University of Connecticut

ENGR 5315

MEng Capstone Project

Nonlinear Optimal Control of the COVID-19 Pandemic in Connecticut

Eugene Ng

August 19, 2020

Acknowledgements

Many thanks to Dr. Abhishek Dutta for his many suggestions and continued interest in my life in our virtual meetings.

Much gratitude to Nimit Sohoni for answering relentless questions on nonlinear systems of equations.

Contents

List of Symbols

α	Vaccination Rate
β	Exposure/Infection Rate
γ	Removal/Recovery Rate
δ	Natural Birth Rate
μ	Natural Death Rate
σ	Incubation Rate
b	Available Hospital Beds
u	Available ICU Beds
c	Case Fatality Rate
h	Hospitalization Rate
N	Population of CT
m	Symptomaticity Ratio
$R0$	Basic Reproduction Rate
t	Time

Abstract

COVID-19 is an infectious disease which has resulted in an ongoing pandemic. This paper attempts to model the spread of the disease in the state of Connecticut, and formulate a nonlinear closed-loop feedback controller to minimize the number of infected. A model is first developed and studied. Difficulties in applying the controller - frequency of policy changes, lagging data, an unpredictable population - are explored.

1 Background

The Coronavirus disease 2019 (COVID-19) has rocked the world since it's first identification in Wuhan, China in December 2019. The ongoing pandemic has caused roughly 165,000 deaths in the USA as of August 15, 2020, roughly 4,500 of which are from Connecticut, our study of choice. The initial outbreak and subsequent attempts to control the spread of the virus are the subject of much scrutiny and criticism - missing data, lack of testing, and transparency are all issues. This pandemic and the American reaction to it can be considered the first of its kind in the US in terms of new responses and lasting effects - such as N95 mask usage and social distancing - both of which are now new standard terms in the American vernacular. The problem is certainly one that is time-sensitive and incredibly impactful.

There are several difficulties with implementing a closed-loop control, the first of which is that there are inherent uncertainties when applying such a control to an unpredictable population. In addition, the lack of a coordinated response, a generally accepted delay in between COVID infection and reporting, and the unfortunate lack of adequate testing supplies, all serve to exacerbate the issue. Some experts predict the actual number of infected is ten times what is reported. Finally, only certain metrics associated with the virus are tracked (i.e. number of active infected and recovered), which makes parameter estimation difficult.

Similar studies have attempted to simulate the spread of COVID-19 with the SEIR model. Olivier and Craig [10], for example, model the virus with a vaccination rate and assume the rate of infection is a decreasing function of time. Castilho [4] simulates several different rates of infection to determine which path is most likely. However, there is little in the way of attempting to truly control the model to drive it towards a wanted state with specific state policy changes - the closest is Tsay et. al [16].

2 Objective

Despite these challenges and unknowns, this paper will attempt to apply a nonlinear optimal control strategy to the system existing in the state of Connecticut. So far,

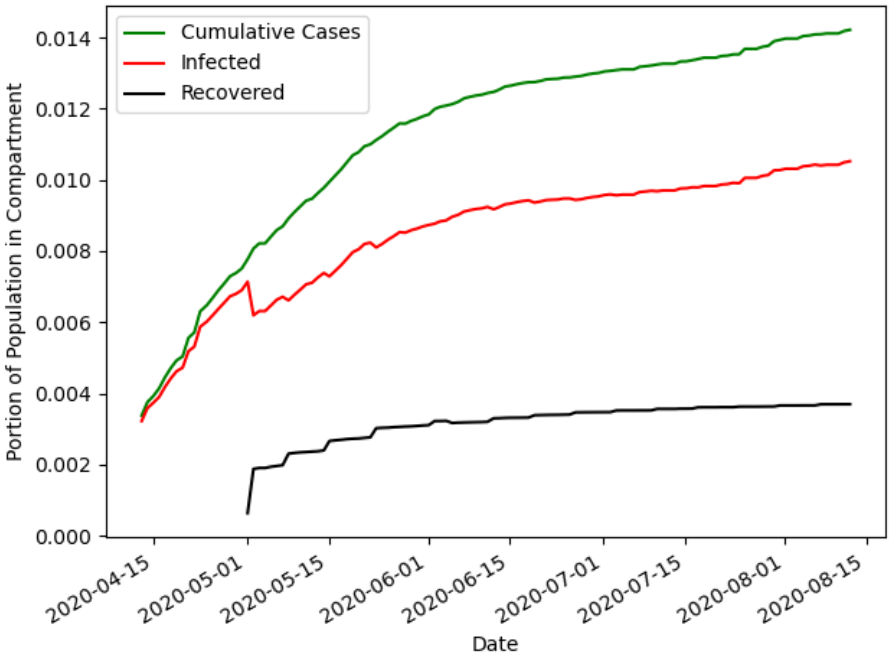


Figure 1: Current COVID-19 status in the state of Connecticut

the state has done relatively well in controlling the spread of the virus but stands to benefit from additional knowledge.

In this paper the system shall be modeled, and its open and closed loop characteristics will be studied. The initial study will be used to estimate model parameters in Connecticut, set target variables, and choose a suitable method of control. The objective will be focused on minimizing hospital visits and ultimately driving down the number of infected, with a consideration for the state's economy and it's population's well-being.

3 System Modeling and Analysis

3.1 System Model

We can begin our study by looking at the simplest compartmental model in epidemiology - the SIR model. It consists of three compartments, normalized against the entire population, N:

- S(t): the ratio of **S**usceptible individuals
- I(t): the ratio of **I**nfected individuals
- R(t): the ratio of **R**emoved individuals. This bucket encompasses both recov-

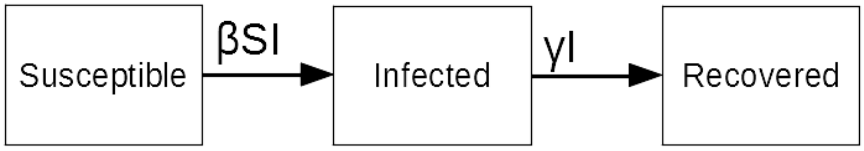


Figure 2: Compartments in the SIR Model. Additional compartments, such as exposed, deceased, and vaccinated are often added.

ered and deceased people.

In total, this is a nonlinear time-invariant, non-autonomous system, described by the following set of differential equations.

$$\dot{S}(t) = \beta S(t) I(t) \tag{1}$$
$$\dot{I}(t) = \beta S(t) I(t) - \gamma I(t) \tag{2}$$
$$\dot{R}(t) = \gamma I(t) \tag{3}$$

It has a unique, simplifying characteristic: by knowing R, we know S.

$$\begin{aligned} \frac{dR}{dt} &= \gamma I \\ dR &= \gamma I dt \\ R &= \gamma \int I dt + c_1 \\ \int I dt &= \frac{R}{\gamma} + c_2 \end{aligned}$$

We substitute this result in equation 2, with initial conditions $S_0 = 1$ and $R_0 = 0$:

$$\begin{aligned} \frac{dS}{dt} &= -\beta S I \\ \frac{dS}{S} &= -\beta I dt \\ \int \frac{dS}{S} &= -\beta \int I dt \\ S &= \exp\left(\frac{-\beta R}{\gamma}\right) \end{aligned}$$

We have effectively turned the three-dimensional system into a two-dimensional system. This allows us to study the phase portrait in many different ways. For example, Figure 3 shows the Susceptible vs Infected phase portrait defined by the differential equations:

$$\dot{x}_1 = \dot{S} = -\beta SI \tag{4}$$
$$\dot{x}_2 = \dot{I} = \beta SI - \gamma I \tag{5}$$

The phase portrait in Figure 3 shows, intuitively, initial conditions being driven towards the lower left hand corner of the graph, indicating that as the number of susceptible individuals drop, the number of infected also tends towards zero. A caveat: large sections of the phase portrait are not valid in the feasible region considering the constraints upon the system as a whole, namely, $S + I + R = 1$. Still the behavior of the phase portrait is as expected. The behavior of the phase portrait in Figure 4 is similar - as the number of infected rises, the number of recovered is driven upwards as well.

Of course, COVID-19 is much more complicated than the simple SIR model. We can expand our model to include both birth and death rates, δ and μ, respectively, as well as add an additional bucket, **Exposed**, which defines the ratio of individuals exposed to the virus and who may be carriers but may not show symptoms.

$$\dot{S}(t) = \delta - \beta S(t)I(t) - \mu S(t) \tag{6}$$
$$\dot{E}(t) = \beta S(t)I(t) - \sigma E(t) - \mu E(t) \tag{7}$$
$$\dot{I}(t) = \sigma E(t) - \gamma I(t) - \mu I(t) \tag{8}$$
$$\dot{R}(t) = \gamma I(t) - \mu R(t) \tag{9}$$

Typically, δ and μ are so small (and similar) over the time span of the period of study (months/years) that they are effectively negligible. Nevertheless, they are useful to study the system and can help us define a feasible region for the model, which is simply:

$$\frac{dN}{dt} = \frac{dS}{dt} + \frac{dE}{dt} + \frac{dI}{dt} + \frac{dR}{dt} \geq 0$$
$$\frac{dN}{dt} = \delta - \mu N \geq 0$$
$$\delta \geq \mu N$$

