CODEE Journal

Volume 18 Article 4

8-13-2024

Mathematical Modeling, Analysis, and Simulation of Patient Addiction Journey

Adan Baca
University of Arizona

Diego Gonzalez
University of La Verne

Alonso G. Ogueda
George Mason University

Holly C. Matto
George Mason University

Padmanabhan Seshaiyer
George Mason University

Mathematical Modeling, Analysis and Simulation of Patient Addiction Journey

Adán Baca
University of Arizona

Diego González
University of La Verne

Alonso Ogueda-Oliva, Holly Matto and Padmanabhan Seshaiyer
George Mason University

Keywords: Compartmental Models, Neural Networks, Drug Addiction, Literate Programming.

Manuscript received on April 18, 2024; published on To Be Determined.

Abstract: This paper aims to develop a mathematical model to study the dynamics of addiction as individuals go through their addiction journey. The motivation for this work is three fold. First, there has been a significant increase in drug overdose and drug addiction following the COVID-19 pandemic, and addiction may be interpreted as a infectious disease. Secondly, the dynamics of infectious disease could be modeled via compartmental models described by differential equations and one can therefore leverage the existing analytical and numerical methods to model addiction as a disease. Finally, the work helps to inform how mathematical models governed by differential equations, can help to provide insights into societal challenges such as addition. The proposed work describes a modified SIR differential equation system for studying the non-linear dynamics various sub-populations among the drug users. A detailed mathematical analysis along with the derivation of the basic reproduction number for the proposed model is also presented. Furthermore, we numerically simulate the dynamics and validate the model against synthetic data for COVID-19 cases to show the robustness and reliability of our model. We also introduce a Physics Informed Neural Network approach that helps to estimate the parameters in the model efficiently. Integrated with the research is an instructional exposition using Literate Programming that an help educators to enhance their own pedagogical practices to engage student interaction in the classroom.

1 Introduction

It is well known that undergraduate students are exposed to two foundational concepts in Calculus, which include differential equations through the notion of a derivative, followed by the anti-derivative and connecting these through the Fundamental Theorem of Calculus. Once exposed, they learn that these differential equations or system of differential equations often do not admit exact solutions and hence recognize the need to develop numerical methods to solve them [14].

Along with evolving interests in modeling real-world dynamical systems motivated by Calculus, there have also been technological advances to incorporate graphical and numerical approaches to understand and analyze solutions to differential equations that were previously reserved for advanced undergraduate or graduate study. Technology is also often used to enhance the student learning of differential equations and to help them appreciate the applications of these equations to real-world problems through mathematical modeling [15, 2, 7]. An example is the applications to understand and study infectious diseases using compartmental models described via coupled differential equations [14].

With the developments in models and modeling, one of the innovative instructional approaches to engage students in learning the needed content along with a structured programming was introduced about four decades back [4]. Specifically, the goal was to change a traditional attitude of the construction of the programs, but to imagine the main task was to instruct a computer on what to do giving more time for the instructor to explain to students what we want a computer to do. This idea of programming became known as *literate programming*, where instruction drives the coding and the output of the piece of code informs the instruction in real-time. Over the years the literate programming style has been used to enhance student learning of differential equations [15, 8].

Another grand challenge is the need to develop problem-solving frameworks that enable researchers to blend differential equations with the vast amount of available data sets. These data sets are also needed to efficiently estimate the parameters used in the governing differential equations [13]. Over the recent years, parameter estimation has gained the attention of researchers and educators with a need to develop data proficiency for students to apply the mathematics they know to solve problems arising in everyday life, society, and the workplace and make data-driven predictions.

In this work, we introduce a novel approach to using differential equations to model, analyze and simulate the nonlinear dynamics of a patient's journey through addiction. In particular, the mathematical framework considered builds on well-established compartmental modeling from epidemiology. In section 2, we introduce the mathematical model to understand the dynamics of addiction. Section 3 presents an analysis including verifying the non-negativity and boundedness of the solutions of the equations along with derivation of the basic reproduction number. Following the analysis, results of computational experiments are presented in Section 4 on a benchmark problem that includes dynamic analysis and parameter estimation techniques. Specifically, the section presents an application in a literate programming style for readers to follow along and learn to implement the model in Python. In particular, this section compares the traditional nonlinear least-squares estimation to an alternate approach introduced called Physics Informed

Neural Networks (PINNs) technique. Section 5 summarizes this work as a conclusion and discusses potential extensions for future work. Finally, source code is publicly available on https://github.com/AdanXBaca/detox_journey.

2 Mathematical Model and Background

Over the past several decades, compartmental models have been employed with mathematical modeling to study the spread of infectious diseases [1]. One of the original mathematical conceptualizations of how disease spreads was introduced to describe the interaction between Susceptible (S), Infectious (I), and Recovered (R) human sub-populations, which are known as SIR models [3]. These models have since expanded to SEIR that includes an 'E' compartment for the latent period during which individuals have been infected but are not yet infectious; SEIRS model that allows the possibilit of individuals losing immunity over time and returning to the susceptible population; SIRS model which is similar to SEIRS that allows individuals to move from the recovered (R) compartment back to the Susceptible (S) compartment after a certain period indicating loss of immunity over time. Besides these models have been studied extensively with Vaccination and Quarantine as well. One can also find network-based models for these that consider the structure of social networks and how individuals interact within them. Other types of models include age-structured models that divide the population into different age groups and consider age-specific transmission rates, susceptibility, and mortality rates.

While most of these models were used to track the dynamics of diseases, recently there have been applications to employ such compartmental models to study the spread of other societal challenges such as depression [17], alcoholism [6], drug epidemics [9] and domestic violence [11]. Modeling addiction as an infectious disease can be an insightful approach because it allows us to explore how behaviors and social interactions spread within populations. Just as infectious diseases spread from person to person, addictive behaviors can also spread within social networks. For example, if someone in a social group uses substances or engages in addictive behaviors, others in the group might be more likely to do the same due to social influence or peer pressure. Furthermore, viewing addiction through the lens of infectious disease models can help in designing effective prevention strategies. For example, targeting high-risk individuals or communities, much like vaccinating in disease outbreaks, can be a strategy to reduce the spread of addictive behaviors. Next, we will describe how we employ compartmental differential equations to model the journey of an addict as they transition through their addiction journey after returning from rehabilitation.

2.1 Modeling addiction behavior

The first model proposed in this work considers five compartments including, Susceptible (S), Exposed (E), Drug Addicts (I), Hospitalized (H) and Recovered (R) individuals yielding a SEIHR model, for simplictiy. Here drug addiction is considered as a disease that allows the patient to transition through the various compartments. In our model, we assume a Susceptible (S) person can get exposed (E) through casual interactions with other

addicted (or infected) individuals and then later get fully addicted (I). The idea behind the journey of the individual is that once that person becomes infected he/she may have the possibility to overdose in which case they will be rushed to a hospital (H) to receive detoxification or decide to go into a recovery program (R). In addition, upon completion of the recovery program, assuming they are recovered from addiction they can now go back to the exposed population (*recidivism*), now classified as prior drug users, with the cyclic possibility that that person can become addicted or infected once again. Note that there is also a possibility of those that were hospitalized to either return to being addicted or recommended to enter a recovery program. We will describe the model with vital dynamics including Λ, that relates to *recruitment* and μ, that denotes the *natural* death rate.

