_i)\in T_B} \left|F_\theta(\mathbf{y_i}, T_i) - \tilde{u}(\mathbf{y}_i, T_i)\right| < \epsilon + \delta_1$,
2. $\frac{1}{m_D}\sum_{(\mathbf{x}_i,T_D)\in T_D} \left|LF_\theta(\mathbf{x_i}, T_i) - u(\mathbf{x}_i, T_i)\right| < \epsilon + \delta_2$, *where $\mathbf{X}$ is uniformly sampled from D,*
3. *$F_\theta, W_{F_\theta}, u$ are $\ell/2$-Lipschitz on D,*

then the error of F_θ over D is bounded by:

$$\sup_{(x,T)\in D} \left|F_\theta(x,T) - u(x,T)\right| \le \tilde{\delta_1} + \frac{\tilde{\delta_2}}{\lambda}, \tag{32}$$

where C is a constant depending on the diameters of D, Σ, and the operator L(effectively the physics error W), and where

$$\tilde{\delta_1} = \max\left\{\frac{2\delta_1}{R_\Sigma}, 2\ell \cdot \left(\frac{\delta_1|\Sigma| \cdot \Gamma(d/2+1)}{\ell R_\Sigma \pi^{d/2}}\right)^{\frac{1}{d+1}}\right\}, \tag{33}$$

$$\tilde{\delta_2} = \max\left\{\frac{2\delta_2}{R_D}, 2\ell \cdot \left(\frac{\delta_2|D| \cdot \Gamma((d+1)/2+1)}{\ell R_D \pi^{(d+1)/2}}\right)^{\frac{1}{d+2}}\right\}. \tag{34}$$

Since $c = 0$ in our convection-diffusion equation for the risk probability, the above theorem applies.

4 Numerical Experiments

In this section, we describe our numerical experiments towards estimating the risk probability for new (x,T) pairs given access to limited sample data. We use the above physics informed neural networks to minimize the loss:

$$\mathcal{L}(\theta) = \omega_B \mathcal{L}_B(\theta) + \omega_P \mathcal{L}_P(\theta), \tag{35}$$

where the boundary data loss is $\mathcal{L}_B(\theta) = \mathbb{E}_\Sigma\left[\frac{1}{2}\left\|F_\theta(\mathbf{y},T) - \tilde{u}(\mathbf{y},T)\right\|_2^2\right]$ and the physics loss is $\mathcal{L}_P(\theta) = \mathbb{E}_D\left[\frac{1}{2}\left\|LF_\theta(\mathbf{x},T) - u(\mathbf{x},T)\right\|_2^2\right]$, where L is the parabolic operator (PDE: $Lu = 0$) F_θ is the neural network model for computing the risk with model parameters θ and the expectation denotes uniform sampling over the corresponding domains. In practice, we replace the expectation by the empirical averages:

$$\hat{\mathcal{L}}_B(\theta) = \frac{1}{2m_\Sigma} \sum_{(\mathbf{y}_i, T_i) \in T_B} \left\| F_\theta(\mathbf{y}_i, T_i) - \tilde{u}(\mathbf{y}_i, T_i) \right\|_2^2,$$

and

$$\hat{\mathcal{L}}_P(\theta) = \frac{1}{2m_D} \sum_{(\mathbf{x}_i, T_i) \in T_D} \left\| L F_\theta(\mathbf{x}_i, T_i) \right\|_2^2$$

and minimize the empirical loss $\hat{\mathcal{L}}(\theta) = \omega_1 \hat{\mathcal{L}}_B(\theta) + \omega_2 \hat{\mathcal{L}}_P(\theta)$ using gradient descent.

