Notes on DDEs with Epidemic Models

K. J. Bunkers

May 7, 2020

Contents

Chapter 1

Setting Up the DDE Model

We begin by laying the groundwork for delay differential equations and the epidemiological models that I will use to investigate some possible influences on the spread of a disease. This work should be viewed with extreme caution, as my code is not optimized, nor has it been benchmarked extensively enough to ensure that its results should be free of error. In addition, the models I will use for epidemiology are still fairly crude and so should not be taken to fully indicate what should be done. Instead, they should be used for allowing us to formulate and ask questions that more advanced models and experts should be able to answer or explain. That is, this helps us gain a little intuition and should not be taken as any sort of final word.

1.1 Delay Differential Equations

Because many ideas in epidemiology require us to use information from past data, it makes sense to consider the use of delay differential equations (DDEs). These are like ordinary or partial differential equations, but include data from the past. An ODE is of the form

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{F}(\mathbf{X}(t), t) \tag{1.1.1}$$

where $\mathbf{X}(t)$ is a time-dependent vector array,[1] $\mathbf{F}$ is vector array function of $\mathbf{X}(t)$ and t, and t is a time-like variable.

A DDE involves a time delay (say τ), so an example would be

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{F}(\mathbf{X}(t-\tau), t-\tau) \tag{1.1.2}$$

although generalizations abound. For example let τ_i be a set of different delays. A more generic DDE would then be of the form

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{F}(\mathbf{X}(t-\tau_1, t-\tau_2, \ldots), t-\tau, t-\tau_1, t-\tau_2, \ldots) \tag{1.1.3}$$

with multiple different time delays allowed.

[1]By vector array, I mean a vector in the mathematical sense rather than the geometric "Euclidean" vector sense.

Solving DDEs analytically is often impossible beyond a few time delays (if at all), and so numerical methods present themselves as the easiest way to solve these problems. We can convert a DDE into an ODE form and then apply normal ODE methods of solution. This arises from the fact that $\mathbf{F}$ can be rewritten into a simple function of t and parameters τ_i as

$$\mathbf{F}(\mathbf{X}(t-\tau_1, t-\tau_2, \ldots), t-\tau_1, t-\tau_2, \ldots) \rightarrow \mathbf{G}(t, \tau_1, \tau_2, \ldots) \tag{1.1.4}$$

because we know the form of $\mathbf{X}(t)$ for times in the past.[2]

The biggest difference between the ODE and DDE is that one must supply a history function instead of an initial condition. This is so that the beginning of the integration can actually be done. Typically, people use constant beginning history functions, and I will not deviate from that here. In principle, one could use a history function based on the actual history of the epidemic, but usually a constant history function gives results fairly similar to a more realistic history function so long as the delay is not large and the constant history function is not vastly different from the actual history.

1.2 SIR Models

I will use the SIR model as the basis for these investigations. This model uses a fixed population of N people divided into three groups in its most basic form. They are S(usceptible), I(nfected), R(emoved). The S represents those who are susceptible to getting the disease, the I are those that are currently infected and can spread the disease and R are those that have had the disease but can no longer infect others (so either dead or recovered). Then $S + I + R = N$ is a constant for this model, because this exhausts all possibilities for all of the people. I will normalize all of these equations (by dividing by N) because there is no actual reason for including N. Then $\widetilde{S} = S/N$, $\widetilde{I} = I/N$ and $\widetilde{R} = R/N$ are the proportion of the population in each category. I will remove the tildes for convenience from now on. Then the new equations become

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t)S(t)I(t) \tag{1.2.1}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t)S(t)I(t) - \gamma(t)I(t) \tag{1.2.2}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma(t)I(t) \tag{1.2.3}$$

$$\frac{\mathrm{d}(S+I+R)}{\mathrm{d}t} = 0 \tag{1.2.4}$$

where the last equation shows us that N (1 in our normalized case) remains invariant through time.

Note that β^{-1} represents the typical time between contact of people in the S and I group and γ^{-1} represents the typical time that a person remains infectious (remains in the I group). Simplistic models leave β and γ as constants. A somewhat useful figure of merit for an epidemic is β/γ, or the reproduction number, which can be interpreted as the number of people typically infected by an infectious person as it is the typical time a person remains infections over the typical time

[2] As an aside, one could consider ADEs or "advance" differential equations, which would require knowledge of the future rather than the past, but physical systems rarely, if ever, exhibit this feature.

between contacts (or the number of people a person contacts while being infectious). This number at its initial value is typically denoted $R_0 = \frac{\beta(0)}{\gamma(0)} = r_0$ and used to parameterize how infectious the disease is.[3] The reproduction number $r = \frac{\beta(t)}{\gamma(t)}$ can change dramatically as an epidemic progresses, however, it is a useful theoretical concept. Because the progress of the epidemic depends sensitively on r, one should approach with caution any model's predictions that use r explicitly in modeling

This simple model gives us some useful predictions, but when using real data, fitting an exponential will cause us problems if we put a lot of confidence in noisy data.

A more interesting model is the SIRDC model, which affords us a few more groups. The S and I groups remain the same, but R changes to R(esolving and not infectious) into either D(ead) or C(ompletely recovered and non-spreading).[4] We introduce the parameters θ and δ where θ^{-1} represents a typical time a person is in the R(esolving) time (how long they are sick but not infectious) and δ is the proportion of the R dying, or the death rate.

The equations then become

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t)S(t)I(t) \tag{1.2.5}$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t)S(t)I(t) - \gamma(t)I(t) \tag{1.2.6}$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma(t)I(t) - \theta(t)R(t) \tag{1.2.7}$$
$$\frac{\mathrm{d}D}{\mathrm{d}t} = \delta\theta(t)R(t) \tag{1.2.8}$$
$$\frac{\mathrm{d}C}{\mathrm{d}t} = (1-\delta)\theta(t)R(t) \tag{1.2.9}$$
$$\frac{\mathrm{d}(S+I+R+D+C)}{\mathrm{d}t} = 0 \tag{1.2.10}$$

1.3 Combining Ideas

Now the previous SIR models are all ODEs, and we would like to introduce some delays. There are many possibilities available for introducing delays with varying degrees of realism. One could suppose that I actually depends on the number of susceptible S at an earlier time (τ before), for example, in which case one could try

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t)S(t)I(t) \tag{1.3.1}$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t)S(t-\tau)I(t) - \gamma(t)I(t) \tag{1.3.2}$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma(t)I(t) - \theta(t)R(t) \tag{1.3.3}$$
$$\frac{\mathrm{d}D}{\mathrm{d}t} = \delta\theta(t)R(t) \tag{1.3.4}$$
$$\frac{\mathrm{d}C}{\mathrm{d}t} = (1-\delta)\theta(t)R(t) \tag{1.3.5}$$

[3] It is somewhat unfortunate since R is a group in the above, and so we must keep these variables separate and not confused. I will use r_0 to prevent any confusion from now on.

[4] If you prefer, ReCovered.

Note that a core assumption of the model is now broken, however. $S+I+R+D+C$ is no longer necessarily a constant. One way to avoid such a problem is to replace (1.3.1) with

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t)S(t-\tau)I(t) \tag{1.3.6}$$

which is more consistent with the idea that I depends on the previous number of susceptible people. For we are now saying that it is that group of "delayed" people interacting with the I group that drives the dynamics.

This model is fine, but one can easily question whether it is really offering us any further insight into the problem. First, it is questionable that this sort of delayed time dependence is physically reasonable. Second, and perhaps more importantly, we have not changed $\beta(t)$ which would appear to be the most influenced by delays. For it is not the population of people who are delayed, but their choices, which are based on information from the past. This because the β parameter is related to how well people isolate from each other, among other things. The other parameters are mostly determined by the disease itself, and so should not necessarily be delayed, and one might guess should remain mostly constant barring health care advances or evolution of the disease.

The SIR model is extremely well-known in the epidemiology literature. The SIRDC model comes from Jones and Villaverde via John Cochrane (aka, the Grumpy Economist), which feature good discussions of the problem and insights into what we can glean from these models.

I will use a RK4 (Runge-Kutta, 4th order) integrator for the DDEs. A description of the algorithm is exactly the same as for ODEs, except that the right hand sides include delayed functions. This only requires a little more machinery (interpolation of our solution function into the past) to correctly program.

