# EpiPINN: Neural Network for Parameter Estimation of Epidemic Model

Ashton Cole

October 28, 2025

## 1 Introduction

In this project, we attempted to replicate the work of [1], using a physics-informed neural network (PINN) to learn epidemic data and estimate the parameters for a fractional-order SEIRD model. We had some difficulties in implementing parts of the paper, and the PINN did not learn SEIRD parameters accurately. In conclusion, more development is necessary for PINNs to be a reliable choice for epidemic modeling.

### 1.1 Caputo Fractional Calculus

The Caputo fractional derivative and integral are a means of extending derivatives and integrals past integer orders. The Caputo integral and derivative for $\alpha \in (0, 1]$ are defined using convolutions.

$$ {}^{\mathcal{C}}\mathcal{I}_t^{\alpha}[f](t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau)\,\mathrm{d}\tau \tag{1}$$

$$ {}^{\mathcal{C}}\mathcal{D}_t^{\alpha}[f](t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-\tau)^{-\alpha} f'(\tau)\,\mathrm{d}\tau \tag{2}$$

Because the fractional derivative requires integration from an initial time, it is dependent on the function globally, as opposed to being an instantaneous rate of change. This dependence is referred to as memory effects.

### 1.2 SEIRD Model

The SEIRD Model is an extension of the SIR compartmental model, used to model pandemic progression. The population is divided into five compartments: susceptible, exposed, infected, recovered, and dead. Changes the compartments are defined by the following system of ordinary differential equations.

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta \frac{SI}{N-D} \tag{3}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \beta \frac{SI}{N-D} - \sigma E \tag{4}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \sigma E - (\gamma + \mu) I \tag{5}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I \tag{6}$$

$$\frac{\mathrm{d}D}{\mathrm{d}t} = \mu I \tag{7}$$

Susceptible people are exposed at an infection rate $\beta$, exposed people become infected at an incubation rate $\sigma$, and infected people either recover at a recovery rate $\gamma$ or die at a mortality rate $\mu$. Importantly, the whole population $N = S + E + I + R + D$ is conserved.

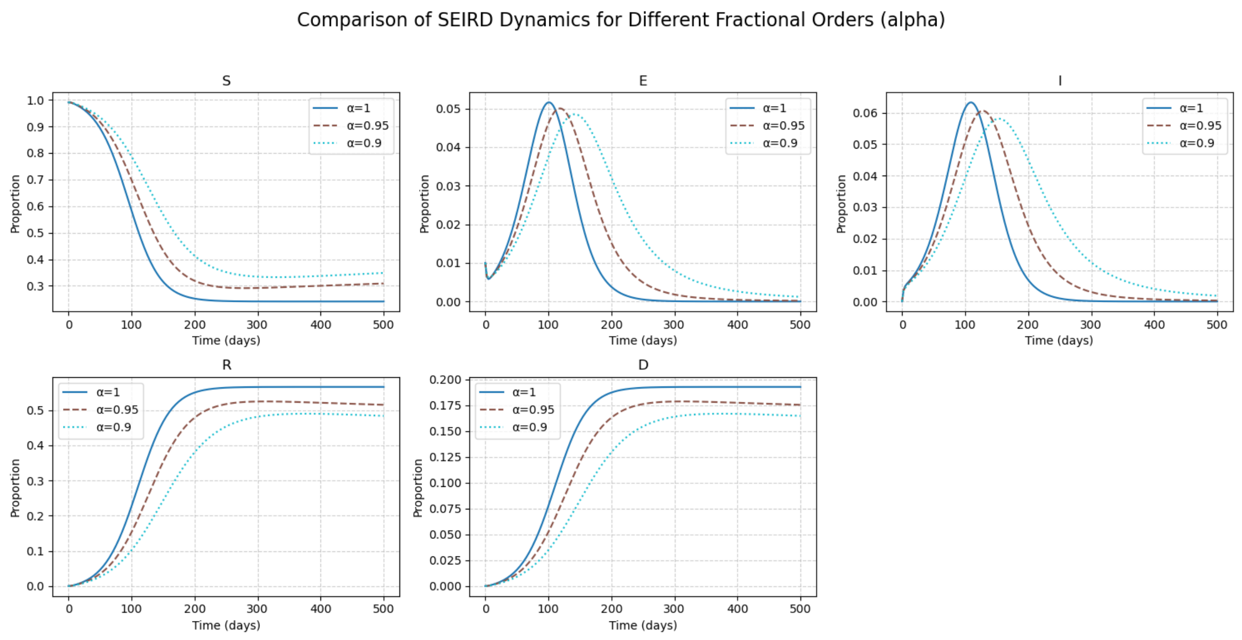

Figure 1: Normalized SEIRD dynamics for different fractional orders $\alpha$.