Figure 1 shows the flow between each compartments. The governing differential

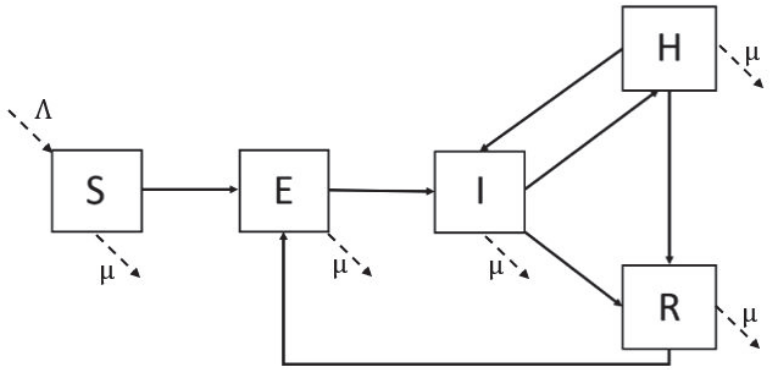

Figure 1: Patient Addiction Journey SEIHR Model Flow Diagram.

equation system for this dynamics can be described as follows:

$$\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{d}t} &= \Lambda - \frac{\beta SI}{N}(1-b) - \frac{\epsilon\beta SE}{N}(1-b) - \mu S \\
\frac{\mathrm{d}E}{\mathrm{d}t} &= \frac{\beta SI}{N}(1-b) + \frac{\epsilon\beta SE}{N}(1-b) - \sigma E + r_1 R - \mu E \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= \sigma E - \gamma I - \eta I - \mu_1 I + r_2 H - \mu I \\
\frac{\mathrm{d}H}{\mathrm{d}t} &= \eta I - \omega H - \mu_2 H - r_2 H - \mu H \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I + \omega H - r_1 R - \mu R,
\end{aligned} \tag{2.1}$$

where N is the total population, β, σ, γ, η and ω are the transmission, infection, direct recovery, hospitalization rate, recovery rate through hospitalization respectively, ϵ is a proportion of infection for exposed individuals; r_1 corresponds to recidivism from recovery to exposed, r_2 is relapse rate for hospitalized individuals, μ_1 and μ_2 are the *overdose* death rates for infected and hospitalized individuals, respectively, (which is different from *natural* death rates) and b is a measure of human behavior that eventually could be under

the users control. For example, if someone practices good behavior and avoids addiction, the value of b could be considered to be 1. Therefore, an increase in the value of b would correspond to increased level of addiction.

2.2 Modeling with Treatment

The second model considered in this work includes six compartments, an enhancement of the SEIHR model described and corresponding to adding a treatment (T) compartment making it a SEIHRT model. This treatment compartment is vital in understanding how some part of the treated population can relapse after receiving detoxification at the hospital. Additionally, the R compartment is now the fully recovered individuals. The recovered individuals can still go back to the exposed population but with a slighter chance of becoming addicted again, assuming the recovery program is unsuccessful (see Figure 2). This new model corresponds to the system of differential equations (2.2), where α is the

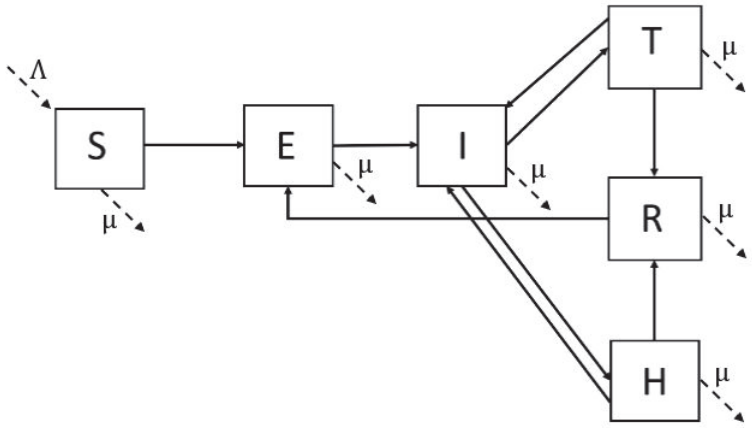

Figure 2: Patient Addiction Journey SEIHRT Model Flow Diagram.

rate of infected individuals who enrolled in the treatment program, δ is the rate of treated individuals who recover and r_3 is the relapse rate of individuals in treatment who become addicted again.

$$
\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{d}t} &= \Lambda - \frac{\beta SI}{N}(1-b) - \frac{\epsilon\beta SE}{N}(1-b) - \mu S \\
\frac{\mathrm{d}E}{\mathrm{d}t} &= \frac{\beta SI}{N}(1-b) + \frac{\epsilon\beta SE}{N}(1-b) - \sigma E + r_1 R - \mu E \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= \sigma E - \gamma I - \eta I - \alpha I - \mu_1 I + r_2 H + r_3 T - \mu I \\
\frac{\mathrm{d}H}{\mathrm{d}t} &= \eta I - \omega H - \mu_2 H - r_2 H - \mu H \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= \delta T + \gamma I + \omega H - r_1 R - \mu R \\
\frac{\mathrm{d}T}{\mathrm{d}t} &= \alpha I - r_3 T - \delta T - \mu T
\end{aligned}
\tag{2.2}
$$

3 Mathematical Analysis

In this section, we will show the well-posedness for the SEIHR model described in (2.1). Following that we will derive the basic-reproduction number for the model proposed.

First, we have the following result for the non-negativity of the solutions.

3.1 Non-negativity of the solutions

Theorem 3.1. *Let the initial condition $S(0), E(0), I(0), H(0), R(0)$ be non-negative. Then the solutions of the SEIHR system (2.1) are non-negative in $t \in [0, \infty)$.*

Proof. All the right hand terms of the SEIHR system are continuous and locally Lipschitz on $\mathbb{R}$. The solution $\{S(t), E(t), I(t), H(t), R(t)\}$ with their initial conditions exist and are unique in the interval $[0, \infty)$ [5]. From the first equation in (2.1)

$$\frac{\mathrm{d}S}{\mathrm{d}t} = \Lambda - \frac{\beta SI}{N}(1-b) - \frac{\epsilon\beta SE}{N}(1-b) - \mu S,$$

we let $G(t) = \dfrac{\beta I}{N}(1-b) + \dfrac{\epsilon\beta E}{N}(1-b) + \mu$, we obtain the following result

$$\frac{d}{dt}\left[Se^{\int_0^t G(\tau)\,d\tau}\right] = \Lambda e^{\int_0^t G(\tau)\,d\tau}.$$

Integrating both sides with respect to t, we obtain,

$$S(t_1)e^{\int_0^{t_1} G(\tau)\,d\tau} - S(0) = \int_0^{t_1} \Lambda e^{\int_0^x G(\tau)\,d\tau}\,dx.$$

Since $S(0) \geq 0$, we have,

$$S(t_1)e^{\int_0^{t_1} G(\tau)\,d\tau} \geq \int_0^{t_1} \Lambda e^{\int_0^x G(\tau)\,d\tau}\,dx,$$

which yields:

$$S(t_1) \geq e^{-\int_0^{t_1} G(\tau)\,d\tau} \int_0^{t_1} \Lambda e^{\int_0^x G(\tau)\,d\tau}\,dx > 0.$$

This proves $S(t) > 0$. Similarly, we can show $E(t) > 0, I(t) > 0, H(t) > 0, R(t) > 0$, starting with the respective equation in the system (2.1). □

Next, we prove boundedness of the solutions in a region.