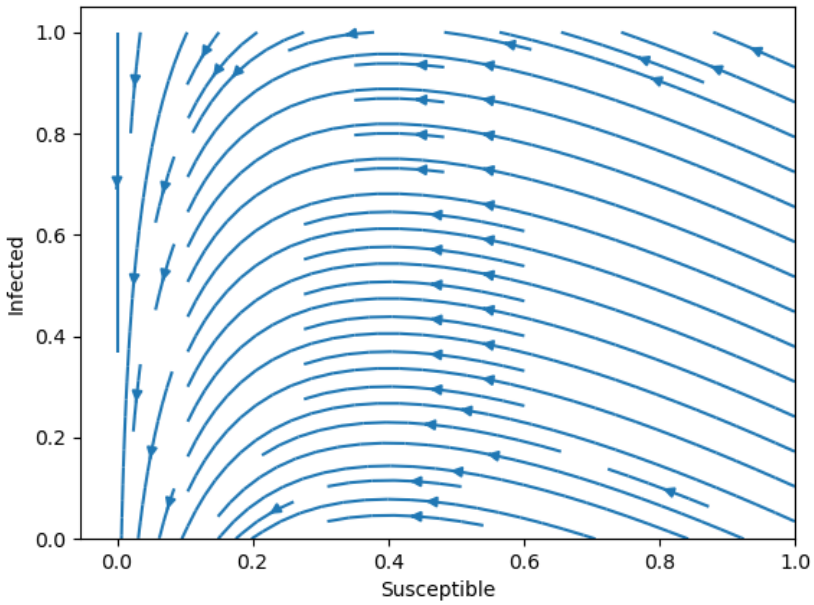


Figure 3: Two Dimensional Phase Portrait of Susceptible and Infected

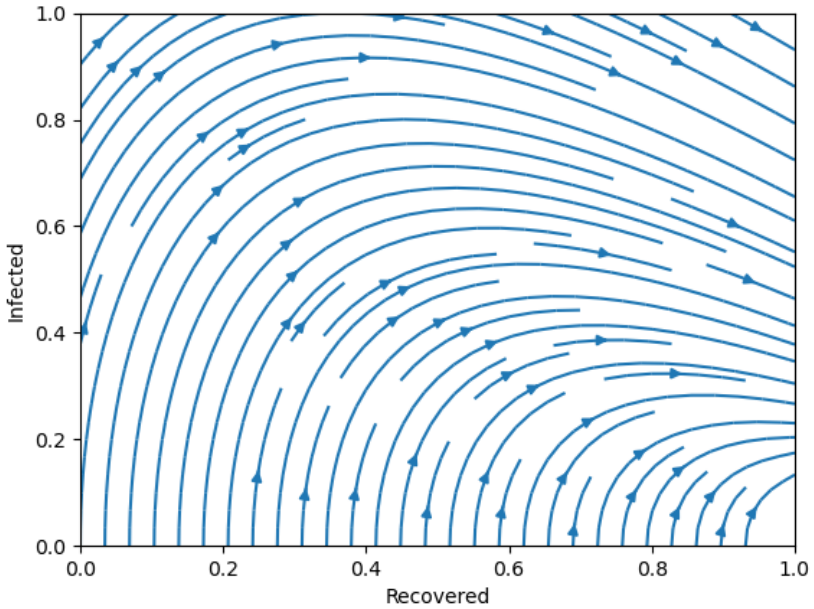


Figure 4: Two Dimensional Phase Portrait of Recovered and Infected

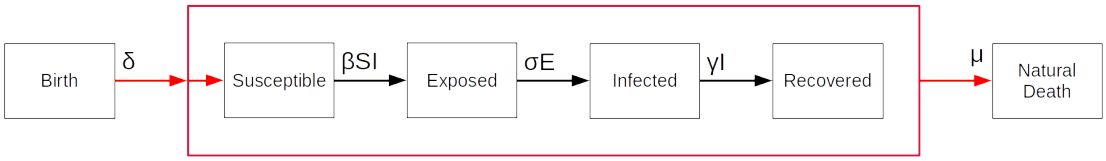


Figure 5: Compartments in the SEIR Model

Alternatively, if $\delta = \mu$ (birth and death rates are equal and negligible),

$$N \leq 1 \tag{10}$$

3.2 Parameter Estimation

There are a multitude of issues that make modeling an epidemic difficult, and this is largely reflected in already existing models of the coronavirus. Early on, different models suggested a total death count of between 200,000 and 2.2 million [8], representing a very large spread. Common concerns include trustworthiness of the source data, under-reporting data, consistency across datasets, and testing rate and availability. In addition, inconsistencies across counties and cities mean that localized behavior and laws affect the spread of the virus in different ways in different places. Any one or a combination of these variables could skew data in large ways, resulting in false conclusions.

The main sources of data used in this paper are the New York Times [14] and Johns Hopkins University [3]. In addition, we can estimate initial guess values for SEIR parameters based on available knowledge from previous papers or public knowledge. For example, JHU estimates a median incubation period for the virus of 5 days, with a maximum of 14, which can be used as an initial starting guess for σ. Table 1 shows a complete list of initial guesses, as well as constants for the state of CT.

In estimating these parameters, we must be cognizant of what Tsay [16] calls "hidden states" - those variables which are not measured in practice. The most glaring issue is that the number of asymptomatic cases is almost impossible to determine, since those individuals are unlikely to get tested but yet would still be considered part of the Infectious compartment.

3.2.1 Necessary Assumptions

There are a number of assumptions that must be made about the data and how it is interpreted in the SEIR model.

[1]Without interventions. See 3.2.3

Name	Parameter	Initial Guess	Source
Average Incubation Period	σ	0.2	Johns Hopkins University [9]
Basic Reproduction Number [1]	R0	2.5	Centers for Disease Control [13][15]
Case Fatality Rate	c	0.022	University College London [18]
CT Hospital Beds	b	8798	American Hospital Directory [6]
CT ICU Beds	u	674	COVID Act Now [2]
CT Natural Birth Rate	δ	2.70×10^{-5}	Centers for Disease Control [11]
CT Natural Death Rate	μ	2.403×10^{-5}	Centers for Disease Control [7]
CT Population	N	3,565,287	United States Census Bureau [17]
Rate of Exposure	β	0.175	$R0 = \beta/\gamma$ [Ref Sec. 3.2.3]
Rate of Hospitalization	h	0.0014	Centers for Disease Control [13]
Rate of Removal	γ	0.07	World Health Organization [12]
Symptomaticity Ratio	m	0.5	Diamond Princess Cruise Ship [8]

Table 1: Initial Guesses for SEIR Parameters, and Other Constants

1. **Age, Gender, and Race are not contributing factors.**

In the SEIR model, each variable is a function of time and time only. In other words, $x = x(t)$. This is particularly dangerous because of the different rates at which different groups contract and die from the virus. For example, it is well known that younger individuals are more likely to contract the virus, while older, retirement-age individuals are more likely to die from it [19]. The SEIR model as a whole makes no such distinction between individuals of different ages, and while the effect is very pronounced when looking at just age, the same assumption applies towards gender and race as well.

2. **Testing is sufficient to capture all cases. Testing does not lag.**

Testing strategies and patterns must be robust. For example, an increasing number of cases might suggest an actual increase in virus spread, *or* it could represent a simple increase in testing - it would be difficult to distinguish between the two without also looking at hospital admission rates. Failure to account for these testing strategies can skew comparisons and ruin conclusions [13]. Testing must first be sufficient enough to capture nearly all cases of COVID-19 - if it does not, the data being input into the model is false. At the time of this writing, Connecticut's positive test rate is very low [2], suggesting widespread and aggressive testing to detect most new cases. Finally, publicly available data seem to suggest that there is a large portion of the population that is asymptomatic and unaware that they are carriers for the disease. It is unlikely that these individuals will get tested - and yet they are still spreading the disease [8].

3. **Implementation of control is instantaneous. There is no time lag between compartments.**

Typically it might take several days to implement policy changes that would affect

the spread of coronavirus. News and mandates would have to be communicated to the populace and the population would have to adjust their behavior accordingly. This would take time and coordination. The control system, however, assumes that any changes to the control would happen immediately.

3.2.2 Parameter Refinement

Data for the number of infected in the state goes back only until March 3rd, 2020, and only the first month of data is used, as the infected curve begins to flatten out due to executive measures taken by CT's state government (as measured by the daily percent change in cases approaching less than 3%). The same caveats as explained in Section 3.2.1 apply - as, at that time, testing was not as available or widespread.

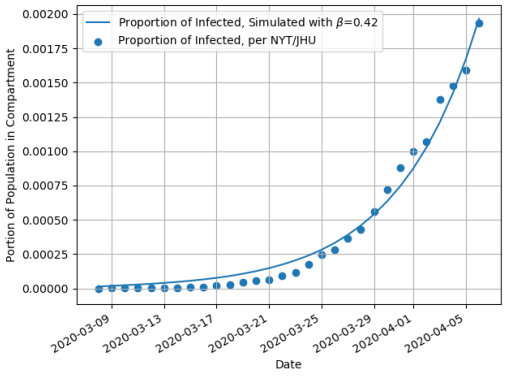
A slightly different method is used for parameter estimation than in Castilho [4], or Tsay [16]. There are only three parameters to be determined for the control model: β, σ, and γ. The values for σ and γ given by other literature are accepted and only β refinement is sought, which we are interested in controlling. It is assumed that initial conditions on March 3rd do *not* start at 0, so the initial number of exposed and infected on March 3rd must also be estimated. In this manner the SEIR model is simulated for various values of β, e_0, and i_0. The minimum error state is found as follows (See Figure 6):

$$\begin{aligned} \beta &= 0.42 \\ e_0 &= 100 \\ i_0 &= 50 \end{aligned}$$

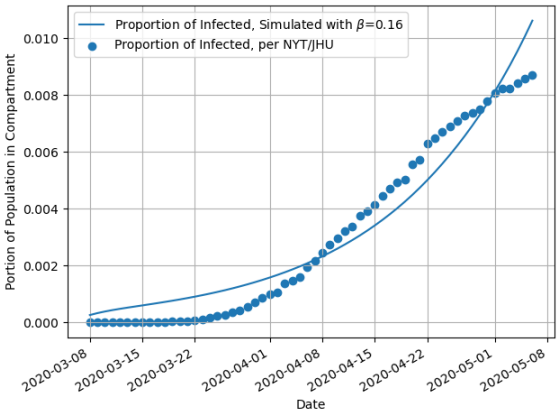
In addition, we wish to know what effect the current restrictions in CT have to do with limited the spread of COVID. To that end, the first 60 and 150 days are also used for parameter refinement (these date ranges include the start and continuation of restrictions). In these cases, β is found to be 0.16 and 0.11, respectively, although the comparison starts to degrade, especially at 150 days.