The paper replaces these first-order derivatives with the Caputo fractional ones, all of some order $\alpha$. In the ODE model, the rate of change of each compartment at a given time is dependent only on the state of that population at the same instant. Fractional differential equations (FDEs), however, make present dynamics dependent on the past, and have been shown to better fit epidemic data. The modified system, with compartments normalized by total population $N$, are shown below.

$$ {}^{C}\mathcal{D}_t^{\alpha}[s] = -\beta \frac{si}{1-d} \tag{8} $$

$$ {}^{C}\mathcal{D}_t^{\alpha}[e] = \beta \frac{si}{1-d} - \sigma e \tag{9} $$

$$ {}^{C}\mathcal{D}_t^{\alpha}[i] = \sigma e - (\gamma + \mu) i \tag{10} $$

$$ {}^{C}\mathcal{D}_t^{\alpha}[r] = \gamma i \tag{11} $$

$$ {}^{C}\mathcal{D}_t^{\alpha}[d] = \mu i \tag{12} $$

A potential flaw of fractional derivatives, however, is that they introduce unintuitive and nonphysical behavior. Although fractional derivatives will still conserve the total, they cause undesired movement between compartments. For example, notice how in Figure 1, after about 200 days, the number of susceptible people $s$ increases, while recovered people $r$ decreases. In the ODE case, this would be impossible, as their derivatives are strictly negative and positive, respectively. For a fractional derivative, however, a positive value does not always correspond to an increase, or vice-versa. Perhaps memory effects are useful in the initial stages of an epidemic, but clearly fractional derivatives fail at predicting long-term equilibrium. This behavior was not considered in the paper.

## 2 Methodology

In this section, we discuss our implementation of the work in the paper. We successfully implemented fractional calculus, the PINN, and training as described. However, we were not able to find the referenced compartmental time series, instead opting to generate our own data.

## 2.1 Implementation of Caputo Fractional Calculus

The Caputo fractional derivative was implemented for discrete time series of fixed step. The L1 scheme is used, meaning that the continuous definition is made discrete by approximating the function as piecewise linear between points. The following equation results, and is made very efficient by rewriting as a Toeplitz matrix multiplication.

$$f'_0 = 0 \tag{13}$$

$$f'_{k+1} = \frac{1}{\Gamma(2-\alpha)} \sum_{j=0}^{k} \frac{f(t_{j+1}) - f(t_j)}{t_{j+1} - t_j} \left( (t_{k+1} - t_j)^{1-\alpha} - (t_{k+1} - t_{j+1})^{1-\alpha} \right) \tag{14}$$

$$= \frac{(\Delta t)^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{k} \left( f(t_{j+1}) - f(t_j) \right) \left( (k+1-j)^{1-\alpha} - (k-j)^{1-\alpha} \right) \tag{15}$$

$$= \frac{(\Delta t)^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{k} \Delta f_j w_{k-j} \tag{16}$$

$$\Delta f_j = f(t_{j+1}) - f(t_j) \tag{17}$$

$$w_{k-j} = (k+1-j)^{1-\alpha} - (k-j)^{1-\alpha} \tag{18}$$

$$\begin{bmatrix} f'_1 \\ f'_2 \\ f'_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} w_0 & & & \\ w_1 & w_0 & & \\ w_2 & w_1 & w_0 & \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \Delta f_1 \\ \Delta f_2 \\ \Delta f_3 \\ \vdots \end{bmatrix} \tag{19}$$

The Caputo-Euler method for a FDE ${}^{C}\mathcal{D}_t^{\alpha}[\mathbf{y}] = f(t, \mathbf{y})$ is implemented with a similar form, but must be computed in a loop.