3.2 Boundedness of the Solutions

Theorem 3.2. *All solutions of the proposed system (2.1), are bounded inside the region* $\left\{\mathfrak{X}(t) \in \mathbb{R}^5 : 0 \leq N(t) \leq \frac{\Lambda}{\mu}\right\}$.

Proof. Let us denote $m = \min(\mu_1, \mu_2)$ and $M = m(N - S - E - R)$. Recalling that the total population

$$N(t) = S(t) + E(t) + I(t) + H(t) + R(t),$$

and adding the equations in the system (2.1) we have

$$N'(t) \quad = \quad \Lambda - \mu N(t) - M \leq \Lambda - \mu N(t)$$

We then have the following inequality,

$$N'(t) + \mu N(t) \leq \Lambda$$

which can be simplified to yield,

$$N(t) \leq \left(N(0) - \frac{\Lambda}{\mu}\right) e^{-\mu t} + \frac{\Lambda}{\mu}$$

Letting $t \longrightarrow \infty$, we then get the solution $N(t) \in \left[0, \max\left\{N(0), \frac{\Lambda}{\mu}\right\}\right]$. □

Next, we derive the basic reproduction number $\mathcal{R}_0$.

3.3 Basic Reproduction Number

In this section, we will derive a basic reproduction number $\mathcal{R}_0$ that can be used to measure the transmission potential of a disease, utilizing the SEIRH model (2.1). $\mathcal{R}_0$ is the average number of secondary infections produced by a typical case of an infection in a population where everyone is susceptible. We employ a general approach called the *Next Generation Matrix* to find the basic reproduction number $\mathcal{R}_0$ which is given by the following theorem.

Theorem 3.3. *The basic reproduction number* $\mathcal{R}_0$ *of models (2.1) and (2.2) is given by*

$$\mathcal{R}_0 = \mathcal{R}_0^1 + \mathcal{R}_0^2 \tag{3.1}$$

where

$$\mathcal{R}_0^1 = \frac{\epsilon\beta(1-b)}{\sigma + \mu} \quad \textit{and} \quad \mathcal{R}_0^2 = \frac{\sigma\beta(1-b)}{(\sigma+\mu)(\gamma+\eta+\mu_1+\mu)}.$$

Proof. For the next generation matrix of the SEIHR model (2.1), given that the infectious states include E and I, we can create a vector $\mathcal{F}$ that represents the new infections flowing only into the exposed compartments given by:

$$\mathcal{F} = \{\beta I(1-b) + \epsilon\beta E(1-b), 0\}$$

Along with this we also consider another vector $\mathscr{V}$ which denotes the outflow from the infectious compartments in (2.1) which is given by:

$$\mathscr{V} = \{\sigma E - r_1 R + \mu E, -\sigma E + (\gamma + \eta + \mu_1 + \mu)I + r_2 H\}$$

Note that this computation assumes that the susceptible population at the beginning is almost the total population. Then we can compute the respective Jacobian matrices F and V, respectively, as:

$$F = \begin{bmatrix} \epsilon\beta(1-b) & \beta(1-b) \\ 0 & 0 \end{bmatrix}$$

and

$$V = \begin{bmatrix} \sigma + \mu & 0 \\ -\sigma & \gamma + \eta + \mu_1 + \mu \end{bmatrix}$$

Note that the inverse of V is given by:

$$V^{-1} = \frac{1}{(\sigma + \mu)(\gamma + \eta + \mu_1 + \mu)} \begin{bmatrix} \gamma + \eta + \mu_1 + \mu & 0 \\ \sigma & \sigma + \mu \end{bmatrix}$$

The basic reproduction can now be computed as the largest nonzero eigenvalue of the resulting matrix, FV^{-1} given by:

$$FV^{-1} = \begin{bmatrix} \dfrac{\epsilon\beta(1-b)}{\sigma + \mu} + \dfrac{\sigma\beta(1-b)}{(\sigma + \mu)(\gamma + \eta + \mu_1 + \mu)} & \dfrac{\beta(1-b)}{\gamma + \eta + \mu_1 + \mu} \\ 0 & 0 \end{bmatrix}$$

The eigenvalues are $\lambda = \dfrac{\epsilon\beta(1-b)}{\sigma + \mu} + \dfrac{\sigma\beta(1-b)}{(\sigma + \mu)(\gamma + \eta + \mu_1 + \mu)}$ and $\lambda = 0$ where the first one is the largest one and we get the desired result. □

4 Computational Experiments

To gain a better understanding of the dynamics of our proposed models, and as well as the behavior of the basic reproduction numbers, we implemented them in Python. During this process, we were also comparing the two models, and gaining insight into how the effects of recidivism and treatment have an effect on addicted populations.

We began our analysis by graphing each model individually, to ensure that we can visualize the dynamics and growth curves that resemble that of a typical SIR model. In order to achieve that, we used predetermined parameters (see Table 1).

4.1 Basic Reproduction Number

We first plot the basic reproduction number for the given parameter values and for varying *beta* and b. Figure 3 shows how $\mathscr{R}_0$ from (3.1) varies as β and b change. Note that for high values of b which captures human behavior and low values of β (transmission rate), the basic reproduction number $\mathscr{R}_0$ is well controlled as expected.

Parameter	Value	Parameter	Value	Parameter	Value
β	0.500	η	0.142	ω	0.500
σ	0.900	α	0.200	δ	0.700
ϵ	0.700	r_1	0.400	μ_1	0.050
b	0.000	r_2	0.500	μ_2	0.010
γ	0.071	r_3	0.100	Λ, μ	0.000

Table 1: Preset parameter values.