Chapter 2

SIRDC DDEs

We will here only consider changing $\beta(t)$ into $\beta(t-\tau)$, and so will use the equations

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t-\tau)S(t)I(t) \tag{2.0.1}$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t-\tau)S(t)I(t) - \gamma(t)I(t) \tag{2.0.2}$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma(t)I(t) - \theta(t)R(t) \tag{2.0.3}$$
$$\frac{\mathrm{d}D}{\mathrm{d}t} = \delta\theta(t)R(t) \tag{2.0.4}$$
$$\frac{\mathrm{d}C}{\mathrm{d}t} = (1-\delta)\theta(t)R(t) \tag{2.0.5}$$

We would like to set the values in rough accordance with the covid-19 epidemic, which has been given rough values of $\gamma = 0.2\,\text{days}^{-1}$, $\theta = 0.1\,\text{days}^{-1}$ and $r_0 = 5$ which implies that $\beta(0) = 1\,\text{days}^{-1}$. This corresponds to a time between contacts of $1/\beta(0)^{-1}$ or 1 day, an infectious period of $1/\gamma$ or 5 days, and a time in the hospital of $1/\theta$ or 10 days. The death rate is roughly given by $\delta = 0.008$ or 0.8%.[1]

He proposes a mechanism by which as more deaths occur people change their behavior, making β smaller (that is, making times between contact longer) such that one goes to $r \to 0.5$ which means $\beta \to 0.1$. His proposal is that the behavior matches one of the following

$$\beta_I(t) = \beta_0 \exp(-\alpha_I I(t)) \tag{2.0.6}$$
$$\beta_D(t) = \beta_0 \exp(-\alpha_D \frac{\mathrm{d}D}{\mathrm{d}t}(t)) \tag{2.0.7}$$

One then chooses α_X such that when $X(t)$ (equal to $I(t)$ or $\frac{\mathrm{d}D}{\mathrm{d}t}$) is a certain number, we approach the desired r. For example, we could choose $\alpha_I = 5\times10^{-3}$ and $\alpha_D = 5\times10^{-5}\,\text{day}^{-1}$ so that

$$0.1 = \beta_0 \exp(-\alpha_I I(t)) = \exp(-\alpha_I[5\times10^{-3}]) \tag{2.0.8}$$
$$\alpha_I = -\frac{\ln(0.1)}{5\times10^{-3}} \approx 460.5 \tag{2.0.9}$$

[1]All the numbers are taken from Cochrane.

and so

$$0.1 = \beta_0 \exp(-\delta_D \frac{\mathrm{d}D}{\mathrm{d}t}) = \exp(-\alpha_D[5 \times 10^{-5}\,\mathrm{day}^{-1}]) \tag{2.0.10}$$

$$\alpha_D = -\frac{\ln(0.1)}{5 \times 10^{-5}\,\mathrm{day}^{-1}} \approx 4.6 \times 10^{4}\,\mathrm{day} \tag{2.0.11}$$

Convergence of this model should require a small enough time step, but not drastically smaller than a day. My testing (not presented) has shown that a time step of a 0.1 day is sufficient for convergence, but that 1 day is sometimes not converged.

Let's start our calculations by looking at ODEs, or without any time delays. We use the numbers provided above and find that for an initial infected proportion of 10^{-6}. We find the results are given in Figure 2.1.

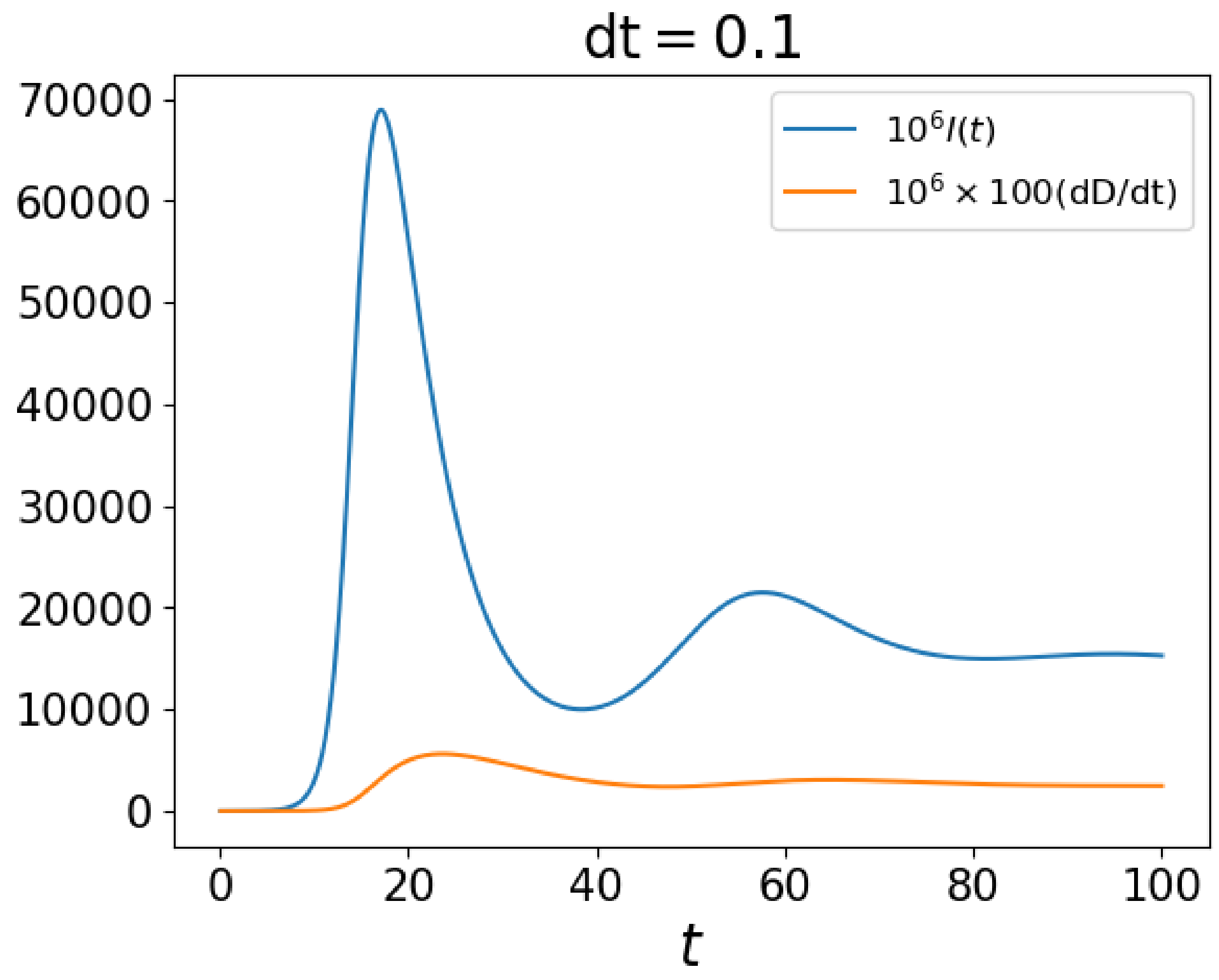


Figure 2.1: This shows the solutions for $I(t)$ and $\frac{\mathrm{d}D}{\mathrm{d}t}$ for α_I dependent on the current infection population with $\alpha_I \approx 460.5$ corresponding to $r = 0.5$ for $I = 5 \times 10^3/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$, and the figure uses $\mathrm{d}t = 0.1\,\mathrm{day}$.

We can then do a calculation based on the death rate which is shown in Figure 2.2 with an initial sick proportion of 1×10^{-6} again.