$$\mathbf{y}_{k+1} = \mathbf{y}_0 + \frac{(\Delta t)^{\alpha}}{\Gamma(\alpha+1)} \sum_{j=0}^{k} f(t_j, \mathbf{y}_j) \left( (k+1-j)^{1-\alpha} - (k-j)^{1-\alpha} \right) \tag{20}$$

$$= \mathbf{y}_0 + \frac{(\Delta t)^{\alpha}}{\Gamma(\alpha+1)} \sum_{j=0}^{k} \mathbf{f}_j w_{k-j} \tag{21}$$

$$\mathbf{f}_j = f(t_j, \mathbf{y}_j) \tag{22}$$

$$w_{k-j} = (k+1-j)^{1-\alpha} - (k-j)^{1-\alpha} \tag{23}$$

## 2.2 Implementation of Neural Network

We implemented a PINN in PyTorch based on the structure described in the paper. The network models the state as a function of time, $(s, e, i, r, d) = NN(t)$. It has one input dimension and five output dimensions. Perceptron dimenions were not specified, so after some experimentation, a sequential neural network with three layers of 64 perceptrons was assumed. Tanh activation functions are used between layers, while a softmax function is applied to the output layer to force normalized population conservation. Fractional order and FDE coefficients and $\alpha, \beta, \sigma, \gamma, \mu$ are included as trainable parameters, but are not stored directly. To ensure positivity for the disease rates, raw scores are stored internally, and a softplus function is applied whenever they are used. Meanwhile, for $\alpha$, the range is restricted to $(\alpha_{\min}, 1)$ by a scaled sigmoid function.

## 2.3 Training Process

Training takes place in two stages, outlined in table 1. The first stage fits the data, and the second learns the FDE. First, weights are initialized using uniform Xavier initialization. The following losses are used for training.

Table 1: Configuration for each training stage.

| | Parameters Trained | Loss Function | Optimizer |
|---|---|---|---|
| Stage 1 | $\Theta$ | $\mathcal{L}_{\text{data}} + \mathcal{L}_{\text{IC}}$ | Adam |
| Stage 2 | $\Theta, \alpha, \beta, \gamma, \mu$ | $\mathcal{L}_{\text{data}} + \mathcal{L}_{\text{IC}} + 10,000\mathcal{L}_{\text{phys}}$ | L-BFGS |

Table 2: Epidemiological parameters for experiment 1.

| | $\alpha$ | $\beta$ | $\gamma$ | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|
| Exact | 0.994 | 0.166 | 0.117 | 0.077 | 0.02 |
| Initial Guess | 0.96 | 0.26 | 0.14 | 0.053 | 0.006 |
| Estimate | 0.962 | 0.151 | 0.0789 | 0.116 | 0.00662 |

$$\mathcal{L}_{\text{data}} = \frac{1}{N}\sum_{n=1}^{N} \|(s(t_n), e(t_n), i(t_n), r(t_n), d(t_n)) - NN(t_n)\|_2^2 \tag{24}$$

$$\mathcal{L}_{\text{IC}} = \|(s_0, e_0, i_0, r_0, d_0) - NN(t_0)\|_2^2 \tag{25}$$

$$\mathcal{L}_{\text{phys}} = \frac{1}{M}\sum_{m=1}^{M} \left\|{}^{C}\mathcal{D}_t^{\alpha}[NN](\hat{t}_n) - f(t_n, NN(\hat{t}_m))\right\|_2^2 \tag{26}$$

Data loss is the MSE of predictions against data. IC loss is the MSE of the initial condition. Physics loss is the square norm of the FDE residual computed at collocation points $\hat{t}_m$, where the FDE is ${}^{C}\mathcal{D}_t^{\alpha}[(s, e, i, r, d)] = f(t, (s, e, i, r, d))$. Unlike in the paper, conservation loss is not implemented directly. but, instead enforced on the output with softmax.

Both training stages are relatively standard. They just minimize different objectives and train different parameters. Early stopping is implemented in both with a patience counter. If loss does not improve after a certain number of epochs, training stops. The second stage uses the L-BFGS optimizer, because it is shown to work well for parameter estimation in the literature.