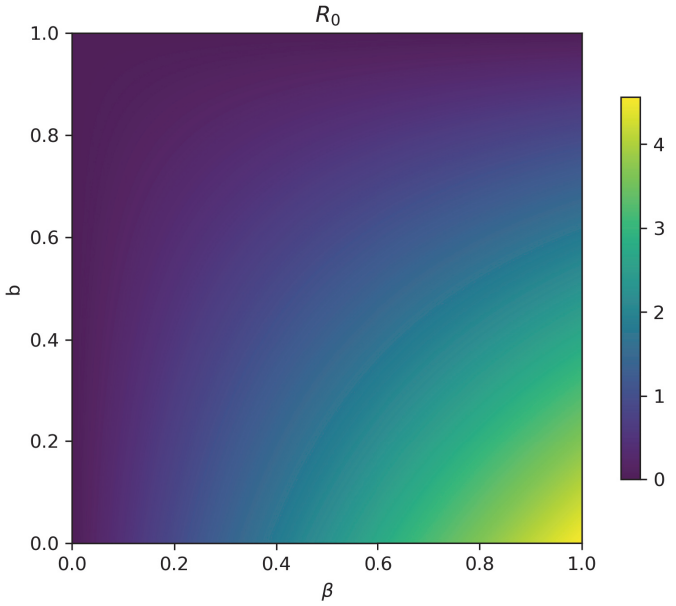

Figure 3: Basic reproduction number (3.1) varying β and b.

4.2 Literate Programming

In this section, we employ a literate programming framework [10] to explain how to implement the two models described by system (2.1) and system (2.2). The implementation is in Python using Google colab notebooks.

The first step is to import packages for scientific computation in Python (`NumPy` and `SciPy`), data analysis (`Pandas`) and visualization (`matplotlib` and `Seaborn`). This can be initiated using the sequence of commands in the following block.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy.integrate import solve_ivp
```

Next, the SEIHR model (2.1) is defined as a function script depending of time `t`, compartmental data `y` and its parameters.

```
def seihr(t, y, beta, b, epsilon, r1, r2, sigma, gamma, eta, omega, mu1, mu2,
```

```
    mu, Lambda):
    S, E, I, H, R = y  # Unpack compartments
    dS_dt = Lambda - beta * S * I * (1-b) - epsilon * beta * S * E * (1-b) - mu
     * S
    dE_dt = beta * S * I * (1-b) + epsilon * beta * S * E * (1-b) - sigma * E +
     r1 * R - mu * E
    dI_dt = sigma * E - gamma * I - eta * I - mu1 * I + r2 * H - mu * I
    dH_dt = eta * I - omega * H - mu2 * H - r2 * H - mu * H
    dR_dt = gamma * I + omega * H - r1 * R - mu * R
    return np.array([dS_dt, dE_dt, dI_dt, dH_dt, dR_dt])
```

In order to run the script, we need to provide the parameter values according to Table 1 which can be entered next.

```
N = 1000  # Population
S_0, E_0, I_0, H_0, R_0 = N - 51, 1, 50, 0, 0  # Initial conditions
y0 = np.array([S_0, E_0, I_0, H_0, R_0]) / N
beta = 0.5
b = 0
epsilon = 0.7
r1 = 0.4
r2 = 0.5
sigma = 0.9
gamma = 1 / 14
eta = 1 / 7
omega = 0.5
mu1 = 0.05
mu2 = 0.01
mu = 0
Lmbda = 0
params = (beta, b, epsilon, r1, r2, sigma, gamma, eta, omega, mu1, mu2, mu,
    Lmbda)
n_days = 60  # 2 months
t_data = np.arange(0, n_days, 1)  # Array of days
t_span = (t_data[0], t_data[-1])  # Time span
```

The next step involves solving the initial value problem numerically using a high order method. This can be done using `integrate.solve_ivp` from `SciPy` package, which uses a Runge-Kutta approach by default.

```
sol = solve_ivp(seihr, t_span, y0, args=params, dense_output=True)
y_data = sol.sol(t_data)
```

The array `y_data` contains the numerical prediction of each compartment. Visualizing the dynamics could be done from different approaches, even using other packages. We have taken an approach that combines `Pandas` for a data frame structure and `matplotlib` and `Seaborn` as visualization tools.

```
model_name = "SEIHR"
populations_names = list(model_name)
data = (
    pd.DataFrame(y_data.T, columns=populations_names)
    .assign(time=t_data)
    .melt(id_vars="time", var_name="status", value_name="population")
)
```

```
fig, ax = plt.subplots(figsize=(8, 4))
sns.lineplot(
    data=data,
    x="time",
    y="population",
    hue="status",
    legend=True,
    ax=ax
)
ax.set_title(f"{model_name} model")
fig.tight_layout()
fig.savefig(f"{model_name}_dynamics.png", dpi=300)
```

The result of this last cell of code is shown in Figure 4. Figure 5 can similarly be generated

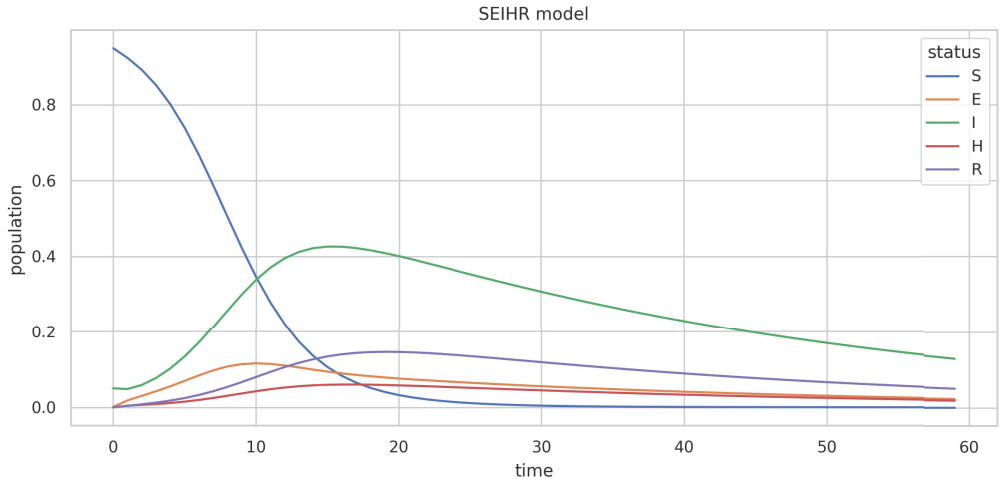

Figure 4: Non-linear dynamics of model SEIHR (2.1)

for system (2.2) also. Note that the peaks of the exposed and infected are lower in Figure 5 in comparison to Figure 4 showing the efficacy of the treatment in model (2.2). Also, the models contain behavior that resembles that of a standard SIR model, however, the differences can be accounted to the influences of different parameters, as well as the influence of recidivism on specific compartments.

4.3 Dynamic Analysis

After observing the dynamics of our models, next we consider the effects of various parameters. More specifically, we wish to observe the impact of the more important parameters, such as the transmission rate β. For this we altered the value of β and recorded the maximum measurement of the infectious states that included the exposed, infected and the sum of their values over time. We then plotted these values for various β values.

The approach taken here is to generate an array of many values of β and then solve the system of differential equations for each one of these values. This is illustrated using the following block of code.

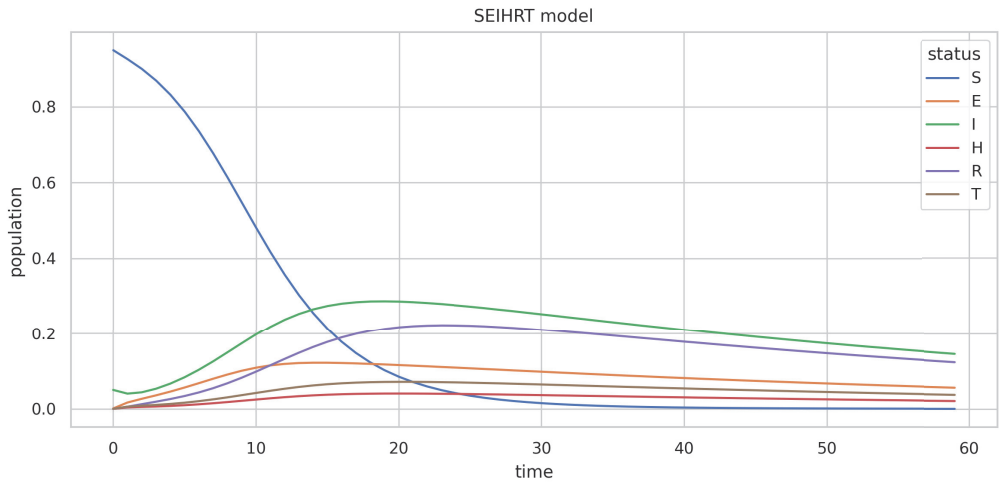

Figure 5: Non-linear dynamics of model SEIHRT (2.2)