3.2.3 Basic Reproductive Number

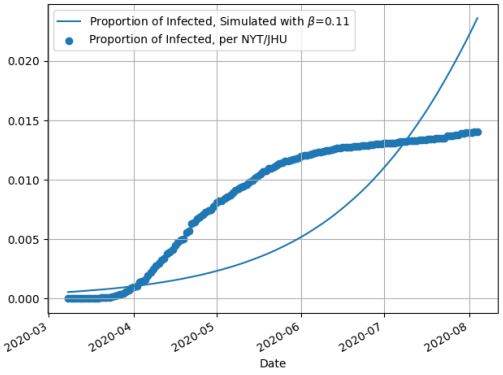
The basic reproductive number bears special importance; as a descriptive number it directly relates to the rate spread of the virus. It can be thought of as the expected number of cases generated by one starter case - therefore, if $R0 < 1$, then the rate of spread is decreasing. If $R0 > 1$, the infection will spread. We can derive $R0$ from the SEIR model by comparing the two equilibriums in the SEIR model (Equations 18 and 21) as shown:



(a) Infection Rate in the first 30 days of available data



(b) Infection Rate in the first 60 days of available data



(c) Infection Rate in the first 150 days of available data

Figure 6: Parameter Refinement at 30, 60, and 150 days yield different values for rate of spread, β: 0.42, 0.16, and 0.11, respectively. This shows that CT was able to reduce the infection rate dramatically within the first months of the start of the infection.

$$\frac{(\mu+\sigma)(\mu+\gamma)}{\beta\sigma} \geq \frac{\delta}{\mu}$$
$$\frac{\delta\beta\sigma}{\mu(\mu+\gamma)(\mu+\sigma)} \leq 1$$

This leads to Equation 11

$$R0 = \frac{\delta\beta\sigma}{\mu(\mu+\gamma)(\mu+\sigma)} \tag{11}$$

And, if δ and μ are equal and negligible:

$$R0 = \frac{\beta}{\gamma} \tag{12}$$

Since $R0$ is a function of β, β will be one of our primary control variables. If we set $R0 = 1$ and $\gamma = 0.07$, this relationship allows us to determine what value of β is ideal to limit the rate of spread.

$$\beta = 0.07 \tag{13}$$

In other words, β needs to be lower than 0.07 in order to eradicate the virus over time.

3.3 System Stability

Lyapunov's indirect, or linearization, method suggests that the stability properties of a nonlinear system in the close vicinity of an equilibrium point are essentially the same as those of its linearized approximation. We can derive multiple equilibrium points, each pointing to a specific scenario.

3.3.1 Linearization

The SEIR model can be linearized in a state space format by taking the Jacobian around the fixed points. Let

$$
\begin{aligned}
&\vec{z} = [S - S_e, E - E_e, I - I_e, R - R_e]^T, \\
&\vec{v} = \beta \\
&\text{ and} \\
&\dot{\vec{z}} = J\vec{z} + K\vec{v}
\end{aligned}
\tag{14}
$$

We also define a linear output matrix $\vec{y}$ such that

$$
\vec{y} = \vec{z} \tag{15}
$$

Thus,

$$
J = \begin{bmatrix} -\beta I & 0 & -\beta S & 0 \\ \beta I & -\sigma & \beta S & 0 \\ 0 & \sigma & -\gamma & 0 \\ 0 & 0 & \gamma & 0 \end{bmatrix} \tag{16}
$$

and

$$
K = \begin{bmatrix} -SI & SI & 0 & 0 \end{bmatrix}^T \tag{17}
$$

3.3.2 Disease-Free Equilibrium

Disease-free equilibrium (certainly a good thing) occurs when the number of exposed individuals is 0 (that is, $E = 0$). Based on the system model, this will cause the **I** and **R** buckets to also be equal to 0. That leaves us with Equation 18 and the fixed point Equation 19.

$$
\begin{aligned}
\frac{dS}{dt} &= 0 = \delta - \mu S \\
S_e &= \frac{\delta}{\mu} \\
S_e &= 1,\ \delta = \mu
\end{aligned}
\tag{18}
$$

Therefore the fixed point is

$$
P_0 = (\frac{\delta}{\mu}, 0, 0, 0) \tag{19}
$$

This is a trivial solution and again is easily intuited. If there are no individuals infected, then there is no disease to spread, and the population grows according to the birth/death rate.

The Jacobian around this fixed point is given by

$$J = \begin{bmatrix} 0 & 0 & -\beta & 0 \\ 0 & -\sigma & \beta & 0 \\ 0 & \sigma & -\gamma & 0 \\ 0 & 0 & \gamma & 0 \end{bmatrix}$$

Assuming that β, σ, and γ are positive, this Jacobian has two sets of eigenvalues and eigenvectors:

$$\begin{aligned} \lambda &= \pm 0 \\ \lambda &= \pm \frac{1}{2}\left(\sqrt{4\beta\sigma + \gamma^2 - 2\gamma\sigma + \sigma^2} - \gamma - \sigma\right) \end{aligned} \tag{20}$$

Using the values derived in Section 3.2, we find that all the eigenvalues are real, with one positive and one negative, suggesting an unstable saddle node. This would mean that any perturbation (any introduction of a virus) would cause instability.

3.3.3 Endemic Equilibrium

Disease-free equilibrium is relatively easy to study; there also exists an equilibrium point that is endemic. In this instance, E being nonzero drives the following equations:

$$\begin{aligned} S_e &= \frac{(\mu+\sigma)(\mu+\gamma)}{\beta\sigma} \\ E_e &= \frac{\delta}{\sigma+\mu} - \frac{\mu(\mu+\gamma)}{\beta\sigma} \\ I_e &= \frac{\delta\sigma}{(\mu+\gamma)(\mu+\sigma)} - \frac{\mu}{\beta} \\ R_e &= \frac{\gamma\delta\sigma}{\mu(\mu+\gamma)(\mu+\sigma)} - \frac{\gamma}{\beta} \end{aligned} \tag{21}$$

If the birth and death rates are treated as negligible and equal, this will lead to Equation 22.

$$P_0 = (\frac{\gamma}{\beta}, 0, 0, 1 - \frac{\gamma}{\beta}) \tag{22}$$

Note that E ends up being equivalent to zero despite the initial opening statement for endemic equilibrium. Essentially, this fixed point represents a point where the virus might have existed and a recovered population came into being, but at some point the virus was eradicated and no new cases exist.

Given our estimated parameters, this would put this fixed point at

$$P_0 = (0.167, 0, 0, 0.833)$$

The Jacobian around this fixed point is given by

$$J = \begin{bmatrix} 0 & 0 & -\gamma & 0 \\ 0 & -\sigma & \gamma & 0 \\ 0 & \sigma & -\gamma & 0 \\ 0 & 0 & \gamma & 0 \end{bmatrix}$$

This fixed point has three eigenvalues and eigenvectors:

$$\begin{aligned} \lambda &= \pm 0 \\ \lambda &= -\gamma - \sigma \end{aligned} \tag{23}$$

Similarly, all eigenvalues are real, with two zero and one negative. This suggests that the equilibrium point is an unstable sink node - any perturbation (a re-introduction of the virus or a new virus) would lead to instability.

3.3.4 Controllability and Observability

A completely parallel theory on controllability and observability to linear systems is not feasible, but the linearized system can similarly be studied to give information about controllability and observability in the vicinity of equilibria. Using the state-space defined in 14 around the fixed points, we find the controllability and observability matrices:

$$\mathrm{Co}_{DFE} = \begin{bmatrix} 0 \end{bmatrix}_{4x4} \tag{24}$$

$$\mathrm{Ob}_{DFE} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 6.93 \times 10^{-18} & 0 \\ 0 & 0 & 3.47 \times 10^{-18} & 0 \end{bmatrix} \tag{25}$$

Neither matrices are full rank, so the linearized systems is neither controllable (we cannot reach any final state from any initial state) nor is it fully observable (any state cannot be completely identified by the outputs).

Similarly, for the other fixed point,

$$\text{Co}_{END} = \big[0\big]_{4x4} \tag{26}$$

$$\text{Ob}_{END} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{27}$$

The lack of full controllability is expected, given that in absolutely no state can you reverse the virus and force the S, E, and I compartments to be "replenished" by the subsequent compartment that an individual has fallen into. Once an individual has entered a compartment they cannot go backwards - thus, that state can never again be reached.

Less obvious is lack of full observability. In this case, it is because we have defined our output $\vec{y}$ to be exactly equal to $\vec{x}$, without taking into account other controlling variables such as β.