4.1 Simulation details

We consider the one dimensional system:

$$dX_t = 0.7dt + dW_t, \tag{36}$$

where W_t is the standard Wiener process, and we have $f(X_t) = 0.7$, $\sigma(X_t) = 1$. The convection diffusion equation for the recovery probability $F(x, T)$ becomes:

$$\frac{\partial F_\theta(x,T)}{\partial T} - 0.7 \frac{\partial F_\theta(x,T)}{\partial x} - \frac{1}{2} \frac{\partial^2 F_\theta(x,T)}{\partial x^2} = 0 \tag{37}$$

$$F(x, 0) = \mathbf{1}\{x \in \mathcal{S}\} \tag{38}$$

$$F(x, T) = 1, \; \forall x \in \partial\mathcal{S}. \tag{39}$$

Here the safe set is $\mathcal{S} = \{x : x \geq 2\} = [2, \infty)$ with a corresponding barrier function $\phi(x) = x - 2$. We consider the bounded domain $\Omega = [-10, 2]$ where $x \in \Omega$ and $\tau = [0, 10]$. So, our joint domain $D = \Omega \times \tau$ and the boundary $\Sigma = \bar{D} = \alpha \cup \beta \cup \gamma \cup \delta$, where $\alpha = \big\{(x, 0) : x \in [-10, 2]\big\}, \beta = \big\{(2, T) : T \in [0, 10]\big\}, \gamma = \big\{(x, 10) : x \in [-10, 2]\big\}, \delta = \big\{(-10, T) : T \in [0, 10]\big\}.$

We generate the monte carlo boundary data T_B by uniformly sampling the initial state x_0 on the perimeter of the above rectangle $\bar{D}$ with measure equal to its perimeter and then running the dynamics (36). The labels $\tilde{u}$ are given by the boundary conditions (38) and (39). And we separately generate the interior data T_D by uniformly sampling x_0 from within the rectangular region D and running the dynamics (36) on them. For each such pair (x, T) we sample 1000 trajectories and compute the ratio of number of trajectories which enter $\mathcal{S}$ at any time within T to the number of total trajectories: 1000 to obtain an estimate $\tilde{u}(x, T)$. These estimates $\tilde{u}(x, T)$ are only observed for the test data, since the training data only needs labels on the boundary points T_B which are anyway amalytically predicted and the points in T_D do not require any label. We sample 10000 points for estiamting each of the physics and boundary loss terms i.e., $|T_B| = |T_D| = 10000$. We set $\omega_P = \omega_B = 1$ and use the Adam optimizer Kingma and Ba [2017] for gradient descent.
In the above setting we consider two cases when the neural network model for $F_\theta(x, T)$ is a fully connected network with four hidden layers ($2 \times 50, 50 \times 50, 50 \times 50, 50 \times 50$) with $ReLU$ activations after each hidden layer and one output layer (50×1) with a total of 7650 parameters and compare against another deep equilibrium model with one linear layer (2×50), a $f_\theta(z; x) = \tanh(Wz + x)$ deep implicit layer where we solve for the fixed point $z^*_\theta(x)$ and $W \in \mathbb{R}^{50 \times 50}$ followed by a final linear output layer (50×1) with a total of 2650 (almost one-third of the previous one) trainable parameters. By using the tricks detailed in section 1, we can train the deep equilibrium model with the same physics informed loss and using an automatic differentiation software like PyTorch. For the forward pass, we have used Newton's method to compute the fixed point $z^*_\theta(x)$ [6]. We observe that the deep equilibrium model outperforms the vanilla fully connected neural network with almost half the parameters. This could be because the deep equilibrium model simulates a network of infinite depth and therefore, it is capable of learning enough representation to predict long term safety probability of such a stochastic dynamical equation.

We observe that the DEQ model performs comparably for the boundary losses but outperforms the fully connected model for the physics loss.

[6]The code can be found here: `https://github.com/ritabrata-ray/DEQ_for_safe_control`.