```
n_betas = 1001
betas = np.linspace(0, 1, n_betas)
max_infect_array = np.zeros(shape=(n_betas, 4), dtype=float)
for i, beta in enumerate(betas):
    sol = solve_ivp(
        seihr,
        t_span,
        y0,
        t_eval=t_data,
        args=(
            beta, b, epsilon, r1, r2, sigma,
            gamma, eta, omega, mu1, mu2, mu, Lmbda
        ),
        dense_output=True
    )
    y_data = sol.sol(t_data)
    E_max = y_data[1, :].max()
    I_max = y_data[2, :].max()
    EI_max = y_data[1:3, :].sum(axis=0).max()
    max_infect_array[i, :] = [beta, E_max, I_max, EI_max]
```

The data can now be visualized using the following lines of code.

```
fig, ax = plt.subplots(figsize=(8, 6), facecolor="white")
ax.plot(max_infect_array[:, 0], max_infect_array[:,1], label="max(E)")
ax.plot(max_infect_array[:, 0], max_infect_array[:,2], label="max(I)")
ax.plot(max_infect_array[:, 0], max_infect_array[:,3], label="max(E+I)")
ax.legend()
ax.set_xlabel(r"$\beta$")
ax.set_ylabel("Population")
ax.set_ylim(0, 1)
ax.set_title(f"{model_name} Exposed and Infected peaks")
fig.savefig(f"{model_name}_infection_peaks.png", dpi=300)
```

Figure 6 combines the results of this approach with both models (2.1) and (2.2) and putting

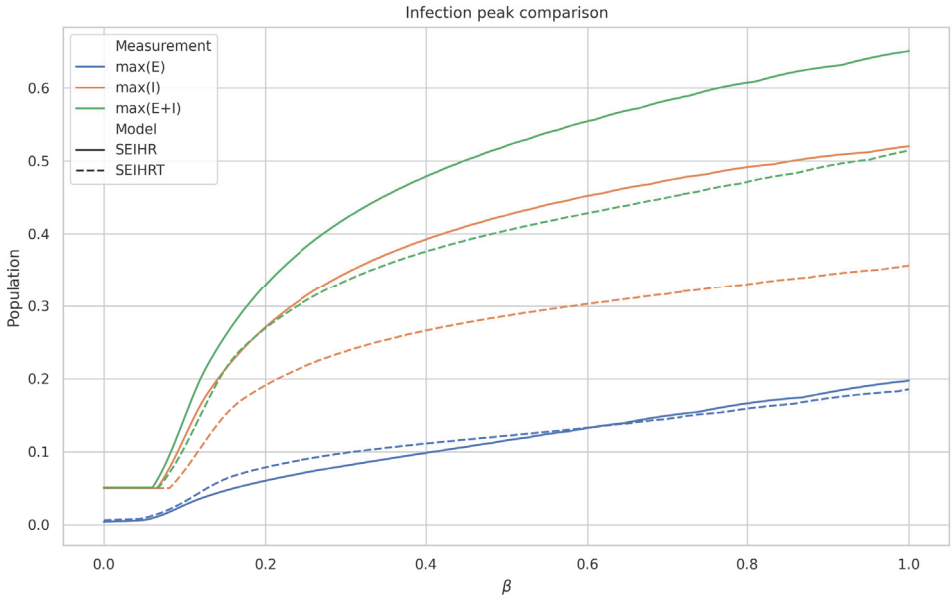

Figure 6: Infection peaks from varying infection rate β.

all them together in just one figure. It shows that as the β value increases, we would also see an increase in addicted individuals. Further, we can see that, for model SEIHRT (2.2), values are lower which shows the influence of the treatment.

Next, we applied a similar analysis to recovery parameters involved in our second model where we introduce treatment as a recovery mechanism. We observed the effects of changing the parameter η, in the same manner. We varied the values, and this time, we considered the infectious state, and the hospitalized state, since those two equations contain the η parameter, and observed the effect. Figure 7 shows that as η increases, we see a decrease in infections, thus showing that introducing treatment would truly help to reduce infections.

4.4 Parameter Estimation

The computational experiments conducted so far were based on assumed values of parameters from Table 1. In this section, we assume that the parameters including β for model SEIHR (2.1), and β and η for model SEIHRT (2.2) are not known. We describe two approaches to estimate these parameters.

4.4.1 Least-squares Approximation

In order to estimate these specific parameter values, we start by creating a dataset that we used to get the graphs in the previous section. Recall that β was preset to a value of 0.5. We first obtain a dataset by solving the forward problem for known parameter values. From this, we then extracted the data from the exposed ($E(t)$) compartment, since the

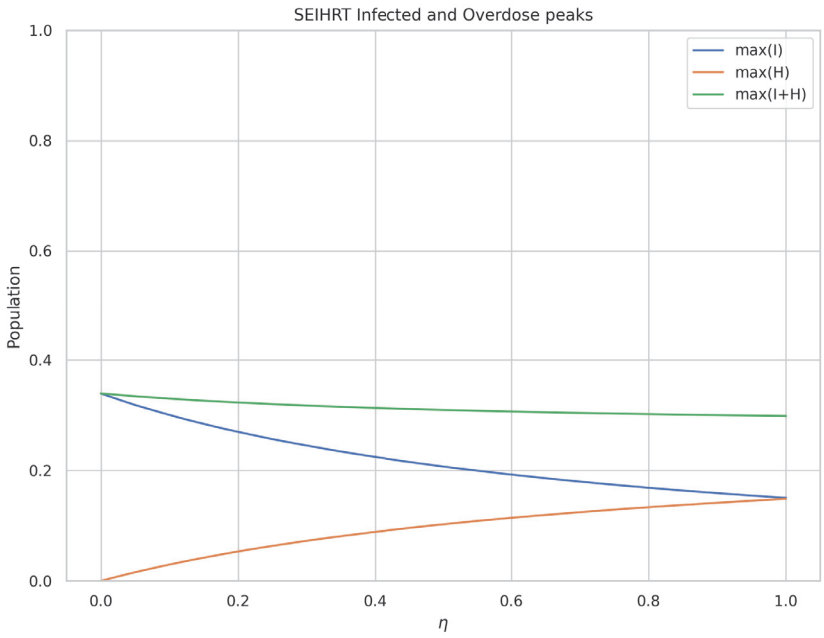

Figure 7: Infection and overdose peaks from varying η in model SEIHRT (2.2).

differential equation contains β in one of the terms. Next, a 5% noise was added to this dataset in order to simulate the behavior of real-world data. Let $\widehat{E}(t, \beta)$ be a numerical estimation of $E(t)$ for some known β, and t_i for $i = 1, \ldots, N_{\text{data}}$ a time grid. We want to find $\widehat{\beta}$ such that it is the solution of the following optimization problem

$$\min_{\beta} \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(\widehat{E}(t_i, \beta) - E(t_i) \right)^2 . \tag{4.1}$$

To evaluate the predictive capability of our model, $\widehat{E}(t, \beta)$ is estimated using the function `integrate.solve_ivp`. The least-square problem (4.1) is solved using another function from `SciPy`, called `optimization.curve_fit`. For model SEIHR (2.1), the estimated value was $\beta = 0.0499$ and the corresponding curve-fit with the data is shown in Figure 8.

Using least squares regression, β was predicted very accurately, and demonstrated that important parameters, such as the transmission rate in model SEIHR (2.1) can be predicted using our model on various data sets. Although this shows the performance of the least-squares methods, we needed to turn to more powerful prediction methods. It is known that least squares regression can get very slow very quickly, especially when trying to do multi-parameter estimation like estimating more than one parameter, for example, β and η for model SEIHRT (2.2). For this, we next introduce an alternate approach.