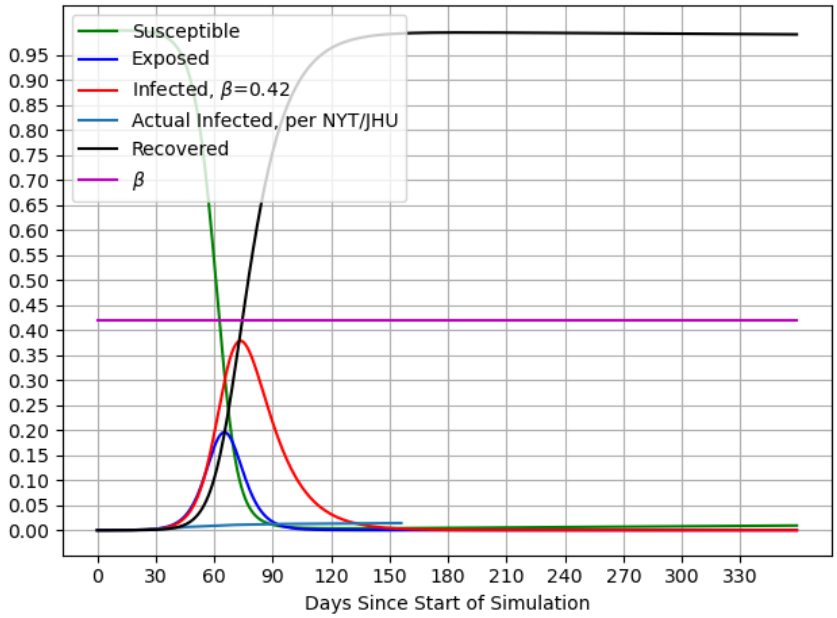
It follows that the nonlinear system is neither fully controllable nor observable.

4 System Control

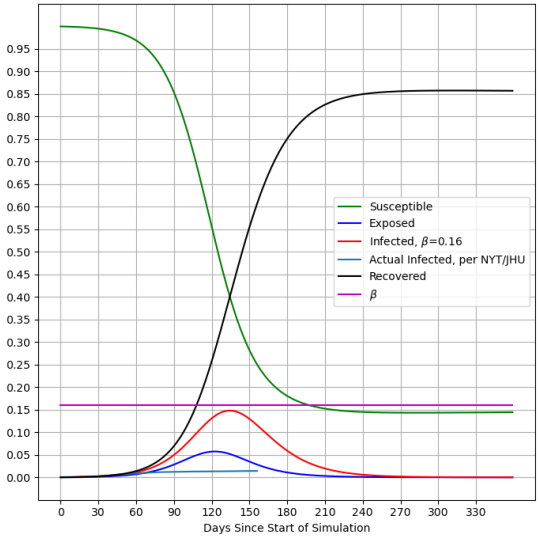
4.1 Open-Loop Analysis

Given our initial conditions from Section 3.2, left unchecked and uncontrolled, it would be very easy for the system to quickly spiral out of control within a year (see Figure 7) - initial conditions give the virus an R0 number of 6 (where Section 3.2.3 concludes an R0 of 0.07 is necessary to reduce numbers. With a case fatality rate of (average) 2.2%, that would put the death toll at roughly 78,000 people!

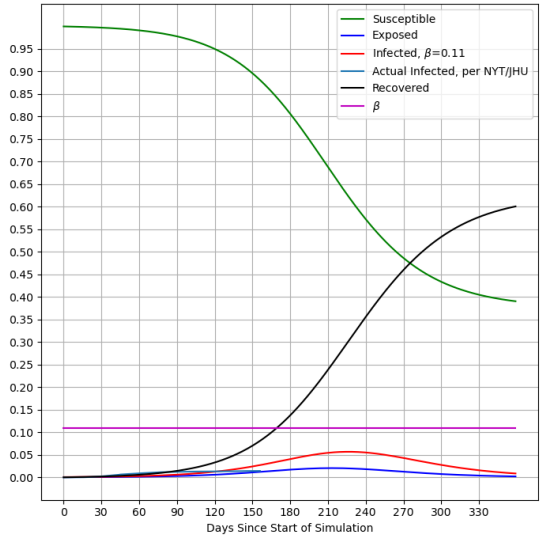
Luckily, the state of CT has handled the situation incredibly well so far, as evidenced in the same figure. Multiple safety measures and quick action have served to curtail the spread of the coronavirus. However, while this is good for the health of the population, CT still has to deal with the many individuals who are out of work or struggling to make ends meet while most public spaces are closed down, severely limiting cash flow in the state. In this section, an optimal control model will be studied and tested in an attempt to balance the reopening of the state with the continuing threat of COVID-19.



(a) Left unchecked (no control), 100% of Connecticut's population would have contracted the coronavirus within one year ($\beta = 0.42$).



(b) CT's response during the first 60 days of available data ($\beta = 0.16$) would still drive CT above 10% of the population infected within the first 6 months.



(c) CT's current response ($\beta = 0.11$) limits the infected population very successfully. Note that a β of 0.07 is required to bring R0 below 1.

Figure 7: CT's current actual rate of infection is much lower than it's rate of infection in the first 30 and 60 days.

4.2 Nonlinear Model Predictive Control

4.2.1 Control Parameters

Our target for the nonlinear optimal control will be to limit the number of infected to 10% of the population, with no overshoot and a response time of one month. In reality this means that the control will attempt to maximize **I** given the inequality constraint:

$$I(t) \leq 0.10 \tag{28}$$

We could have also tried to minimize I given the opposite equality. Initial conditions as of June 8, 2020 are given by the following:

$$\begin{aligned} S_0 &= 0.969865 \\ E_0 &= 0.006550 \\ I_0 &= 0.010616 \\ R_0 &= 0.013146 \end{aligned}$$

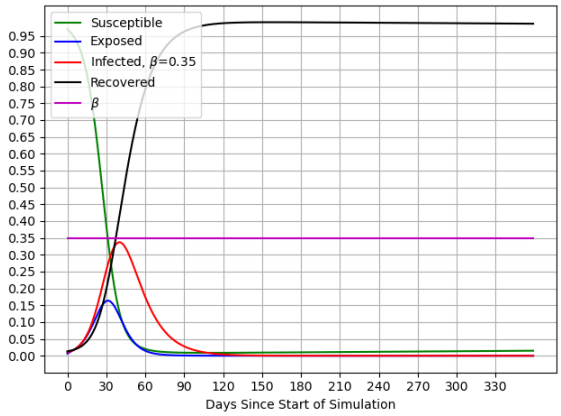
The initial conditions chosen were taken from the *simulation* that most closely matched the actual infected data (Section 3.2.2), and updated daily from the simulation. This is because while we can assume that the initial infected population is correct due to the amount of testing that CT has done (see Section 3.2.1), the same cannot be said for any of the other variables. The population of exposed may never be captured fully because those who are asymptotic generally do not get tested, and the number of recovered is also missing the number of individuals who were asymptomatic and recovered without knowing they were infected (See Section 3.2's discussion on "hidden states").

The primary tool of the control system will be a reduction of the variable β, or the rate of exposure. Control is tested on a biweekly and monthly basis, since it would not be reasonable to expect policy changes to be made on a faster timetable. The dynamic optimization Python library GEKKO [5] is used to optimize the solution given our setpoint from Equation 28. The prediction horizon is set to one year.

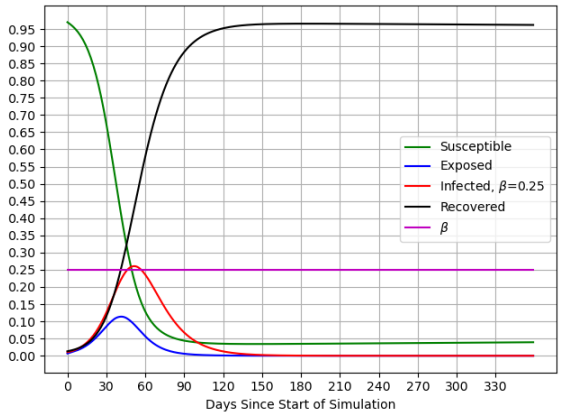
Because there are only a limited amount of control measures available to the state government, we will later treat β as a discrete variable in the midpoint of the ranges given in Table 2, roughly based on New Zealand's Alert System [1]. As evidenced by Section 3.2, CT is currently somewhere in Level 3 or Level 4, after having started out in Level 0.