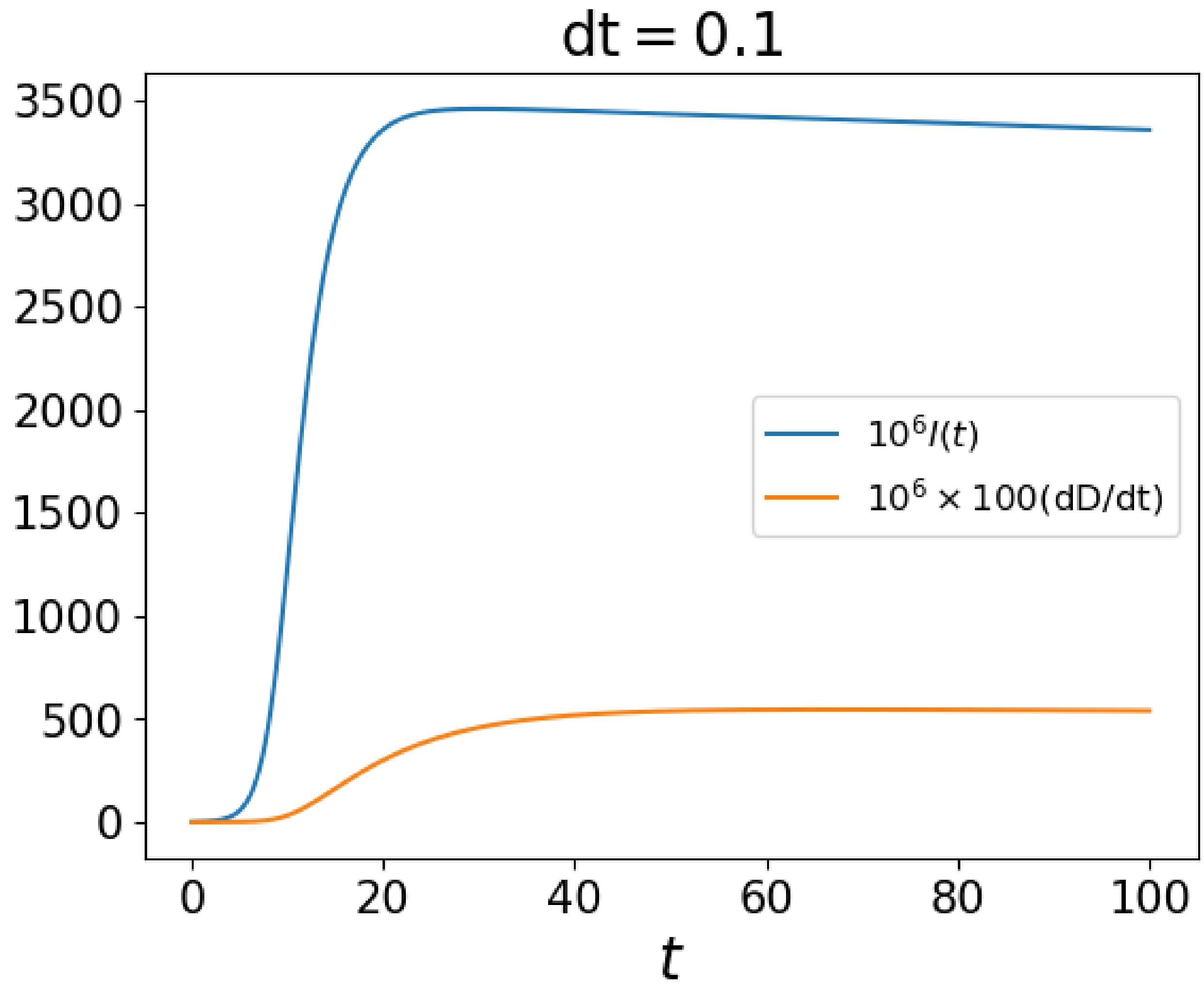


Figure 2.2: The solutions for $I(t)$ and $\frac{\mathrm{d}D}{\mathrm{d}t}$ are shown for α_D dependent on the current death rate with $\alpha_D \approx 46\,050\,\mathrm{day}$ corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$, and the figure uses $\mathrm{d}t = 0.1\,\mathrm{day}$.

We can now think about taking delays. We will use the 0.1 day as our time difference, which as I previously stated, I have checked to be converged. Clearly, if we use a large time delay, then it is as if β is constant for its early evolution and so we will see barely any change from doing a regular ODE solution with a constant β. It will eventually show some oscillations later on, but after it has settled down to a near steady state value. We can see this by looking at $I(t)$ for various τ. These are shown in Figure 2.3 and 2.4. The peak death rate when monitoring the infected actually "only" grows by a factor of about seven. In addition, a delay actually leads to less oscillations in the long run in the death rate, but the diminishing oscillations hardly make up for the increased number of deaths.

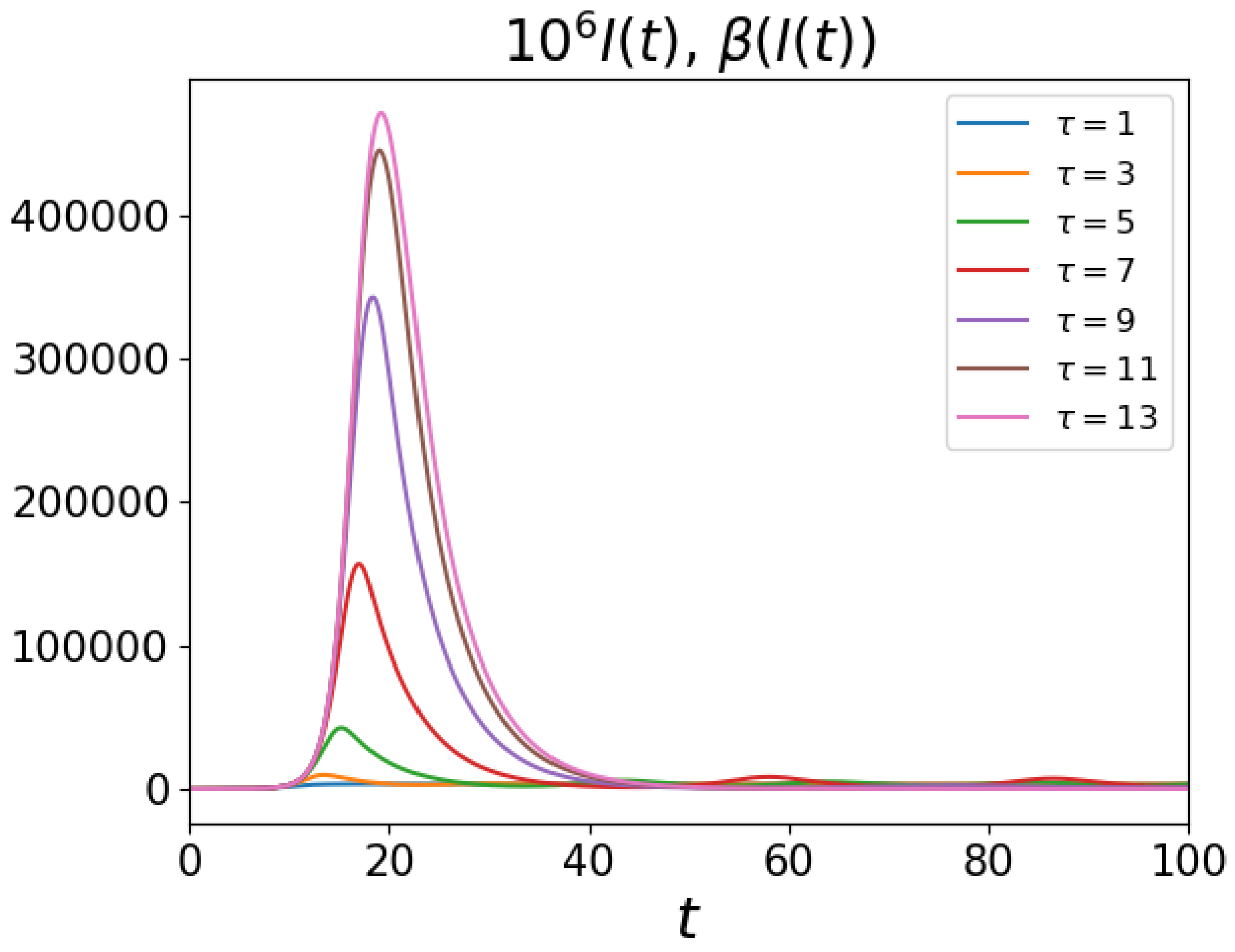
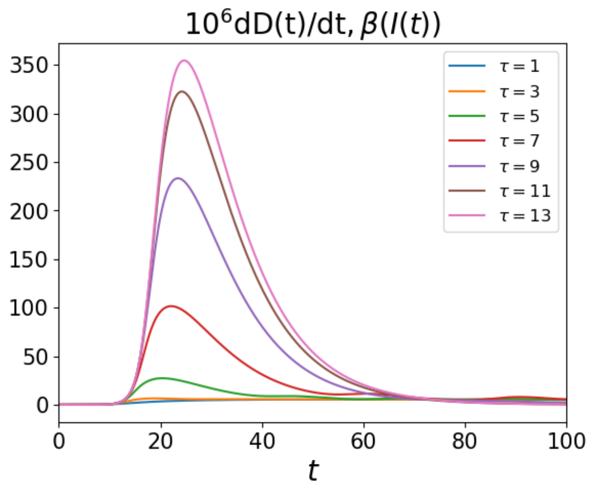


Figure 2.3: This shows the solution for $I(t)$ with α_I dependent on the infected population with $\alpha_I \approx 460.5$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. We see that the longer the delay, the more people get infected.



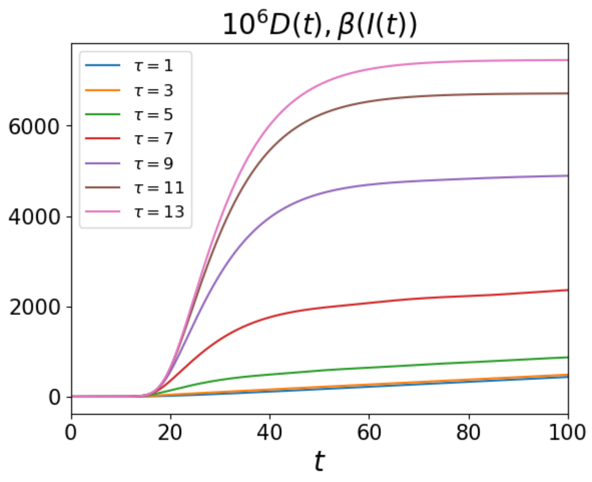


Figure 2.4: These figures show the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ for α_I dependent on the infected population with $\alpha_I \approx 460.5$ corresponding to $r = 0.5$ for $I(t) = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. The left figure shows the death rate and the right figure the cumulative total of deaths. The death rate simply gets worse as we include delays. The oscillations seem to die down a bit in the long run, but at the cost of a far worse peak.

Now we can consider a delay based on the death rate of previous days. Longer delays seem unlikely (people will know the rate fairly accurately within the last couple of days, I would think), but it is worth seeing the behavior regardless. We can see this by looking at $I(t)$ for various τ. These are shown in Figure and . The peak death rate when monitoring the infected actually "only" grows by a factor of about seven. In addition, a delay actually leads to less oscillations in the long run in the death rate, but the diminishing oscillations hardly make up for the increased number of deaths.