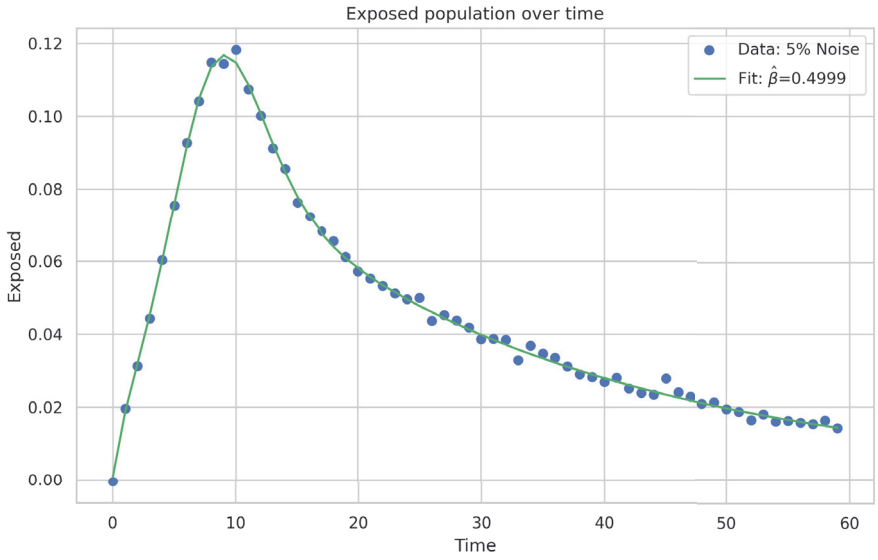

Figure 8: Susceptible sub-population fitting to estimate β for model SEIHR (2.1)

4.4.2 Physics-Informed Neural Networks

Next, we introduce a parameter identification approach called the Physics Informed Neural Networks (PINNs) to further validate the predictability of our models, and to predict additional parameters for model SEIHRT (2.2) reasonably well. PINNs differ from traditional neural networks by giving the network a system of governing equations [12], which in our case, would be our models, for data-driven learning. The network will obey the rules of this system of governing equations, and help to produce results that make sense in the terms of our models. Recently, Disease Informed Neural Networks (DINNs) was proposed to leverage the hidden physics of infectious diseases and infer the latent quantities of interest by approximating them using PINNs [16]. Such approaches can also be applied for the models proposed in this work.

For model (2.2), let $u(t;\lambda) = (S(t), E(t), I(t), H(t), R(t), T(t)) \in \mathbb{R}^6$ the function that we want to approximate through PINNs, and $\lambda = (\beta, \eta) \in \mathbb{R}^2$ the vector of parameters related to the dynamics of the model. The training data has been discretized as $\{t^j, u^j\}_{j=0}^{N_{\text{data}}}$, where N_{data} is the amount of training samples. The approximation is defined as $\widehat{u}(t;\lambda,\theta)$ where θ is the vector of learnable parameters defined by the architecture of the neural network used, for example, for a fully connected neural network θ corresponds to the concatenation of biases and weights. In practice, during the training process λ are also considered as learnable parameters using the loss function (4.2), a linear combination of differential residuals, initial conditions and training data.

$$\mathscr{L}(\widehat{\lambda},\widehat{\theta}) = \mathscr{L}_{\text{ode}}(\widehat{\lambda},\widehat{\theta}) + \mathscr{L}_{\text{ic}}(\widehat{\lambda},\widehat{\theta}) + \mathscr{L}_{\text{data}}(\widehat{\lambda},\widehat{\theta}) \tag{4.2}$$

where ω_{ode}, ω_{ic} and ω_{data} are the loss weights of the loss functions of the system of differential equations, initial conditions and training data, respectively. While the overall

loss function is decomposed in three parts, the loss function of differential residuals that helps to capture the physics from the differential equations [13] is described as:

$$\begin{aligned}\mathscr{L}_{\text{ode}}(\widehat{\lambda},\widehat{\theta}) = \omega_{\text{ode},1}\mathscr{L}_S(\widehat{\lambda},\widehat{\theta}) + \omega_{\text{ode},2}\mathscr{L}_E(\widehat{\lambda},\widehat{\theta}) + \omega_{\text{ode},3}\mathscr{L}_I(\widehat{\lambda},\widehat{\theta}) \\ + \omega_{\text{ode},4}\mathscr{L}_H(\widehat{\lambda},\widehat{\theta}) + \omega_{\text{ode},5}\mathscr{L}_R(\widehat{\lambda},\widehat{\theta}) + \omega_{\text{ode},6}\mathscr{L}_T(\widehat{\lambda},\widehat{\theta}),\end{aligned}$$

where

$$\begin{aligned}\mathscr{L}_S &= \left[\frac{\mathrm{d}S}{\mathrm{d}t} - \left(\Lambda - \frac{\beta SI}{N}(1-b) - \frac{\epsilon\beta SE}{N}(1-b) - \mu S\right)\right]^2 \\ \mathscr{L}_E &= \left[\frac{\mathrm{d}E}{\mathrm{d}t} - \left(\frac{\beta SI}{N}(1-b) + \frac{\epsilon\beta SE}{N}(1-b) - \sigma E + r_1 R - \mu E\right)\right]^2 \\ \mathscr{L}_I &= \left[\frac{\mathrm{d}I}{\mathrm{d}t} - (\sigma E - \gamma I - \eta I - \alpha I - \mu_1 I + r_2 H + r_3 T - \mu I)\right]^2 \\ \mathscr{L}_H &= \left[\frac{\mathrm{d}H}{\mathrm{d}t} - (\eta I - \omega H - \mu_2 H - r_2 H - \mu H)\right]^2 \\ \mathscr{L}_R &= \left[\frac{\mathrm{d}R}{\mathrm{d}t} - (\delta T + \gamma I + \omega H - r_1 R - \mu R)\right]^2 \\ \mathscr{L}_T &= \left[\frac{\mathrm{d}T}{\mathrm{d}t} - (\alpha I - r_3 T - \delta T - \mu T)\right]^2.\end{aligned} \tag{4.3}$$

The loss function corresponding to the data may be expressed as:

$$\mathscr{L}_{\text{data}}(\widehat{\lambda},\widehat{\theta}) = \sum_{i=1}^{6} \frac{\omega_{\text{data},i}}{N_{\text{data}}} \sum_{j=1}^{N_{\text{data}}} \left(u_i^j - \widehat{u}_i^j(\widehat{\lambda},\widehat{\theta})\right)^2,$$