Level	β	Measures Taken
0	(0.4, 0.5)	No control measures
1	(0.3, 0.4)	Intensive Testing for COVID-19 Self-isolation and quarantine required Contact tracing and record-keeping begins No restrictions on personal movement or gatherings Schools and workplaces open
2	(0.2, 0.3)	All Level 1 Measures Physical Distancing of 6 feet when in public, where practical Gatherings limited to 100 people Businesses and public venues can open if they comply
3	(0.1, 0.2)	All Level 2 Measures Businesses must work from home where practical Most public venues are closed Face masks required Gatherings limited to 50 people Inter-regional travel highly limited
4	(0.0, 0.1)	All Level 3 Measures Travel is severely limited Personal Movement is severely limited to necessities Gatherings are limited to 10 people

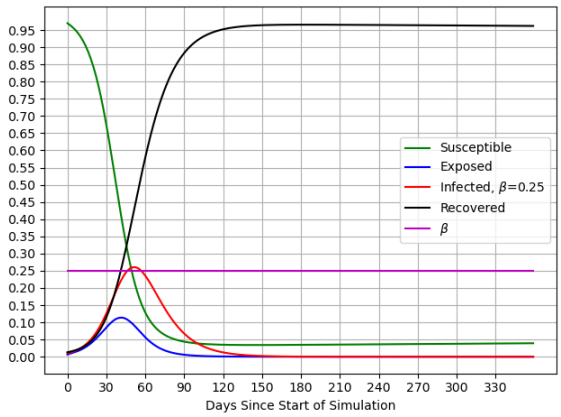
Table 2: β Values



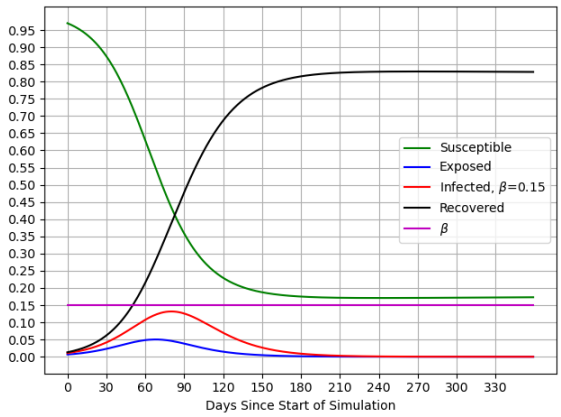
(a) Level 0, $\beta = 0.45$



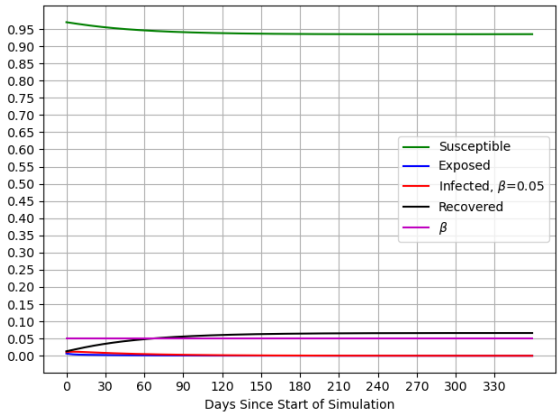
(b) Level 1, $\beta = 0.35$



(c) Level 2, $\beta = 0.25$



(d) Level 3, $\beta = 0.15$



(e) Level 4, $\beta = 0.05$

Figure 8: System behavior as β increases, according to Table 2. Notice that Level 4 is the only level capable of limiting the number of infected to less than 10% of the population, eventually reaching disease-free equilibrium.

4.2.2 Closed Loop Control

The algorithm's first attempt at closed-loop control was done with continuous β as a test - it successfully limited the number of infected to below 10% but is illustrative of some of the issues with the control that need to be corrected.

One such example is β jumping between large values too frequently. Figure 9a, for example, shows multiple large jumps, as the algorithm seems to "relax" when the number of infected drops and "panic" when the number of infected grows. This is likely because the algorithm has a severely limited amount of time to drive the number of infected down, as opposed to Figure 9b, where the only difference is an increase in the number of days between policy implementation. Such wild jumps would also likely antagonize the population.

Implementation of the control with discrete values for β can be done in two ways with GEKKO [2] :

1. Integer Variables with the APOPT Solver
2. Post-Processing the IPOPT Solution

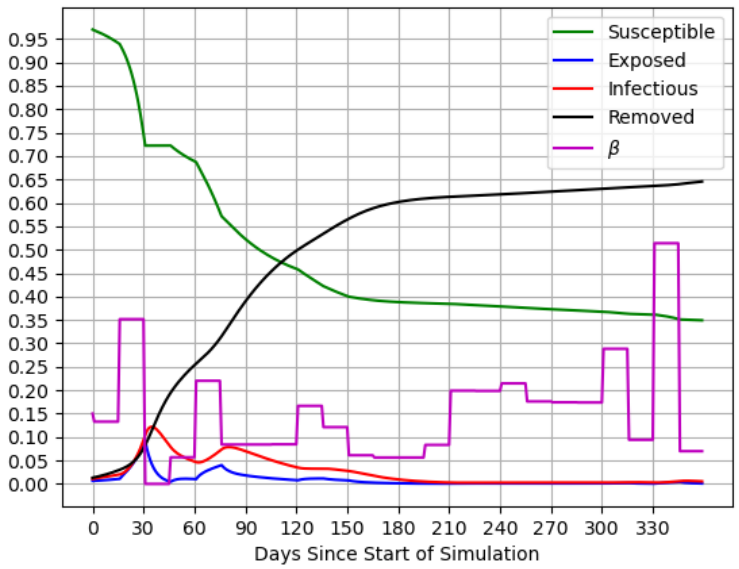
The first method necessitates a conversion between β values and integer values. This is given by Equation 29.

$$\beta = \frac{\beta_M + 0.5}{10} \tag{29}$$

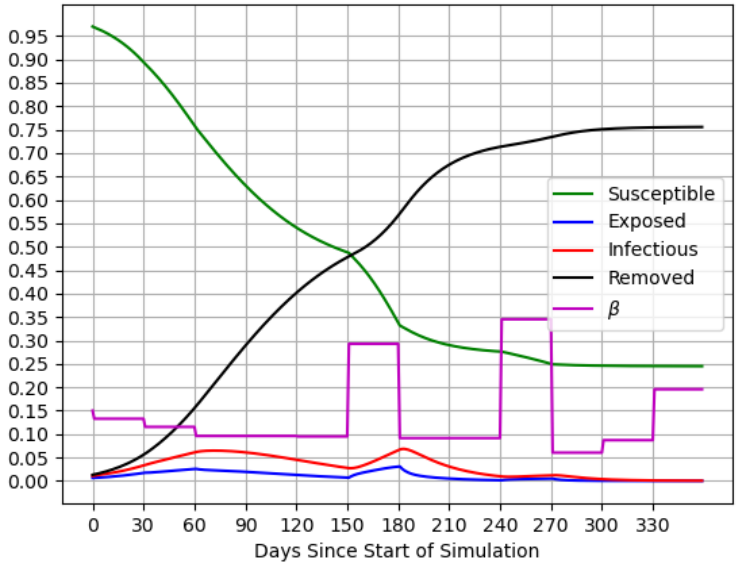
The second method involves rounding the result from the continuous solver to the nearest discrete value. Figure 10 shows the results from each solver, respectively. The results are much calmer, with the post-processed IPOPT solver somewhat imitating it's continuous counterpart. The integer-confined APOPT solution is the smoothest solution as expected, since it is explicitly solving for discrete variables.

One thing that is interesting about the discrete solvers is that they do not seem to be able to drive the system towards the setpoint, possibly because any increase in β would drive it over the setpoint within the prediction horizon.

[2]For a description of GEKKO's solvers, see `https://gekko.readthedocs.io/en/latest/global.html?#solver` [5]

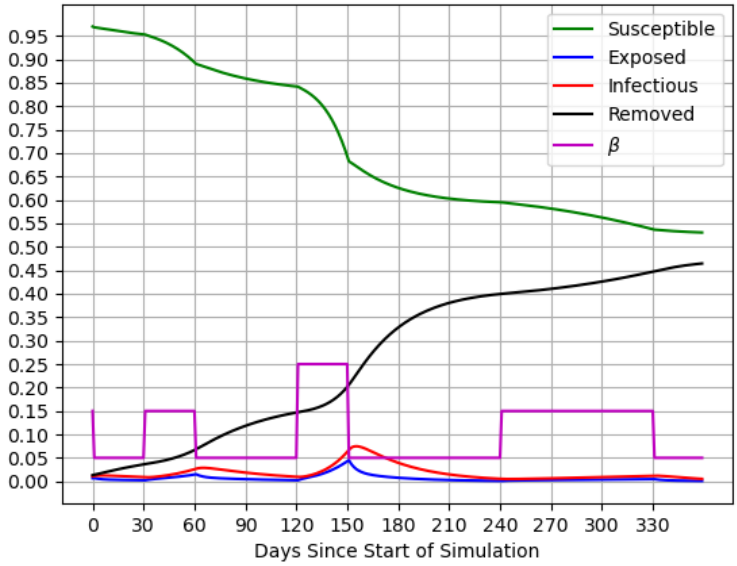


(a) Continuous model predictive control of β, implemented every fifteen days.

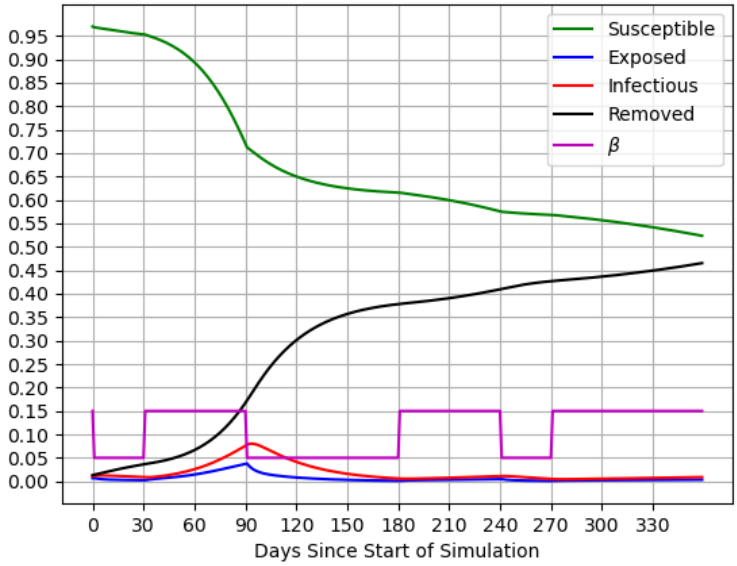


(b) Continuous model predictive control of β, implemented every thirty days.