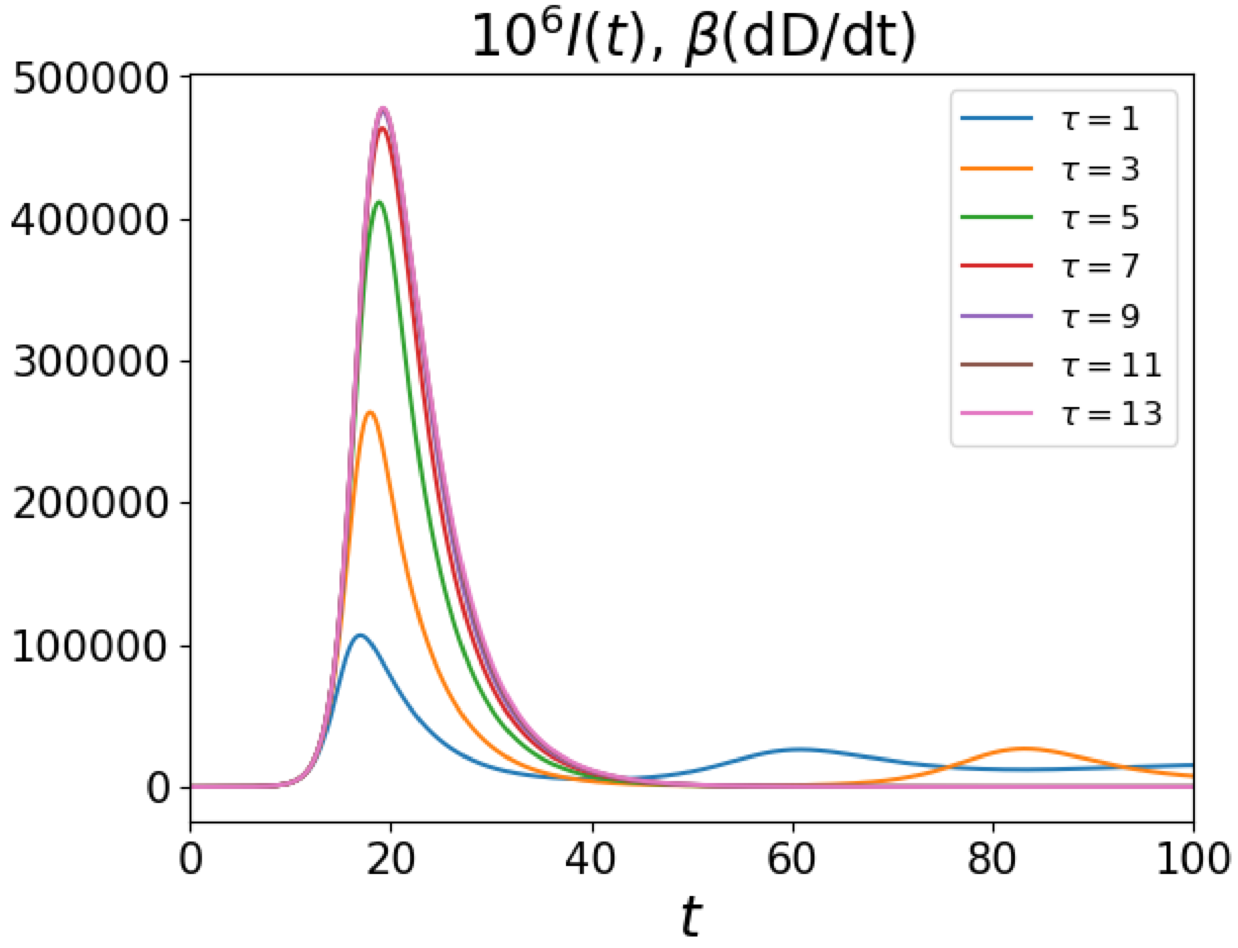


Figure 2.5: This shows the solution for $I(t)$ when α_D is dependent on the current death rate with $\alpha_D \approx 46\,050\,\text{day}$ corresponding to $r = 0.5$ for $\frac{\text{d}D}{\text{d}t} = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days.

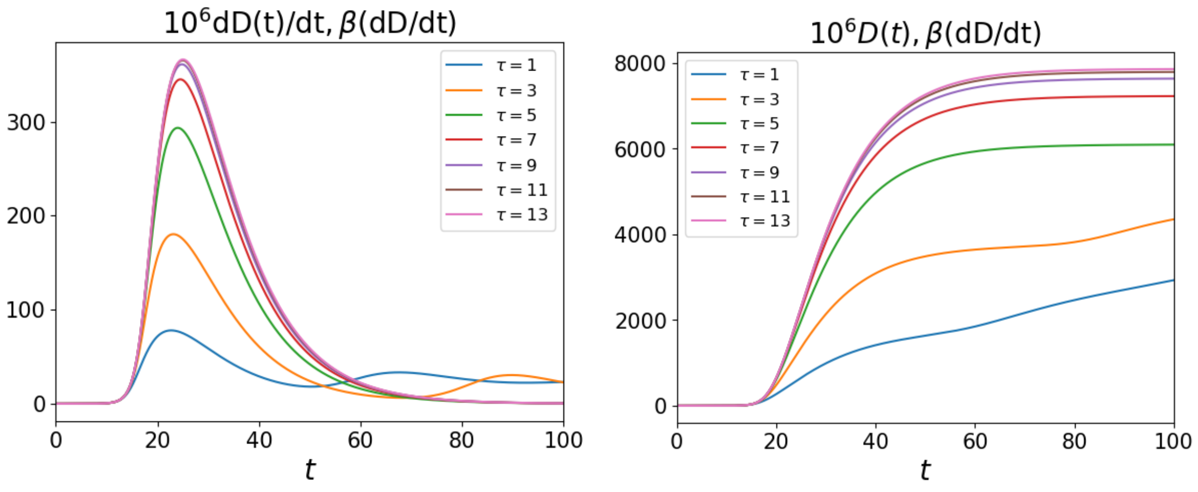


Figure 2.6: These show the solution for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ when α_D is dependent on the current death rate with $\alpha_D \approx 46\,050\,\text{day}$ corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. The figure on the left shows the death rate while the figure on the right shows the cumulative dead. We see that each delay leads to more deaths happening more quickly.

2.1 Random Additions

There are two other concerns I initially had with this model. One is that people are more likely to underestimate r (in a worst case scenario) and so we might expect some random noise above the actual value if people are too willing to underestimate the infectiousness of the disease. Using random number generation to add some number between 0 and 1 onto $\beta(t-\tau)$ will allow us to see if random shifts will induce terrible shifts in the numbers. First let's look at what happens when we use the infectious population to determine β. We see in Figures 2.7 and 2.8 that adding this random factor is essentially the same as increasing R_0 and so we simply see the result asymptote to an $r > 1$ value, which modestly raises the number of infected and dead.

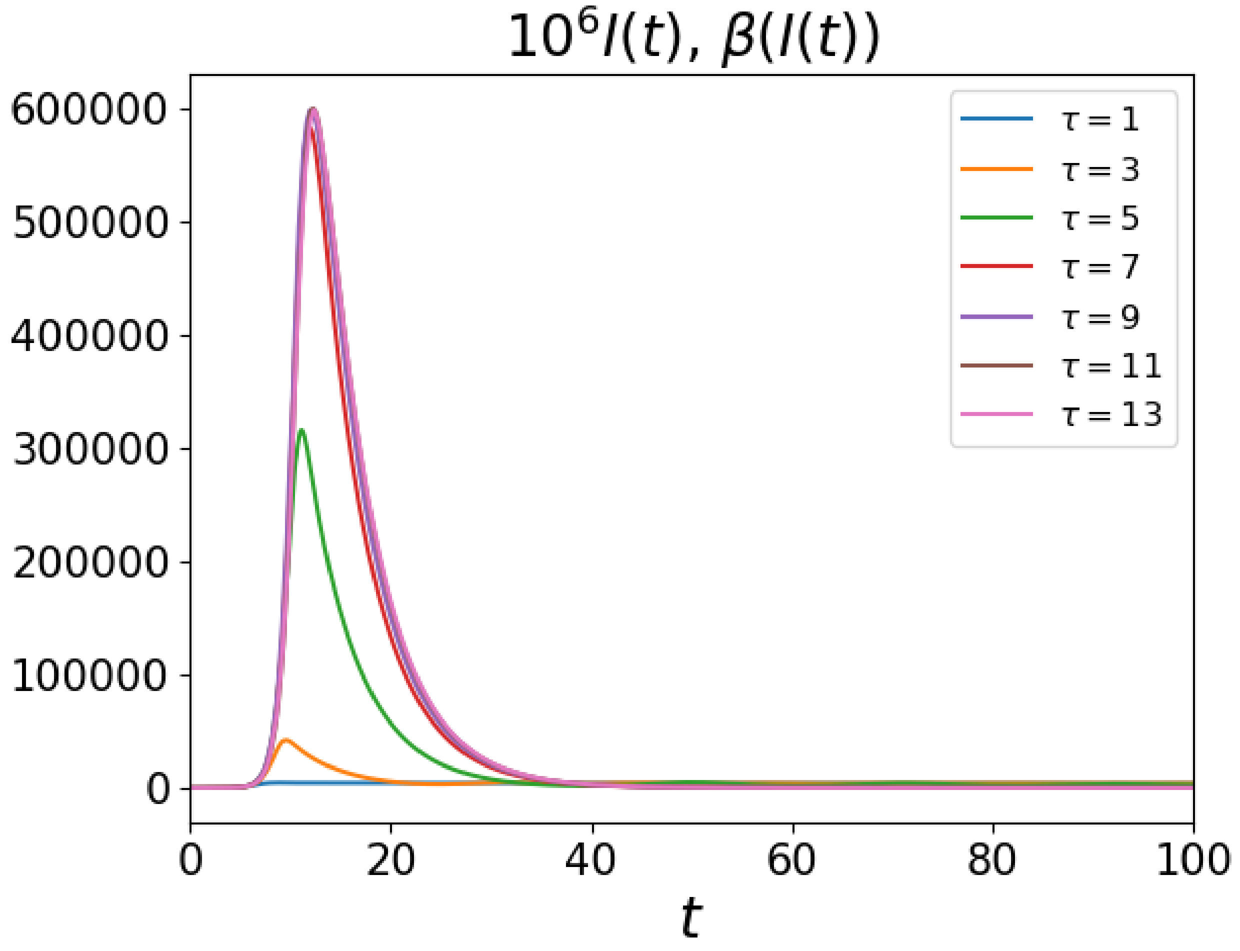
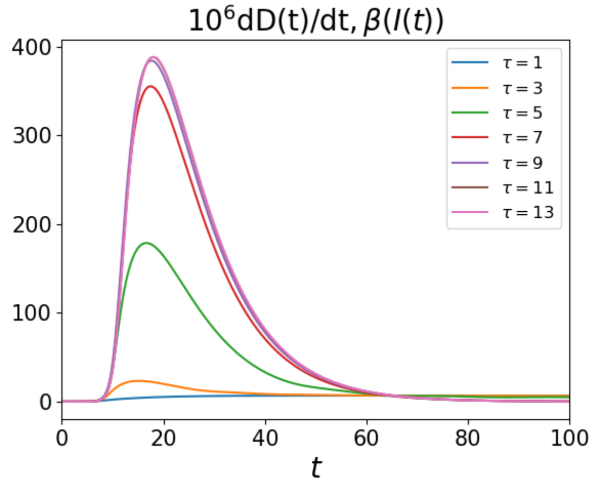