## 3 Results and Discussion

Two experiments were conducted to evaluate the PINN. In experiment 1, it was trained on FDE ground truth data. Stage 1 ran for 5000 epochs using a learning rate of 1e-3 and weight decay 1e-5. Stage 2 ran for 5,000 epochs using a learning rate of 1e-5. Physics loss was weighted by a factor of 1e4 to emphasize FDE residual convergence. After initial training, the model fit the FDE solution quite well, but after stage 2, the neural network had a worse fit. Unlike the paper, we also studied parameter convergence across epochs. Epidemiological parameters seemed to converge quite quickly, but did so to incorrect values. Clearly, this reduced the physics loss computation, but lead to a worse overall.

For experiment 2, a similar setup was used, but noise was added to the data. Parameters were additionally randomized to a similar range. Again, stage 2 of training led to a worse data fit, and eoidemiological parameters converged to inaccurate values.

Table 3: Epidemiological parameters for experiment 2.

| | $\alpha$ | $\beta$ | $\gamma$ | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|
| Exact | 0.95 | 0.25 | 0.13 | 0.052 | 0.005 |
| Initial Guess | 0.96 | 0.26 | 0.14 | 0.053 | 0.006 |
| Estimate | 0.959 | 0.246 | 0.0534 | 0.121 | 0.00587 |

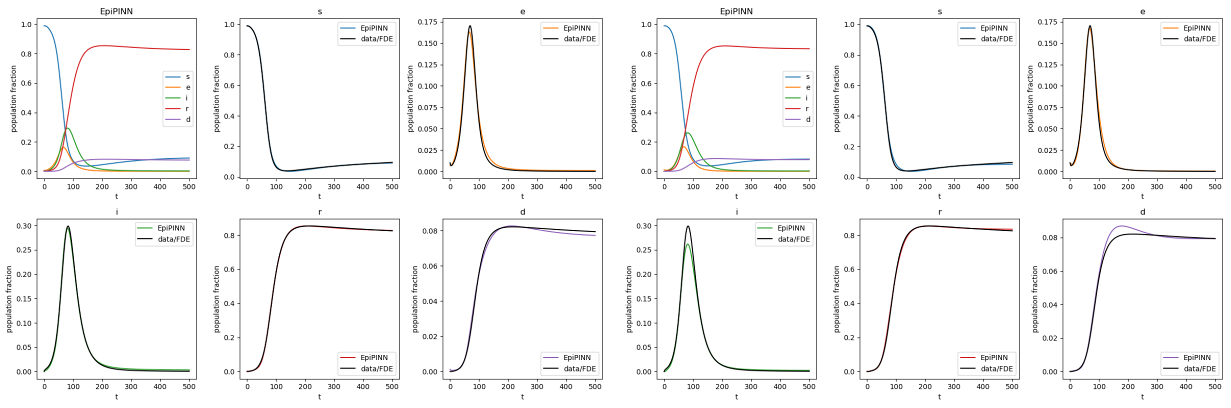

Figure 2: Experiment 1 model outputs after stages 1 and 2.

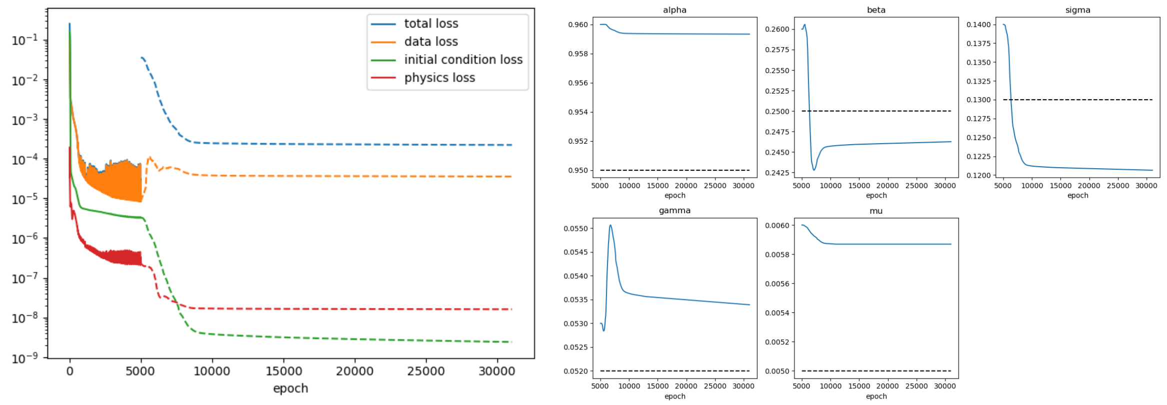

Figure 3: Experiment 1 loss and parameter convergence.