Figure 9: β is continuous here and successfully keeps the number of infected below 10% but is choppy in both time frames (moreso when control is implemented more frequently). Implementing the control on longer time frames reduces the choppiness.



(a) Discrete model predictive control of β with the IPOPT Solver, Post-Processed



(b) Discrete model predictive control of β with the APOPT Integer Variable Solver

Figure 10: Discrete values for β reduce the choppiness further and do a better job limiting the number of infected, but do not seem to point towards reopening.

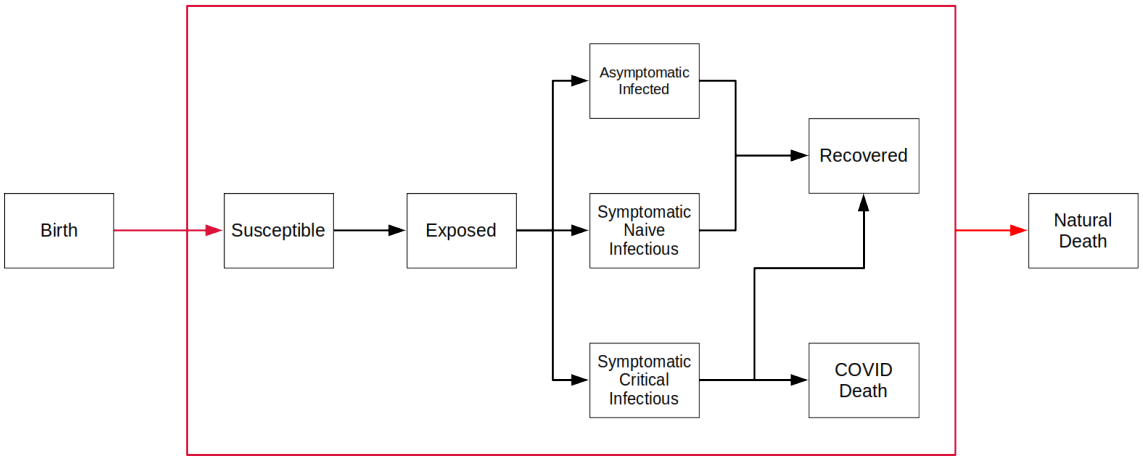


Figure 11: An updated model that includes symptomatic cases, hospitalization, and death.

5 Conclusion

In this paper a mathematical model for the pandemic was introduced and simulated in Python using the GEKKO optimization suite. A discrete and continuous nonlinear model predictive control system, implemented every thirty days, was simulated with the goal of limiting the number of infected under a setpoint. The discrete control gave smoother fluctuations in β while still maintaining the goal of the control.

Future exploration into this topic can expand upon the initial SEIR model and include more buckets, such as asymptomatic, vaccinated, deceased, tested, and hospitalized, such as shown in Figure 11. Control can then be driven towards maximizing ICU headroom, for example. Another area of exploration would be correlating economic impact with pandemic cases - for example, by a regression analysis with the number of weekly jobless claims. Finally, the model itself can improve by taking into account necessary assumptions listed in Section 3.2.1 - age, gender, race, lagging testing, and lagging control.

A Controllability and Observability Matrices in Python

```
import control
import numpy

# GLOBAL CONSTANTS
beta = 0.175
delta = 2.703e-5 # Natural Birth Rate (unrelated to disease)
gamma = 0.07 # Rate of Removal [days^-1]
mu = 2.403e-5 # Natural Death Rate (unrelated to disease)
sigma = 0.2 # Average Incubation Period [days^-1]

N = 3.57e6 # Total Population of CT [persons]
cfr = 0.022 # Case Fatality Rate [1.4% (NY) - 3%]
R = (0.175*sigma)/((mu+gamma)*(mu+sigma)) # Basic
    Reproduction Number

def controlobserve():
    J_dfe = numpy.matrix([
            [0, 0,       -beta,  0],
            [0, -sigma,  beta,   0],
            [0, sigma,   -gamma, 0],
            [0, 0,       gamma,  0]
    ])

    K_dfe = numpy.matrix([
            [0],
            [0],
            [0],
            [0]
    ])

    L_dfe = numpy.matrix([
            [1, 1, 1, 1]
    ])

    C_dfe = control.ctrb(J_dfe, K_dfe)
    O_dfe = control.obsv(J_dfe, L_dfe)

    rankC_dfe = numpy.linalg.matrix_rank(C_dfe)
    print(C_dfe)
```

```
    rankO_dfe = numpy.linalg.matrix_rank(O_dfe)
    print(O_dfe)

    J_end = numpy.matrix([
            [0, 0,       -gamma, 0],
            [0, -sigma,  gamma,  0],
            [0, sigma,   -gamma, 0],
            [0, 0,       gamma,  0]
    ])

    K_end = numpy.matrix([
            [0],
            [0],
            [0],
            [0]
    ])

    L_end = numpy.matrix([
            [1, 1, 1, 1]
    ])

    C_end = control.ctrb(J_end, K_end)
    O_end = control.obsv(J_end, L_end)

    rankC_end = numpy.linalg.matrix_rank(C_end)
    print(C_end)
    rankO_end = numpy.linalg.matrix_rank(O_end)
    print(O_end)


if __name__ == "__main__":
    controlobserve()
```

B Nonlinear Model Predictive Control with GEKKO in Python

```
import glob
import math
import matplotlib.pyplot as plt
import numpy
import os
import pandas
import scipy
import scipy.integrate
import sympy
from gekko import GEKKO

# GLOBAL CONSTANTS
delta    = 2.703e-5 # Natural Birth Rate (unrelated to disease)
gamma    = 0.07 # Rate of Removal [days^-1]
mu       = 2.403e-5 # Natural Death Rate (unrelated to disease)
sigma    = 0.2 # Average Incubation Period [days^-1]
N        = 3565287 # Total Population of CT [persons]
cfr      = 0.022 # Case Fatality Rate [1.4% (NY) - 3%]
beds     = 8798 # Number of hospital beds available
icub     = 674 # Number of ICU beds available

def main():

    # Allows me to switch between discrete and continuous beta easily
    discrete = False

    # Time to solve
    tf  = 360                           # Final Time
    dt  = 1                             # Timestep
    t   = numpy.arange(0, tf, dt)       # Time Array
    cp  = 30                            # Control Period (every 30 days)

    #   ___      __| |    ___
    #  / _ \    / _` |   / _ \
    # | (_) |  | (_| |  |  __/
```

```
#  \---/   \--,-|  \---|

# Define initial values for ODE solver (from
    parameter_estimation.py)
s0      = 0.969865          # Susceptible
e0      = 0.006550          # Exposed
i0      = 0.010616          # Active Infected
isp     = 0.1               # Active Infected Setpoint
r0      = 0.013146          # Recovered
betam0  = 1                 # Integer Beta
beta0   = (betam0 + 0.5)/10 # Beta

# Pre-create arrays to store results
s       = numpy.ones(len(t)) * s0
e       = numpy.ones(len(t)) * e0
i       = numpy.ones(len(t)) * i0
isp     = numpy.ones(len(t)) * isp
r       = numpy.ones(len(t)) * r0
betam   = numpy.ones(len(t)) * betam0
beta    = numpy.ones(len(t)) * beta0

#   -- -    ---  | | -- | | --   ---
#  / _` |  / _ \ | |/ / | |/ /  / _ \
# | (_| | |  __/ |   <  |   <  | (_) |
#  \__, |  \___| |_|\_\ |_|\_\  \___/
# |___/

# Create GEKKO Model
m       = GEKKO(remote=False)
m.time  = numpy.arange(0, tf, dt)

# Define variables and initial value for GEKKO Model
m.s = m.SV(value=s0, lb=0, ub=1)
m.e = m.SV(value=e0, lb=0, ub=1)
m.i = m.CV(value=i0, lb=0, ub=1)
m.r = m.SV(value=r0, lb=0, ub=1)

m.s.FSTATUS = 1
m.e.FSTATUS = 1
m.i.STATUS  = 1
m.i.FSTATUS = 1
m.i.SPHI    = 0.3
```

```
m.i.SPLO     = 0
m.r.FSTATUS = 1

# Beta is the manipulated variable
if (discrete):
    m.betam          = m.MV(value=betam0, lb=0, ub=10,
        integer=True)
    m.betam.STATUS   = 1
    m.betam.FSTATUS = 0
else:
    m.beta           = m.MV(value=beta0, lb=0, ub=1)
    m.beta.STATUS    = 1
    m.beta.FSTATUS   = 0

# Define equations for GEKKO Model
if (discrete):
    m.Equation( m.s.dt() == delta - (( (m.betam + 0.5) /
        10 )*m.s*m.i) - (mu*m.s) )
    m.Equation( m.e.dt() == (( (m.betam + 0.5) / 10 )*m.s
        *m.i) - (sigma*m.e) - (mu*m.e) )
else:
    m.Equation( m.s.dt() == delta - (m.beta*m.s*m.i) - (
        mu*m.s) )
    m.Equation( m.e.dt() == (m.beta*m.s*m.i) - (sigma*m.e
        ) - (mu*m.e) )

m.Equation( m.i.dt() == (sigma*m.e) - (gamma*m.i) - (mu*m
    .i) )
m.Equation( m.r.dt() == (gamma*m.i) - (mu*m.r) )

# MODEL OPTIONS
# m.Obj(-m.i) # Model objective (minimize or maximize i)
m.options.IMODE = 6 # CONTROL

if (discrete):
    # APOPT is the only solver that handles integer
        values
    m.options.SOLVER = 1 # APOPT (Advanced Process
        Optimizer)
    # These options only apply to APOPT
    m.solver_options = ['minlp_gap_tol 10',\
                        'minlp_maximum_iterations 50',\
```

```
                    'minlp_max_iter_with_int_sol 10',\
                    'minlp_as_nlp 0',\
                    'minlp_branch_method 1',\
                    'minlp_integer_tol 0.35']
else:
    m.options.SOLVER = 3 # IPOPT (Interior Point
        Optimizer)

#  ___   (_)  _ __ ___     _   _  | |   __ _  | |_  (_)
   ___     _ __
# / __| | | | '_ ` _ \  | | | | | |  / _` | | __| | | /
   _ \  | '_ \
# \__ \ | | | | | | | | | |_| | | | | (_| | | |_  | | | (
   _) | | | | |
# |___/ |_| |_| |_| |_|  \__,_| |_|  \__,_|  \__| |_|  \
   ___/  |_| |_|

plt.ion()
plt.show()

for ind in range(len(t)-1):
    print(ind, betam[ind], beta[ind])
    tsim    = [ t[ind], t[ind+1] ]
    init    = [ s[ind], e[ind], i[ind], r[ind], betam[ind
       ], beta[ind] ]
    sol     = scipy.integrate.solve_ivp(rhs, tsim, init)

    s[ind+1] = sol['y'][0][-1]
    e[ind+1] = sol['y'][1][-1]
    i[ind+1] = sol['y'][2][-1]
    r[ind+1] = sol['y'][3][-1]

    if ind%cp == 0:
        m.e.MEAS = e[ind]
        m.i.MEAS = i[ind]

        try:
            m.solve(disp=True)
            if (discrete):
                betam[ind+1] = m.betam.NEWVAL
                beta[ind+1] = (betam[ind+1] + 0.5) / 10
            else:
```

```
                    beta[ind+1] = round(m.beta.NEWVAL) + 0.05
            except:
                betam[ind+1] = 1
                beta[ind+1] = 0.15
        else:
            betam[ind+1] = betam[ind]
            beta[ind+1] = beta[ind]

        plt.cla()
        plt.xticks(numpy.arange(0, tf, 30))
        plt.yticks(numpy.arange(0, 1, 0.05))
        plt.grid(True)
        plt.xlabel("Days Since Start of Simulation")
        plt.plot(t, s, 'g-', label='Susceptible')
        plt.plot(t, e, 'b-', label='Exposed')
        plt.plot(t, i, 'r-', label='Infectious')
        plt.plot(t, r, 'k-', label='Removed')
        plt.plot(t, beta, 'm-', label=r'$\beta$')
        plt.legend()
        plt.pause(0.0001)

    plt.ioff()
    plt.show()

def rhs(dt, init):

    # Unpack Arguments
    s       = init[0]
    e       = init[1]
    i       = init[2]
    r       = init[3]
    betam   = init[4]
    beta    = init[5]

    # Solve dynamics
    sdot        = delta - (beta*s*i) - (mu*s)
    edot        = (beta*s*i) - (mu*e) - (sigma*e)
    idot        = (sigma*e) - (gamma*i) - (mu*i)
    rdot        = (gamma*i) - (mu*r)
    betamdot    = 0
    betadot     = 0
```

```
    return [sdot, edot, idot, rdot, betamdot, betadot]

if __name__ == "__main__":
    main()
```