Figure 2.7: This shows the solution for $I(t)$ when α_I is dependent on the infected population with $\alpha_I \approx 460.5$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t - \tau)$ given in units of days and a random number added to β at each time step between 0 and 1. The results are fairly similar to the cases with no randomly larger β.



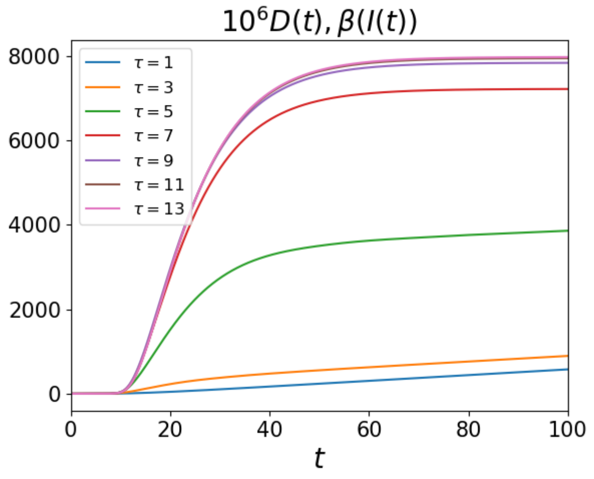


Figure 2.8: Both $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ are shown for α_I dependent on the infected population with $\alpha_I \approx 460.5$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days and a random number added to β at each time step between 0 and 1. The results are fairly similar to the cases with no randomly larger β.

We can perform the same random addition of a number between 0 and 1 to the β value now using the death rate sensitive β. This yields Figures 2.9 and 2.10 which again simply shows a modest increase due to us artificially increasing the r with the random number. This seems to support that so long as people are even somewhat sensitive to the reproduction number of the disease, when r is near 1 there is not too much more death.

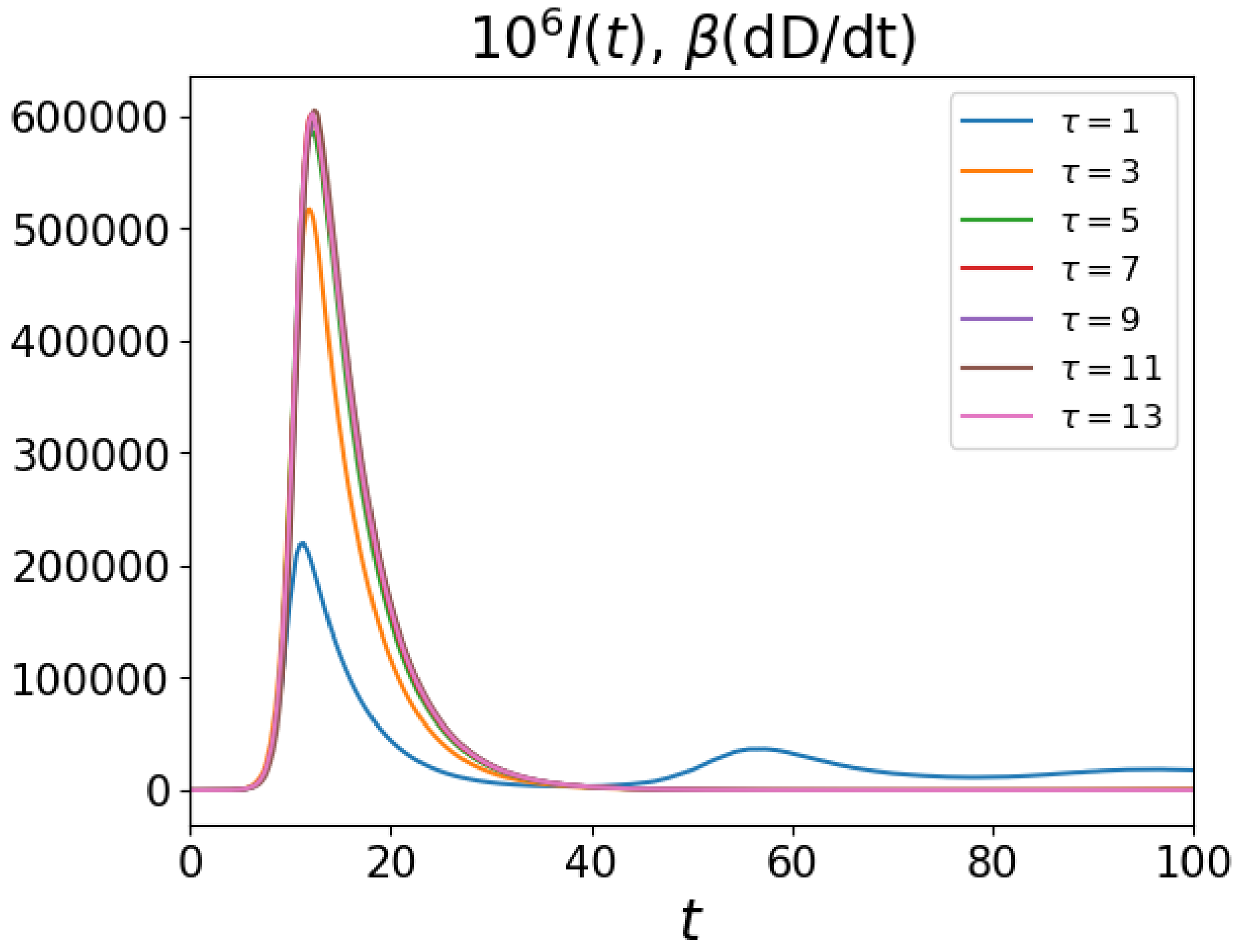


Figure 2.9: The solution of $I(t)$ is shown for α_D dependent on the current death rate with $\alpha_D \approx 46\,050\,\mathrm{day}$ corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days and a random number added to β at each time step between 0 and 1. Again, the results are not substantially different than the model without randomness.