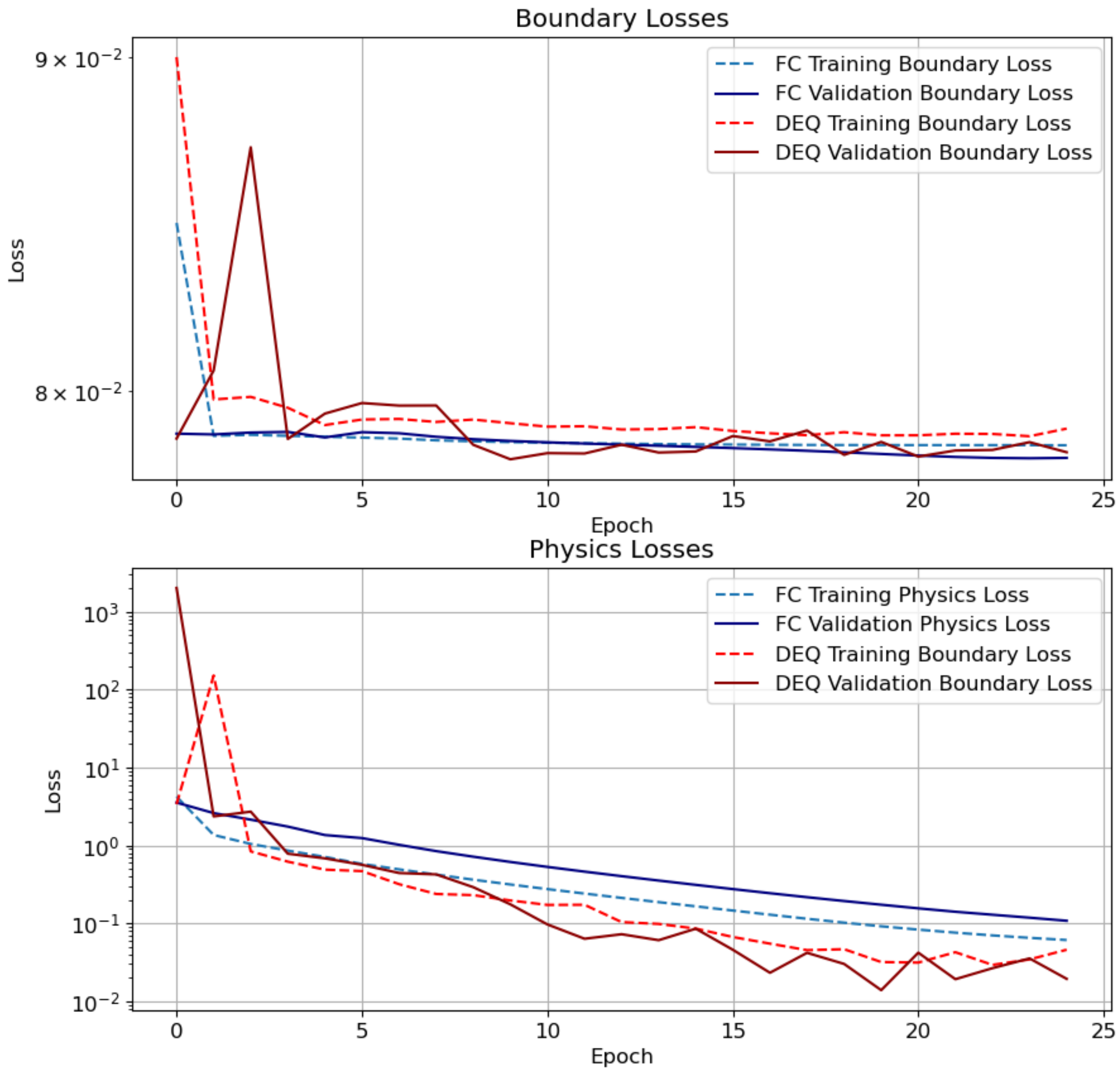

Figure 1: Comparing convergence rate of physics and boundary losses between fully connected and DEQ models.