C Parameter Estimation in Python

```
# Parameter Estimation

import matplotlib.pyplot as plt
import numpy
import os
import pandas
import scipy
import scipy.integrate

# GLOBAL CONSTANTS
delta   = 2.703e-5 # Natural Birth Rate (unrelated to disease)
gamma   = 0.07 # Rate of Removal [days^-1]
mu      = 2.403e-5 # Natural Death Rate (unrelated to disease)
sigma   = 0.2 # Average Incubation Period [days^-1]
N       = 3565287 # Total Population of CT [persons]
cfr     = 0.022 # Case Fatality Rate [1.4% (NY) - 3%]
beds    = 8798 # Number of hospital beds available
icub    = 674 # Number of ICU beds available

def estimate():

    nyt     = pandas.read_csv('data/nytimes.csv')
    nytct   = nyt[(nyt.state == "Connecticut")].copy()
    nytct['date']       = pandas.to_datetime(nytct['date'])
    nytct['difference'] = nytct['cases'].diff()
    nytct['pct_change'] = nytct['cases'].pct_change()
    nytct   = nytct.reset_index()
    nytct.index += 0

    # ODE VARIABLES
    tf      = 360
    dt      = 1
    tspan   = [0, tf]
    teval   = numpy.arange(0, tf, dt)
    targs   = [tf, dt, tspan, teval]

    (betasearch, esearch, isearch) = search(targs, nytct)
    # (betasearch, esearch, isearch) = (0.42, 100, 50) #
```

```
    Results for tf = 30
# (betasearch, esearch, isearch) = (0.16, 1650, 950) #
    Results for tf = 60
# (betasearch, esearch, isearch) = (0.11, 950, 1950) #
    Results for tf = 150
R = (betasearch*sigma)/((mu+gamma)*(mu+sigma)) # Basic
    Reproduction Number
print(betasearch, esearch, isearch, R)

# INITIAL CONDITIONS
e0 = esearch/N  # Exposed
i0 = isearch/N  # Active Infected
r0 = 0          # Recovered (effectively 0)
s0 = 1-e0-i0-r0 # Susceptible

# INIT
init = [s0, e0, i0, r0, betasearch]
beta = numpy.ones(tf) * betasearch

# SOLVE ODE
solution = scipy.integrate.solve_ivp(rhs, tspan, init,
    t_eval=teval)
solution = numpy.c_[numpy.transpose(solution['y'])]
seir             = pandas.DataFrame(data=solution)
seir['date']     = pandas.date_range(start='3/8/2020',
    periods=len(seir), freq='D')
seir.columns     = ['susceptible', 'exposed', 'infected',
    'recovered', 'beta', 'date']

plt.xlabel("Days Since Start of Simulation")
plt.ylabel("Portion of Population in Compartment")
plt.xticks(numpy.arange(0, tf, 30))
plt.yticks(numpy.arange(0, 1, 0.05))
plt.grid(True)
plt.plot(seir.index, seir['susceptible'], 'g-', label='
    Susceptible')
plt.plot(seir.index, seir['exposed'], 'b-', label='
    Exposed')
plt.plot(seir.index, seir['infected'], 'r-', label=r'
    Infected, $\beta$=0.16')
plt.plot(nytct.index, nytct['cases']/N, label='Actual
    Infected, per NYT/JHU')
```

```
    plt.plot(seir.index, seir['recovered'], 'k-', label='
        Recovered')
    plt.plot(seir.index, beta, 'm-', label=r'$\beta$')

    # plt.plot(seir['date'], seir['infected'], label=r'
        Proportion of Infected, Simulated with $\beta$=0.11')
    # plt.scatter(nytct['date'].head(tf), nytct['cases'].head
        (tf)/N, label='Proportion of Infected, per NYT/JHU')
    # plt.gcf().autofmt_xdate()
    plt.legend()
    plt.tight_layout()
    plt.show()

def rhs(dt, init):

    # Unpack Arguments
    s = init[0]
    e = init[1]
    i = init[2]
    r = init[3]
    beta = init[4]

    # Solve dynamics
    sdot = delta - (beta*s*i) - (mu*s)
    edot = (beta*s*i) - (mu*e) - (sigma*e)
    idot = (sigma*e) - (gamma*i) - (mu*i)
    rdot = (gamma*i) - (mu*r)

    return [sdot, edot, idot, rdot, 0]

def search(targs, comparison_data):
    tf = targs[0]
    dt = targs[1]
    tspan = targs[2]
    teval = targs[3]

    # SEARCH FOR SMALLEST ERROR IN SIMULATIONS
    base_error = 10
    for beta in numpy.arange(0.1, 0.6, 0.01):
        for e in numpy.arange(0, 2000, 50):
            for i in numpy.arange(0, 2000, 50):
                print(beta, e, i)
```

```
                    # INITIAL CONDITIONS
                    e0 = e/N          # Exposed
                    i0 = i/N          # Active Infected
                    r0 = 0/N          # Recovered
                    s0 = 1-e0-i0-r0 # Susceptible

                    # INIT
                    init = [s0, e0, i0, r0, beta]

                    # SOLVE ODE
                    solution = scipy.integrate.solve_ivp(rhs,
                        tspan, init, t_eval=teval)
                    solution = numpy.c_[numpy.transpose(solution[
                        'y'])]
                    seir = pandas.DataFrame(data=solution)
                    seir['date'] = pandas.date_range(start='
                        3/8/2020', periods=len(seir), freq='D')
                    seir.columns = ['susceptible', 'exposed', '
                        infected', 'recovered', 'beta', 'date']

                    error = (((comparison_data['cases']/N).head(
                        tf).subtract(seir['infected']))**2).sum()
                    if (error < base_error):
                        print("New Minimum Error")
                        base_error = error
                        betasearch = beta
                        esearch = e
                        isearch = i

    return(betasearch, esearch, isearch)

if __name__ == "__main__":
    estimate()
```