and the loss function corresponding to the initial condition may be expressed as:

$$\mathscr{L}_{\text{ic}}(\widehat{\lambda},\widehat{\theta}) = \sum_{i=1}^{6} \omega_{\text{ic},i} \left(u_i^0 - \widehat{u}_i^0(\widehat{\lambda},\widehat{\theta})\right)^2$$

where u_i for $i = 1,\ldots,6$ corresponds each component of $u(t;\lambda)$, for example, u_1 corresponds to S, u_2 corresponds to E and so on until u_6 that corresponds to T. Similar for the weights ω, which are hyper-parameters of the framework. Algorithm 1 shows how to estimate $u(t;\lambda)$ and the parameters λ.

We begin by recreating the β estimation for model SEHIR (2.1), however, by using PINNs, there is no need to focus on one compartment. It takes into account the entire system of equations. We proceed to train with PINNs on the same dataset we used for least squares regression, with $\beta = 0.5$. Figure 9 shows the learning curve of this approach, where the horizontal axis corresponds to the iteration and the vertical axis to the $\widehat{\beta}$ estimation on each iteration of the training process. This approach estimates $\widehat{\beta} = 0.500046$, which corresponds to a relative error of 0.000092. However, the greatest strength of PINNs come with multi parameter estimation. Note that least squares is a great method for single parameter estimation, but it may be difficult to estimate many parameters. On

Algorithm 1: PINNs algorithm.

Input : Training Data $\{t^j, u^j\}$ where $j = 0, 1, \ldots, N_{\text{data}}$

Output $\widehat{u}$ and $\widehat{\lambda}$

:

Initialize $\widehat{\lambda}_0$ and $\widehat{\theta}_0$

Define time interval where the solution will be found.

Define loss function $\mathscr{L}(\widehat{\lambda}, \widehat{\theta})$, related to residual errors, initial conditions and training data.

Create a fully connected neural network with 1 neuron in the input layer and 6 neurons in the output layer (one per compartment).

Choose optimization hyper-parameters (e.g. Adam optimizer, learning rate and loss weights).

for $\texttt{iter} = 1, \ldots, \texttt{max\_iter}$ **do**

  Compute total loss $\mathscr{L}(\widehat{\lambda}_{\texttt{iter}-1}, \widehat{\theta}_{\texttt{iter}-1})$ using auto-differentiation for ODE residuals.

  Train neural network with optimizer algorithm and update $\widehat{\theta}_{\texttt{iter}-1}$ to $\widehat{\theta}_{\texttt{iter}}$.

  Get approximation $\widehat{u}_{\texttt{iter}}$ and $\widehat{\lambda}_{\texttt{iter}}$.

end

Return $\widehat{u}_{\texttt{max\_iter}}$ and $\widehat{\lambda}_{\texttt{max\_iter}}$.

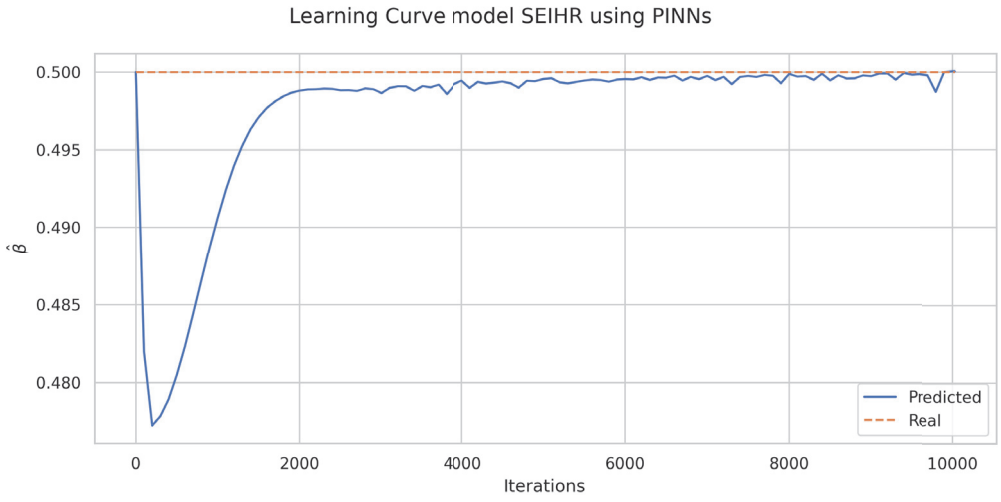

Figure 9: Learning curve of β for model SEIHR (2.1) using PINNs.

the other hand, PINNs is a flexible framework where parameters from the model are also trainable parameters.

Now, it is straightforward to apply PINNs to SEIHRT (2.2) model for predicting β and the treatment parameter η. Figure 10 shows the learning curves for β and η, showing that even around 10000 iterations are enough to get a good accuracy. PINNs predict $\widehat{\beta} = 0.492374$ and $\widehat{\eta} = 0.142727$, which represents relatives errors of 0.015251 and 0.000908,

respectively. The results shown that our PINNs framework is predictable, and can work with varying data sets.

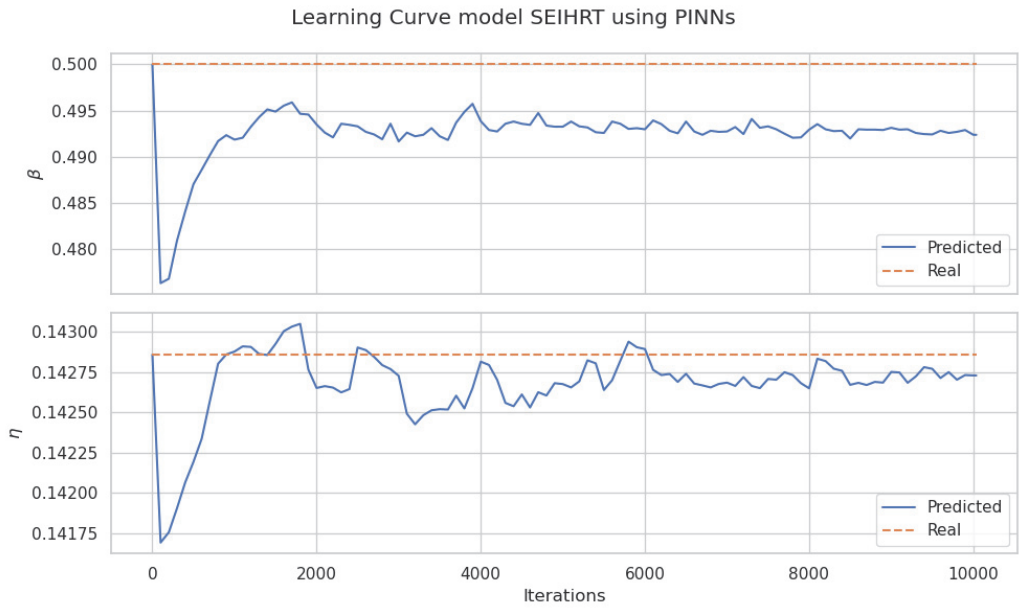

Figure 10: Learning curve of β and η for model SEIHRT (2.2) using PINNs.

The complete code for PINNs application described here is publicly available on https://github.com/AdanXBaca/detox_journey.

5 Discussion and Conclusion

In this work, we develop a mathematical model to examine the progression of addiction during the detoxification process. The rationale behind this endeavor is multifaceted. Initially, the surge in drug-related fatalities and dependencies following the COVID-19 outbreak underscores the urgency of treating addiction as akin to an infectious ailment. Secondly, leveraging compartmental models governed by differential equations, commonly utilized in analyzing infectious diseases, offers a viable approach to conceptualizing addiction dynamics. Lastly, this research underscores the potential of differential equation-based mathematical models in shedding light on societal issues like addiction.