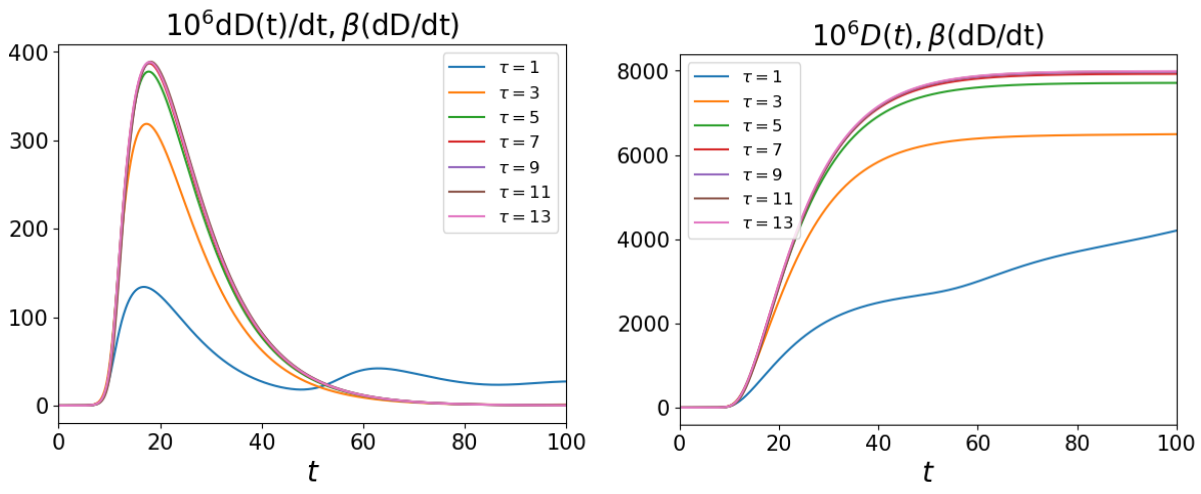


Figure 2.10: This shows the solution for α_D dependent on the current death rate with $\alpha_D \approx 46\,050\,\mathrm{day}$ corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days and a random number added to β at each time step between 0 and 1. The results are not highly impacted by the artificial increasing of β, other than by increasing the effective r.

2.2 Response Functions

Finally, and what I would worry most about is the response function. We have been assuming

$$\beta = \beta_0 \exp\left(-\alpha_X X(t)\right) \tag{2.2.1}$$

for $X(t) = I(t)$ and $X(t) = \frac{\mathrm{d}D}{\mathrm{d}t}$. This means people are adjusting their behavior exponentially to the changes, which might be unrealistic. I am not sure if the literature has a good model for this, but it is definitely worth exploring some other functional dependences. Let's try

$$\beta = \max\Big(\beta_0(1 - C_X[X(t) - X_0]), 0\Big) \tag{2.2.2}$$

$$\beta = \max\Big(\beta_0(1 - E_X[X(t) - X_0]^3), 0\Big) \tag{2.2.3}$$

$$\beta = \beta_0[A_X + B_X \tanh(F_X[X(t) - X_0])] \tag{2.2.4}$$

The first two are testing more gradual responses while the last tanh function tests a rapid shift in response only near the position.

Let's set it so that once again we want $\beta^* = 0.1$ when $I = 5000/10^6$ and $\frac{\mathrm{d}D}{\mathrm{d}t} = 50\,\mathrm{day}^{-1}/10^6$ so simply use X_s for this value. We also want β to be β_0 when $X(t) = 0$. This means $X_0 = 0$ for all of them (we'll treat the tanh separately, however) so that initially we get $\beta = \beta_0$.

$$\beta = \max\Big(\beta_0(1 + C_X X_s), \beta^*\Big) \tag{2.2.5}$$

$$\beta = \max\Big(\beta_0(1 + E_X[X_s]^3), \beta^*\Big) \tag{2.2.6}$$

or

$$C_X = \left(\frac{\beta}{\beta_0} - 1\right)\frac{1}{X_s} \tag{2.2.7}$$

$$E_X = \left(\frac{\beta}{\beta_0} - 1\right)\frac{1}{X_s^3} \tag{2.2.8}$$

The tanh takes a little more thought. It makes more sense to say that β_0 is the value at one end of the tanh and that there is a cutoff point for some X value, beyond which we rapidly change behavior towards a new β value, say β^*. We can set X_0 so that at $X_0 = X_s$ in previous values we get the halfway point. We also want $B_X = \frac{\beta^* - \beta_0}{2}$. Then $A_X = 1 + B_X$ so that we retrieve β_0 when $X(t) \ll -1$. Thus

$$\beta = \beta_0\left[1 + \frac{\beta^* - \beta_0}{2}\left(1 + \tanh(F_X[X(t) - X_s])\right)\right] \tag{2.2.9}$$

Now we want $X(t) = 0$ to yield $\tanh(\cdot) < -0.99$ which implies that $F_X > \tanh^{-1}(0.99)/X_s \approx \frac{2.65}{X_s}$. So $F_I > 530$ and $F_D > 53000$.

I will use $\beta^* = 0.1$ for all the rest of the calculations.

2.2.1 Linear

Linear is surprisingly fairly similar to the exponential behavior when β depends on the infected population $I(t)$. See Figures 2.11 and 2.12. The shape of the number of infected is perhaps a bit narrower, but the overall trends are pretty similar to the exponential model.

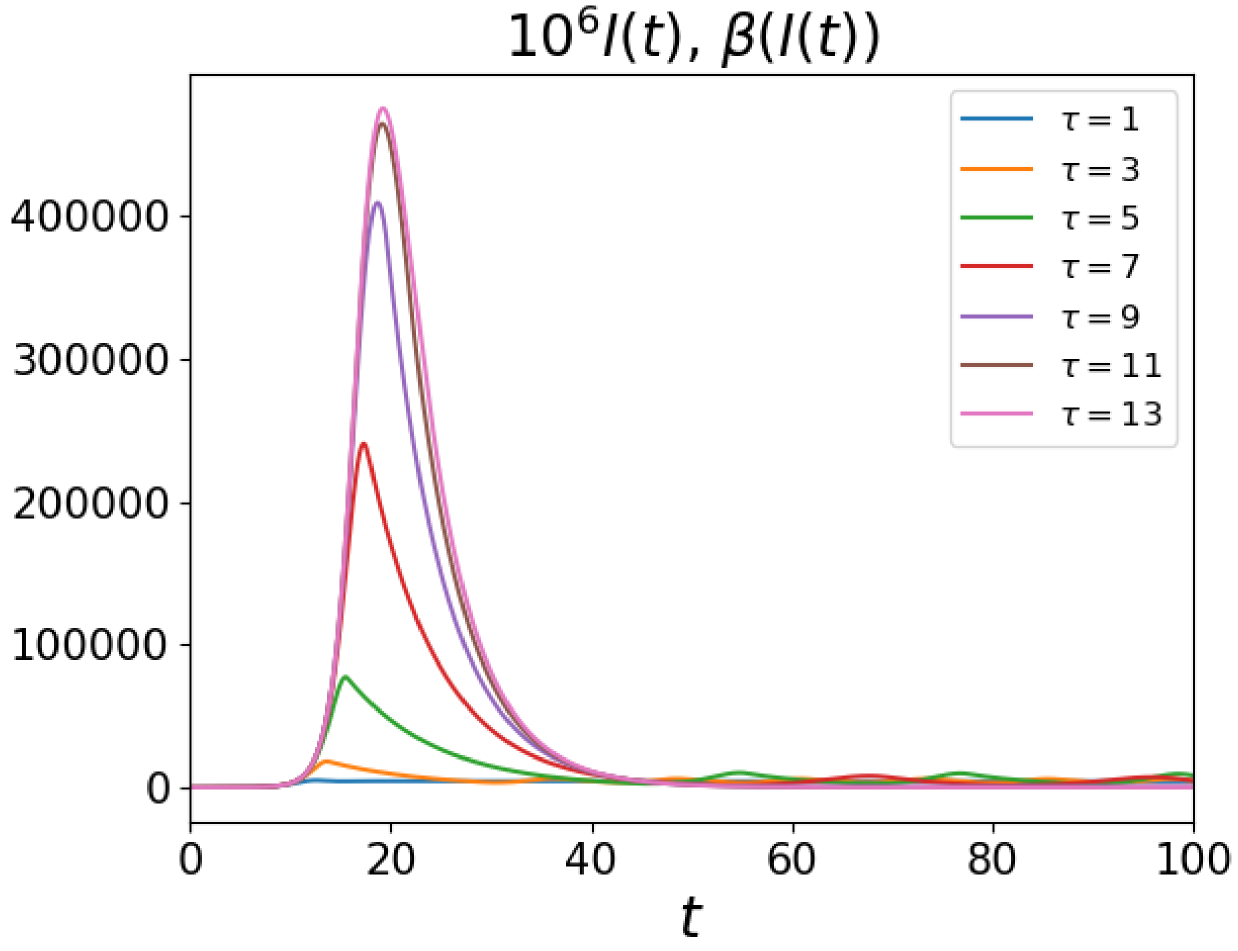


Figure 2.11: This shows the solution for $I(t)$ when using the linear model with C_I dependent on the infected population and $C_I \approx -180$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. We see a linear response is fairly similar to our previous exponential response model.