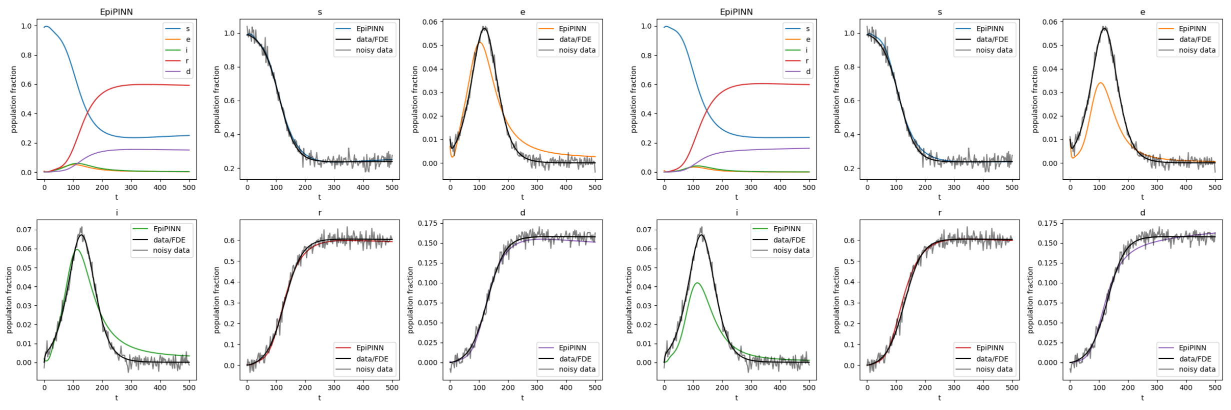

Figure 4: Experiment 1 model outputs after stages 1 and 2.

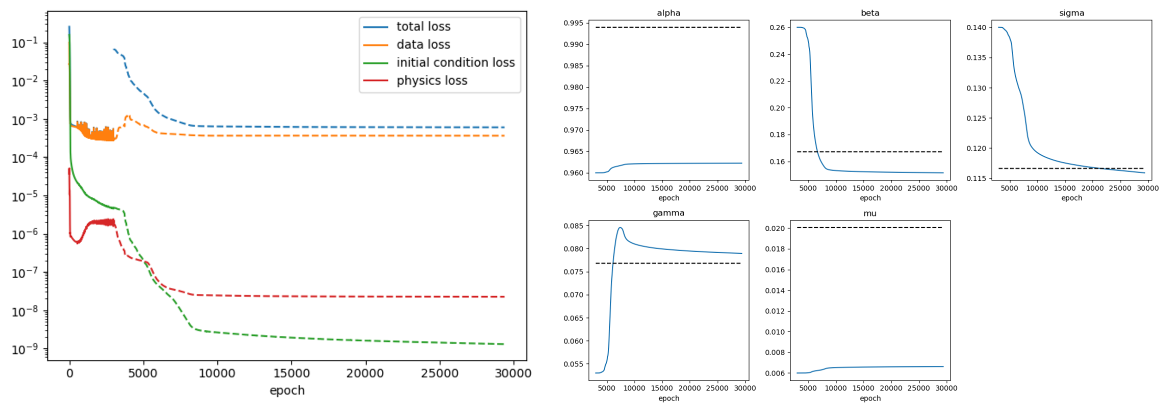

Figure 5: Experiment 1 loss and parameter convergence.

## 4 Conclusions

In conclusion, while the network seemed to converge with precision, it did so to unacceptably inaccurate values, particularly for FDE-generated data. It is possible that too many degrees of freedom were added to the physics FDE. With split objectives of minimizing data and FDE residual loss, the model seems to prefer fitting the network to a new FDE as opposed to recovering the original FDE. We also have questions about the paper's accuracy. From the table they provided, it is clear that their PINN, like our own, generally predicts values in the neighborhood of the correct ones, but they do not provide initial guesses and convergence plots.

We successfully implemented the model described in the paper. However, imrpoving parameter convergence is a major concern to address before such a model could be used to reliably model epidemics.

## References

[1] Achraf Zinihi. Identifying Memory Effects in Epidemics via a Fractional SEIRD Model and Physics-Informed Neural Networks, September 2025. arXiv:2509.22760 [stat].