The proposed work presents a SEIHR (Susceptible, Exposed, Infected, Hospitalized, Recovered) differential equation system to capture the nonlinear dynamics among drug users. The basic reproduction number for the model is derived using the Next-Generation matrix approach. Additionally, numerical simulations corroborate the model's efficacy by validating it against synthetic COVID-19 data. The SEIHR model is also enhanced to incorporate a treatment compartment and the numerical experiments show that this addition can control the addiction better.

Following direct simulation for fixed parameters, some of the model parameters were also estimated using traditional nonlinear least squares approach as well as through the

integration of a Physics Informed Neural Network methodology which facilitates efficient parameter estimation within the model. The numerical results showed that PINNs is a robust and reliable technique for predicting multiple parameters in the model.

Another important contribution includes the exposition of literate programming that is a useful approach to enhance instructional practices and for students to learn the implementation of the models better by simultaneously programming as they build their content knowledge.

Acknowledgement(s)

This work is partially supported by the National Science Foundation grants DMS-2230117 and Simons Foundation award number 1036702.

References

[1] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*, volume 2. Springer, 2012.

[2] Wandi Ding, Ryan Florida, Jeffery Summers, Puran Nepal, and Ben Burton. Experience and lessons learned from using simiode modeling scenarios. *PRIMUS*, 29(6): 571–583, 2019.

[3] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[4] Donald Ervin Knuth. Literate programming. *The computer journal*, 27(2):97–111, 1984.

[5] Maia Martcheva. *An introduction to mathematical epidemiology*, volume 61. Springer, 2015.

[6] Maranya M Mayengo, Gabriel M Shirima, Snehashish Chakraverty, Moatlhodi Kgosimore, Padmanabhan Seshaiyer, and Carmen Caiseda. Mathematical modeling of the dynamics of health risks associated with alcoholism in tanzania: a literature review. *Commun. Math. Biol. Neurosci.*, 2020:Article–ID, 2020.

[7] Chris McCarthy, Ellen Swanson, and Brian Winkel. Special issue of primus: Modeling approach to teaching differential equations, 2019.

[8] Nedialko S Nedialkov. Implementing a rigorous ode solver through literate programming. *Modeling, Design, and Simulation of Systems with Uncertainties*, pages 3–19, 2011.

[9] John Boscoh H Njagarah and Farai Nyabadza. Modelling the role of drug barons on the prevalence of drug epidemics. *Mathematical Biosciences & Engineering*, 10(3): 843–860, 2013.

[10] Alonso Ogueda-Oliva and Padmanabhan Seshaiyer. Literate programming for motivating and teaching neural network-based approaches to solve differential equations. *International Journal of Mathematical Education in Science and Technology*, 55 (2):509–542, 2024. doi: 10.1080/0020739X.2023.2249901.

[11] Comfort Ohajunwa, Carmen Caiseda, and Padmanabhan Seshaiyer. Computational modeling, analysis and simulation for lockdown dynamics of covid-19 and domestic violence. *Electronic research archive*, 2022.

[12] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991.

[13] Maziar Raissi, Niloofar Ramezani, and Padmanabhan Seshaiyer. On parameter estimation approaches for predicting disease transmission through optimization, deep learning and statistical inference methods. *Letters in biomathematics*, 6(2):1–26, 2019.

[14] Padmanabhan Seshaiyer. Leading undergraduate research projects in mathematical modeling. *PRIMUS*, 27(4-5):476–493, 2017.

[15] Padmanabhan Seshaiyer and Pavel Solin. Enhancing student learning of differential equations through technology. *International Journal for Technology in Mathematics Education*, 24(4), 2017.

[16] Sagi Shaier, Maziar Raissi, and Padmanabhan Seshaiyer. Data-driven approaches for predicting spread of infectious diseases through dinns: Disease informed neural networks. *Letters in Biomathematics*, 9(1):71–105, 2022.

[17] Abhinav Tandon, Arti Mishra, and Sankha Banerjee. Transmission dynamics of depression in socially connected population: a nonlinear modeling study. *International Journal of Modeling, Simulation, and Scientific Computing*, 12(05):2150041, 2021.