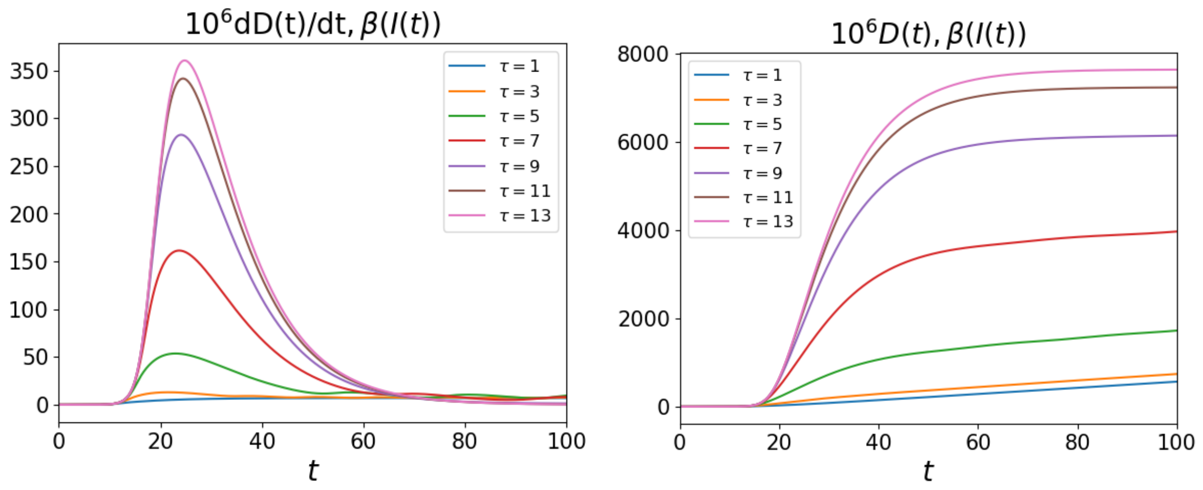


Figure 2.12: Both $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ are shown for C_I dependent on the infected population with $C_I \approx -180$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. The figure on the right simply shows the number of dead. We see that each delay leads to more dead total until the epidemic slows down. We see a linear response is fairly similar to our previous exponential response model.

We can investigate a linear response on the death rate $\frac{\mathrm{d}D}{\mathrm{d}t}$ and find Figures 2.14 2.14 with similar interpretations to our other linear cases. However, we do see significantly more deaths at small delays with the linear response to the death rate, just as we did for the exponential cases.

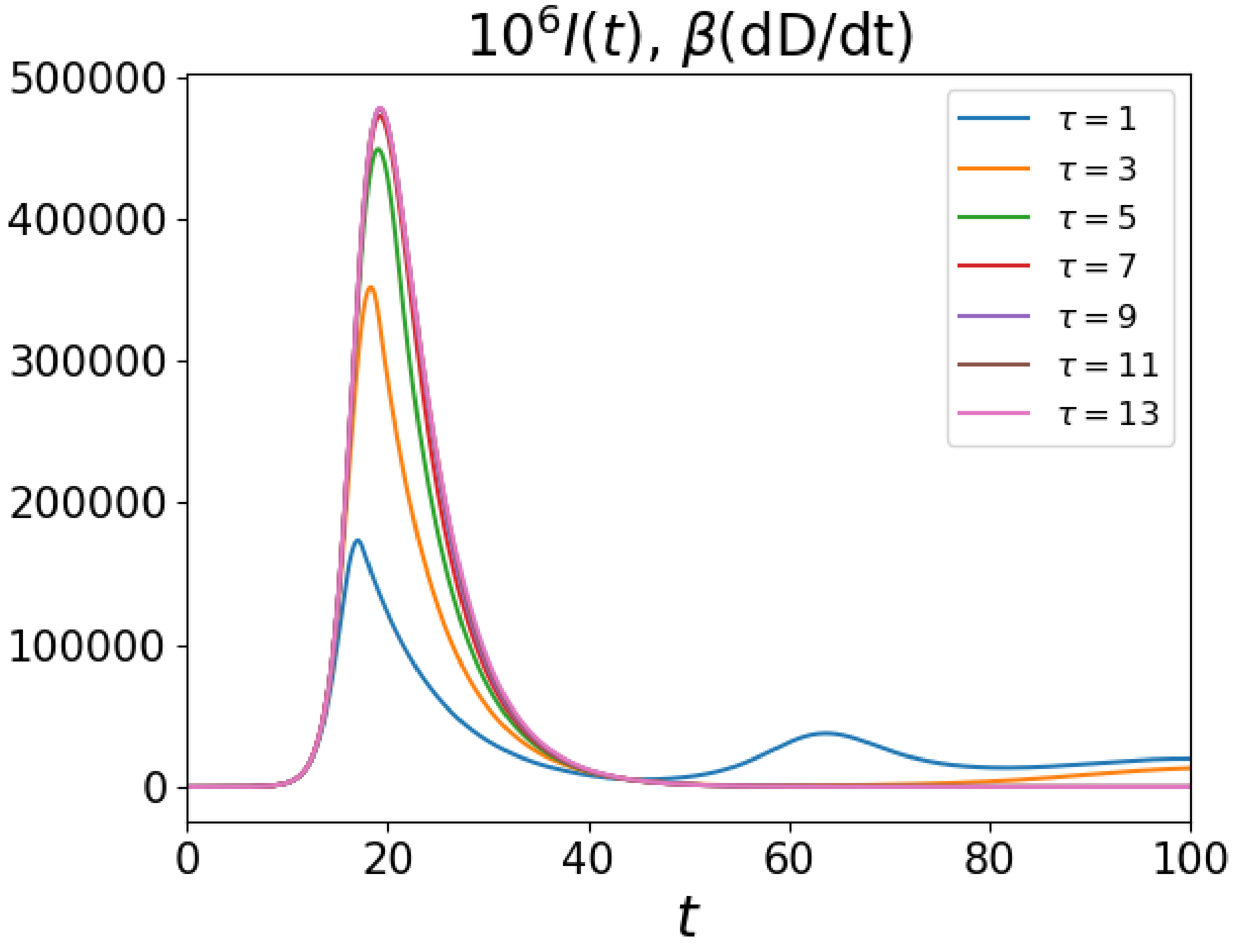


Figure 2.13: This shows the solution for $I(t)$ for C_D dependent on the death rate with $C_D \approx -18\,000$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. We see a linear response is fairly similar to our previous exponential response model.

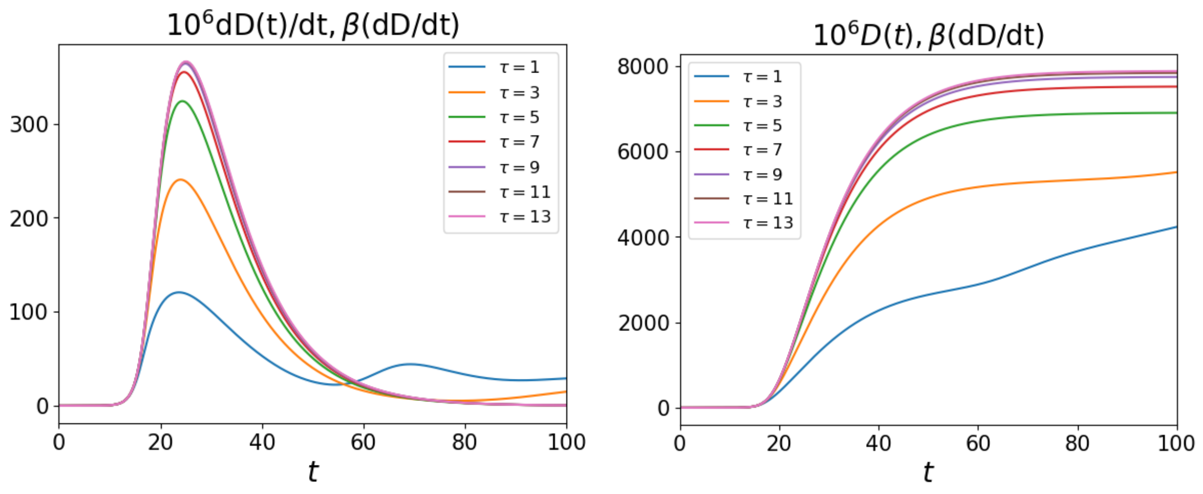


Figure 2.14: This shows the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ when C_D is dependent on the infected population with $C_D \approx -18\,000$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. We see a linear response is fairly similar to our previous exponential response model.

2.2.2 Cubic

The cubic cases with β dependent on $I(t)$ are shown in Figures 2.15 and 2.16. This is fairly similar to the cubic cases, but with an even more peaked looking shape.

