SEIQR Neural Network Optimal Control
README

July 31, 2025

1 Overview

This document provides instructions for using the Python script `SEIQR_NeuralControl.py`, which implements a deep neural network (DNN) optimal control problem for a Susceptible-Exposed-Infected-Quarantined-Recovered (SEIQR) epidemic model. The model uses neural network controls to minimize the following cost functional:

$$\min_{\theta \in \Theta} \left\{ J(x, \theta) = \int_0^T I(t)\, dt + \sum_{i=1}^{4} \frac{1}{2} \int_0^T c_i \mathcal{N}_i^{\theta_i}(t)^2\, dt \right\} \tag{1}$$

subject to the SEIQR differential equations:

$$\begin{aligned}
\frac{dS(t)}{dt} &= \Lambda - \beta S(t)I(t) - (\mathcal{N}_1^{\theta_1}(t) + d)S(t) + \delta R(t), \\
\frac{dE(t)}{dt} &= \beta S(t)I(t) - (\mathcal{N}_2^{\theta_2}(t) + a + d)E(t), \\
\frac{dI(t)}{dt} &= aE(t) - (\mathcal{N}_3^{\theta_3}(t) + \mathcal{N}_4^{\theta_4}(t) + r + m + d)I(t), \\
\frac{dQ(t)}{dt} &= \mathcal{N}_4^{\theta_4}(t)I(t) - (\eta + d)Q(t), \\
\frac{dR(t)}{dt} &= \mathcal{N}_1^{\theta_1}(t)S(t) + \mathcal{N}_2^{\theta_2}(t)E(t) + (\mathcal{N}_3^{\theta_3}(t) + m)I(t) + \eta Q(t) - (\delta + d)R(t),
\end{aligned} \tag{2}$$

where $\mathcal{N}_i^{\theta_i}(t) \in [0, 1]$ are neural network controls parameterized by θ_i.

2 Prerequisites

To run the script, ensure the following are installed:

- **Python**: Version 3.8 or higher.
- **Libraries**:
  - `numpy`: For numerical computations.
  - `matplotlib`: For plotting results.
  - `tensorflow`: For neural network implementation.
  - `scipy`: For numerical integration.
- **Optional**: A GPU with CUDA support for faster TensorFlow computations.

3 Installation

Follow these steps to set up the environment:

1. **Install Python**: Download from `https://www.python.org/downloads/`.
2. **Create a Virtual Environment** (recommended):

```
python -m venv seIQR_env
source seIQR_env/bin/activate  # On Windows: seIQR_env\Scripts\activate
```

3. **Install Dependencies**:

```
pip install numpy matplotlib tensorflow scipy
```

4. **Verify TensorFlow GPU Support** (if using GPU):

```
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

4 Usage

4.1 Running the Script

1. **Download the Script**: Save `SEIQR_NeuralControl.py` to your working directory.
2. **Execute the Script**:

```
python SEIQR_NeuralControl.py
```

3. **Outputs**: The script generates three plots:
   - **State Variables**: Compares controlled and uncontrolled trajectories of S, E, I, Q, R (normalized by population).
   - **Controls**: Shows the four neural network controls $\mathcal{N}_i(t)$.
   - **Loss Convergence**: Displays the loss $J(x, \theta)$ over training epochs.

   Console output includes the loss value every 10 epochs.

4.2 File Structure

- `SEIQR_NeuralControl.py`: Main Python script.
- `README.tex`: This document (compile to PDF).

5 Customization

5.1 Model Parameters

Edit the parameters in `SEIQR_NeuralControl.py`:

```
N = 1.40005e9  # Population
Lambda = 0.01 * N  # Birth rate
beta = 0.3  # Transmission rate
d = 0.01  # Natural death rate
delta = 0.05  # Return rate from R to S
a = 0.2  # Progression rate from E to I
r = 0.1  # Recovery rate
m = 0.05  # Disease-induced death rate
eta = 0.1  # Quarantine removal rate
c = [1.0, 1.0, 1.0, 1.0]  # Control cost coefficients
T = 20  # Time horizon
dt = 0.1  # Time step
S0 = 0.99 * N  # Initial susceptible
E0 = 0.01 * N  # Initial exposed
I0 = 1000  # Initial infected
Q0 = 0  # Initial quarantined
R0 = 0  # Initial recovered
```

5.2 Neural Network Architecture

Modify the neural network model:

```
model = Sequential([
    Dense(64, input_dim=1, activation='relu'),  # Input: time t
    Dense(64, activation='relu'),
    Dense(4, activation='sigmoid')  # Output: 4 controls in [0, 1]
])
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
```

5.3 Training

Adjust the training loop:

```
epochs = 50  # Number of training epochs
```

5.4 Observed Data

To fit observed data, modify the `compute_loss` function to include a data-fitting term (e.g., mean squared error).

6 Notes

- **Numerical Integration**: The Euler method is used for training simplicity. For accuracy, use `scipy.integrate.odeint` in `simulate_with_controls`.
- **Performance**: GPU accelerates training. Adjust `epochs` or `dt` for CPU usage.

- **Control Inputs**: The neural network uses time t. For state-dependent controls, set input dimension to 5 ($[S, E, I, Q, R]$).
- **Convergence**: Adjust learning rate or epochs if loss oscillates.

7 Troubleshooting

- **TensorFlow Errors**: Ensure TensorFlow compatibility (e.g., version 2.10 for Python 3.8–3.10).
- **Memory Issues**: Reduce `epochs` or increase `dt`.
- **No Plots**: Verify `matplotlib` installation and graphical output support (e.g., use Jupyter Notebook).

8 License

This project is for educational purposes and not licensed for commercial use.

9 Contact

For issues, contact the maintainer or open a repository issue.

10 Compiling to PDF

To generate the PDF, compile this file using a LaTeX compiler:

```
pdflatex README.tex
```

Required packages: `amsmath`, `amssymb`, `geometry`, `listings`, `hyperref`, `xcolor`, `noto`. Use Overleaf or a local LaTeX distribution (e.g., TeX Live, MiKTeX).