Also, the above PDE (37), (38), (39) for the above $1 - D$ dynamics (36) has the following analytical solution for the risk probability:

$$F(x, T) = \frac{(2 - x)}{\sqrt{2\pi T^3}} \exp \frac{-((2 - x) - T/2)^2}{2T}. \tag{40}$$

This can be used to directly compute the estimation error on the risk probability by our trained model. We have the following training and validation loss for the fully connected (FC) model and the Deep Equilibrium (DEQ) model.

Further, we have the following plots for the analytical risk probability $F(x, T)$ and the estimation errors for the fully connected and the DEQ models over the whole domain D.

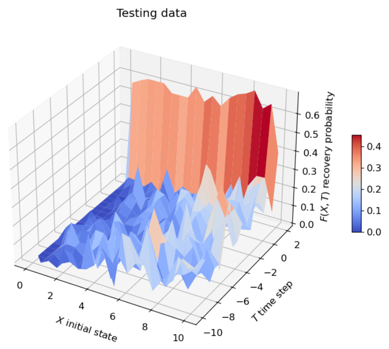

Figure 2: Analytical solution $F(x, T)$

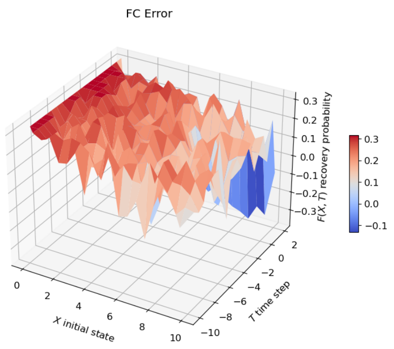

Figure 3: Fully connected layer model's prediction error

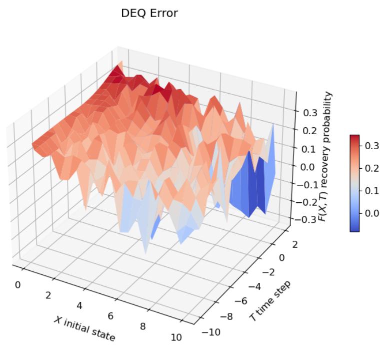

Figure 4: Deep Equilibrium Model's prediction error

5 Future work

We can investigate the case for higher dimensional systems where we do not have a closed form analytical solution and can only use the monte carlo samples $\hat{F}(x, T)$ to evaluate the prediction error on validation/test data. Further, we can extend the problem to autonomous control system where another FC/DEQ model learns he optimally safe control action ($u(x, t)$ maximizing $F(x, T)$) at each (x, T) where T is the remaining time till horizon. Such a controller design problem is similar to model-based RL, but here we have extra information about the exact dynamics as well as a PDE characterization of the risk probability $F(x, T)$. The goal of estimating $F_\theta(x, T)$ can be seen as similar to value function approximation and in RL.

References

D. G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, oct 1965. ISSN 0004-5411. doi: 10.1145/321296.321305. URL `https://doi.org/10.1145/321296.321305`.

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models, 2019.

A. Borodin. *Stochastic Processes*. Probability and Its Applications. Springer International Publishing, 2017. ISBN 9783319623108. URL https://books.google.co.in/books?id=jFE8DwAAQBAJ.

A. Chern, X. Wang, A. Iyer, and Y. Nakahira. Safe control in the presence of stochastic uncertainties, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

T. Marwah, A. Pokle, J. Z. Kolter, Z. C. Lipton, J. Lu, and A. Risteski. Deep equilibrium based neural operators for steady-state pdes, 2023.

G. Moustris, S. Hiridis, K. M. Deliparaschos, and K. Konstantinidis. Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 7, 2011. URL https://api.semanticscholar.org/CorpusID:43405844.

B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications (Universitext)*. Springer, 6th edition, Jan. 2014. ISBN 3540047581. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/3540047581.

Z. Wang and Y. Nakahira. A generalizable physics-informed learning framework for risk probability estimation, 2024.