D Phase Portrait in Python

```
import glob
import matplotlib.pyplot as plt
import numpy
import os
import pandas
import scipy
import scipy.integrate
import sympy

beta0 = 0.175
gamma0 = 0.07

def dRIdt(x, t=0):
    return numpy.array([ gamma0 * x[1],
                         beta0 * numpy.exp( (-beta0/gamma0) *
                            x[0] ) - (gamma0 * x[1]) ])

def dSIdt(x, t=0):
    return numpy.array([ -beta0 * x[0] * x[1],
                         beta0 * x[0] * x[1] - gamma0 * x[1]
                            ])

def dSRdt(x, t=0):
    # return numpy.array([])
    pass

def phaseRI():
    # see https://scipy-cookbook.readthedocs.io/items/
        LoktaVolterraTutorial.html
    r = numpy.linspace(0, 1, 50)
    i = numpy.linspace(0, 1, 50)
    R, I = numpy.meshgrid(r,i)
    dR, dI = dRIdt([R,I])
    M = (numpy.hypot(dR, dI))
    M[ M == 0 ] = 1.
    dR /= M
    dI /= M
    # plt.quiver(R, I, dR, dI, M, pivot='mid')
    plt.streamplot(R, I, dR, dI)
    plt.xlabel("Recovered")
```

```
    plt.ylabel("Infected")
    plt.show()

def phaseSI():
    s = numpy.linspace(0, 1, 50)
    i = numpy.linspace(0, 1, 50)
    S, I = numpy.meshgrid(s,i)
    dS, dI = dSIdt([S,I])
    M = (numpy.hypot(dS, dI))
    M[M == 0] = 1.
    dS /= M
    dI /= M
    # plt.quiver(R, I, dR, dI, M, pivot='mid')
    plt.streamplot(S, I, dS, dI)
    plt.xlabel("Susceptible")
    plt.ylabel("Infected")
    plt.show()

if __name__ == "__main__":
    phaseRI()
    # phaseSI()
```

E Plotting JHU Data in Python

```
import matplotlib.pyplot as plt
import numpy
import os
import pandas
import scipy
import scipy.integrate
pandas.set_option('display.max_rows', None)
pandas.set_option('display.max_columns', None)

N = 3565287 # Total Population of CT [persons]

def explore():

    if os.path.isfile('data/jhuct.pkl'):
        print("Reading saved JHU data")
        jhuct = pandas.read_pickle('data/jhuct.pkl')
    else:
        filelist = os.listdir('data/csse')
        filelist.sort()
        jhuct = pandas.DataFrame()
        for file in filelist:
            jhu = pandas.read_csv('data/csse/' + file)
            jhuct = jhuct.append(jhu[(jhu.Province_State == "
                Connecticut")])
        jhuct.to_pickle('data/jhuct.pkl')
    jhuct['date'] = pandas.to_datetime(jhuct['Last_Update'])

    plt.plot(jhuct['date'], jhuct['Confirmed']/N, 'g-', label
      ='Cumulative Cases')
    plt.plot(jhuct['date'], jhuct['Active']/N, 'r-', label='
      Infected')
    plt.plot(jhuct['date'], jhuct['Deaths']/N + jhuct['
      Recovered']/N, 'k-', label='Recovered')

    plt.xlabel("Date")
    plt.ylabel("Portion of Population in Compartment")
    plt.legend()
    plt.gcf().autofmt_xdate()
    plt.tight_layout()
    plt.show()
```

```
if __name__ == "__main__":
    explore()
```

References

[1] "Alert System Overview — Unite against COVID-19," New Zealand Government [Online]. Available: https://covid19.govt.nz/covid-19/restrictions/alert-system-overview.

[2] "COVID Act Now," COVID Act Now [Online]. Available: https://covidactnow.org/us/ct/?s=716119.

[3] CSSEGISandData, 2020, COVID-19 Data, Johns Hopkins University, Center for Systems Science and Engineering. [9]"COVID-19 Data — Connecticut Data," CT Open Data [Online]. Available: https://data.ct.gov/stories/s/COVID-19-data/wa3g-tfvc/.

[4] Castilho, C., Gondim, J. A. M., Marchesin, M., and Sabeti, M., 2020, "Assessing the Efficiency of Different Control Strategies for the Coronavirus (COVID-19) Epidemic," arXiv:2004.03539 [q-bio].

[5] "GEKKO Dynamic Optimization," Brigham Young University [Online]. Available: https://machinelearning.byu.edu/

[6] "Individual Hospital Statistics for Connecticut," American Hospital Directory [Online]. Available: https://www.ahd.com/states/hospital_CT.html.

[7] Kochanek, K. D., Murphy, S. L., Xu, J., and Arias, E., 2019, "Deaths: Final Data for 2017," National Vital Statistics Reports, 68(9), p. 77.

[8] Koerth, M., Bronner, L., and Mithani, J., 2020, "Why It's So Freaking Hard To Make A Good COVID-19 Model," FiveThirtyEight. Available: https://fivethirtyeight.com/features/why-its-so-freaking-hard-to-make-a-good-covid-19-model/.

[9] Lauer, S. A., Grantz, K. H., Bi, Q., Jones, F. K., Zheng, Q., Meredith, H. R., Azman, A. S., Reich, N. G., and Lessler, J., 2020, "The Incubation Period of Coronavirus Disease 2019 (COVID-19) From Publicly Reported Confirmed Cases: Estimation and Application," Annals of Internal Medicine, 172(9), pp. 577–582.

[10] Olivier, L. E., and Craig, I. K., 2020, "An Epidemiological Model for the Spread of COVID-19: A South African Case Study," arXiv:2005.08012 [physics, q-bio].

[11] Martin, J. A., Hamilton, B. E., Osterman, M. J. K., Driscoll, A. K., and Drake, P., 2018, "Births: Final Data for 2017," National Vital Statistics Reports, 67(8), p. 50.

[12] "Report of the WHO-China Joint Mission on Coronavirus Disease 2019 (COVID-19)," World Health Organization [Online]. Available: https://www.who.int/publications/i/item/report-of-the-who-china-joint-mission-on-coronavirus-disease-2019-(covid-19).

[13] Silver, N., "Coronavirus Case Counts Are Meaningless*," FiveThirtyEight Available: https://fivethirtyeight.com/features/coronavirus-case-counts-are-meaningless/.

[14] The New York Times, 2020, "We're Sharing Coronavirus Case Data for Every U.S. County," The New York Times. Available: https://www.nytimes.com/article/coronavirus-county-data-us.html.

[15] Tierney, D., and Almos, B., "COVID-19 UPDATE: Basic Reproduction Number, Pop-Up Sites, Multi-System Youth Grant," Ohio.gov [Online]. Available: https://coronavirus.ohio.gov/wps/portal/gov/covid-19/resources/news-releases-news-you-can-use/basic-reproduction-number-pop-up-sites.

[16] Tsay, C., Lejarza, F., Stadtherr, M. A., and Baldea, M., 2020, "Modeling, State Estimation, and Optimal Control for the US COVID-19 Outbreak," arXiv:2004.06291 [math, q-bio].

[17] "U.S. Census Bureau QuickFacts: Connecticut; United States," U.S. Census Bureau [Online]. Available: https://www.census.gov/quickfacts/fact/table/CT,US/PST045219.

[18] Verity, R., Okell, L. C., Dorigatti, I., Winskill, P., Whittaker, C., Imai, N., Cuomo-Dannenburg, G., Thompson, H., Walker, P. G. T., Fu, H., Dighe, A., Griffin, J. T., Baguelin, M., Bhatia, S., Boonyasiri, A., Cori, A., Cucunubá, Z., FitzJohn, R., Gaythorpe, K., Green, W., Hamlet, A., Hinsley, W., Laydon, D., Nedjati-Gilani, G., Riley, S., van Elsland, S., Volz, E., Wang, H., Wang, Y., Xi, X., Donnelly, C. A., Ghani, A. C., and Ferguson, N. M., 2020, "Estimates of the Severity of Coronavirus Disease 2019: A Model-Based Analysis," The Lancet Infectious Diseases, 20(6), pp. 669–677.

[19] Wortham, J. M., Lee, J. T., Althomsons, S., Latash, J., Davidson, A., Guerra, K., Murray, K., McGibbon, E., Pichardo, C., Toro, B., Li, L., Paladini, M., Eddy, M. L., Reilly, K. H., McHugh, L., Thomas, D., Tsai, S., Ojo, M., Rolland, S., Bhat, M., Hutchinson, K., Sabel, J., Eckel, S., Collins, J., Donovan, C., Cope, A., Kawasaki, B., McLafferty, S., Alden, N., Herlihy, R., Barbeau, B., Dunn, A. C., Clark, C., Pontones, P., McLafferty, M. L., Sidelinger, D. E., Krueger, A., Kollmann, L., Larson, L., Holzbauer, S., Lynfield, R., Westergaard, R., Crawford, R., Zhao, L., Bressler, J. M., Read, J. S., Dunn, J., Lewis, A., Richardson, G., Hand, J., Sokol, T., Adkins, S. H., Leitgeb, B., Pindyck, T., Eure, T., Wong, K., Datta, D., Appiah, G. D., Brown, J., Traxler, R., Koumans, E. H., and Reagan-Steiner, S., 2020, "Characteristics of Persons Who Died with COVID-19 — United States, February 12–May 18, 2020," MMWR Morb. Mortal. Wkly. Rep., 69(28), pp. 923–929. Available: https://www.cdc.gov/mmwr/volumes/69/wr/mm6912e2.htm?s_cid=mm6912e2_w

[20] All data and code written for this paper can be found at https://github.com/squeegene/SEIR_CT