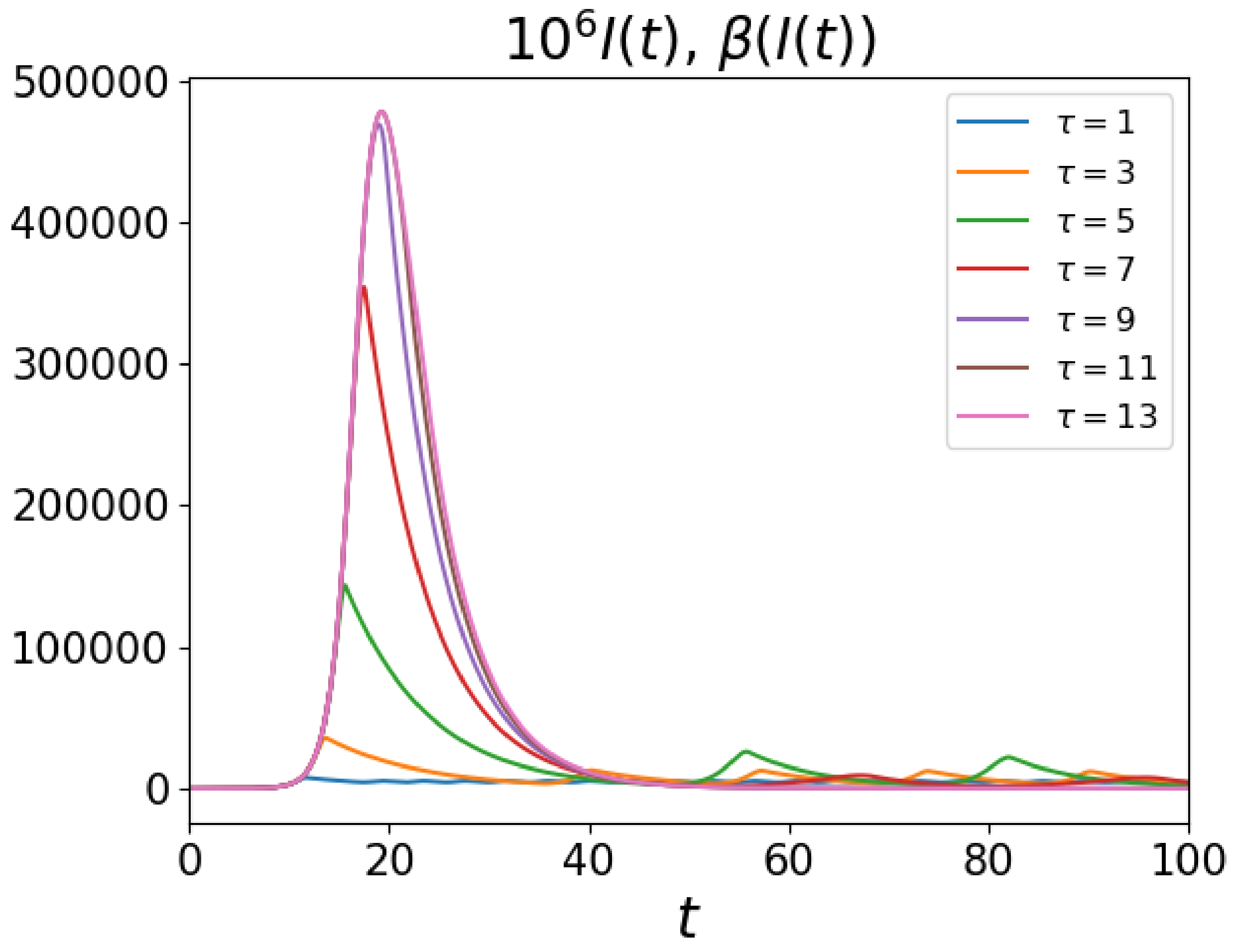


Figure 2.15: This shows the solution for $I(t)$ when E_I is dependent on the infected population with $E_I \approx --7.2 \times 10^{12}$ corresponding to $r = 0.5$ for $I = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t - \tau)$ given in units of days.

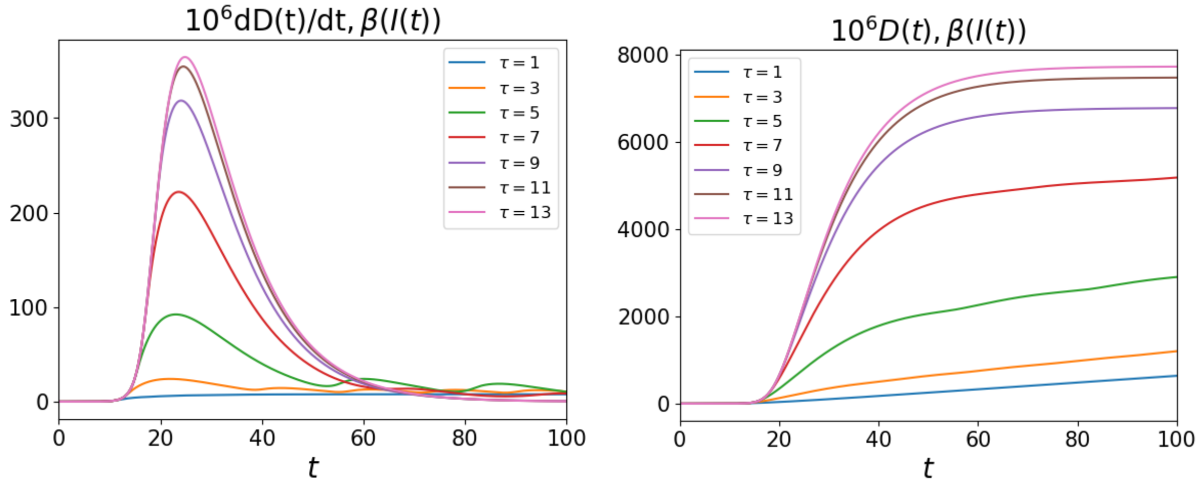


Figure 2.16: Both $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ are shown for E_I dependent on the infected population with $E_I \approx --7.2 \times 10^6$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t - \tau)$ given in units of days. The figure on the right simply shows the number of dead. We see that each delay leads to more dead total until the epidemic slows down. We see a linear response is fairly similar to our previous response model.

The cubic cases with β dependent on $\frac{\mathrm{d}D}{\mathrm{d}t}$ are shown in Figures 2.17 and 2.18. Other than looking a bit more narrow in $I(t)$ the general trends are similar to those for the linear and exponential cases.

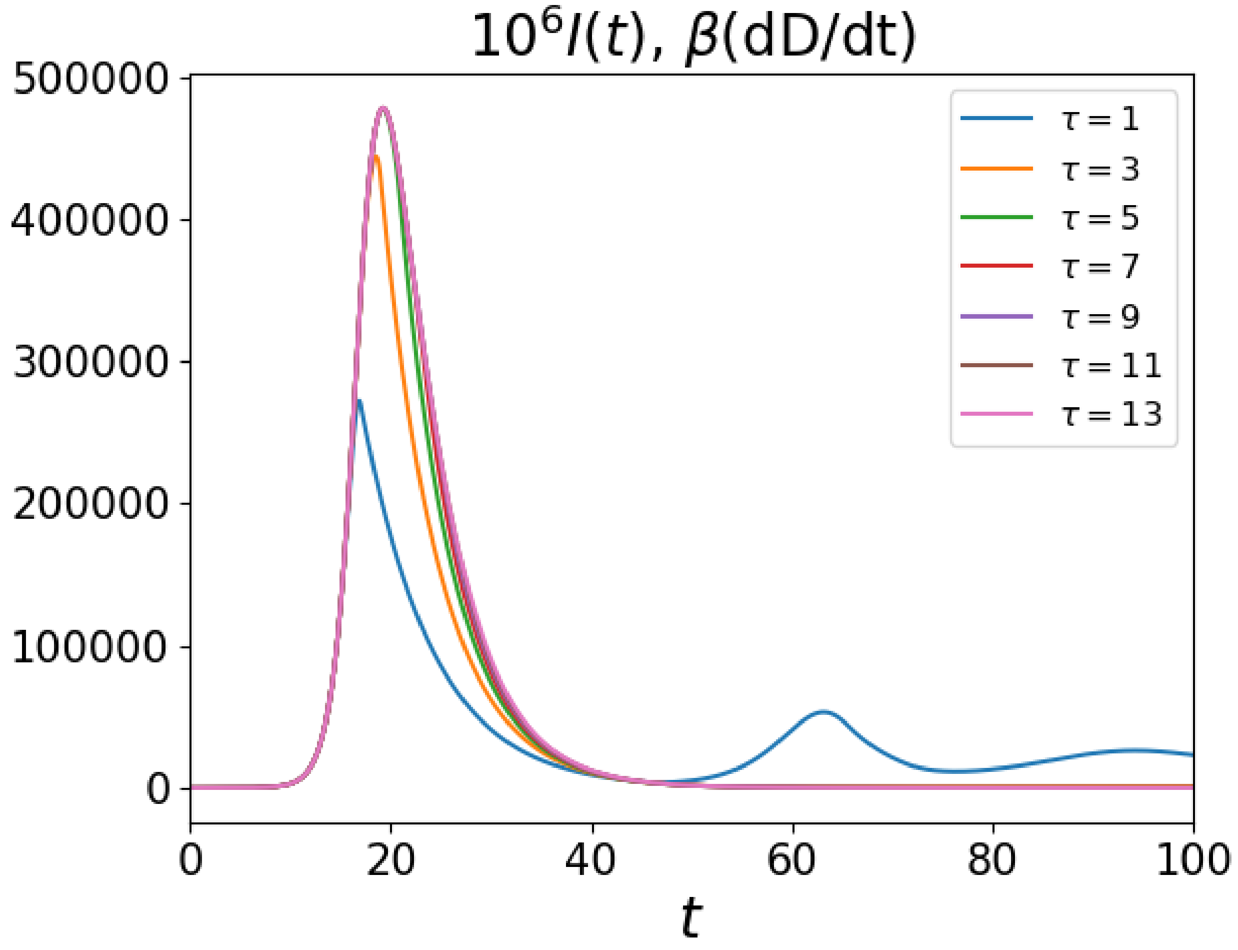


Figure 2.17: This shows the solution $I(t)$ for E_D dependent on the death rate with $E_D \approx -7.2 \times 10^{12}$ corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days.

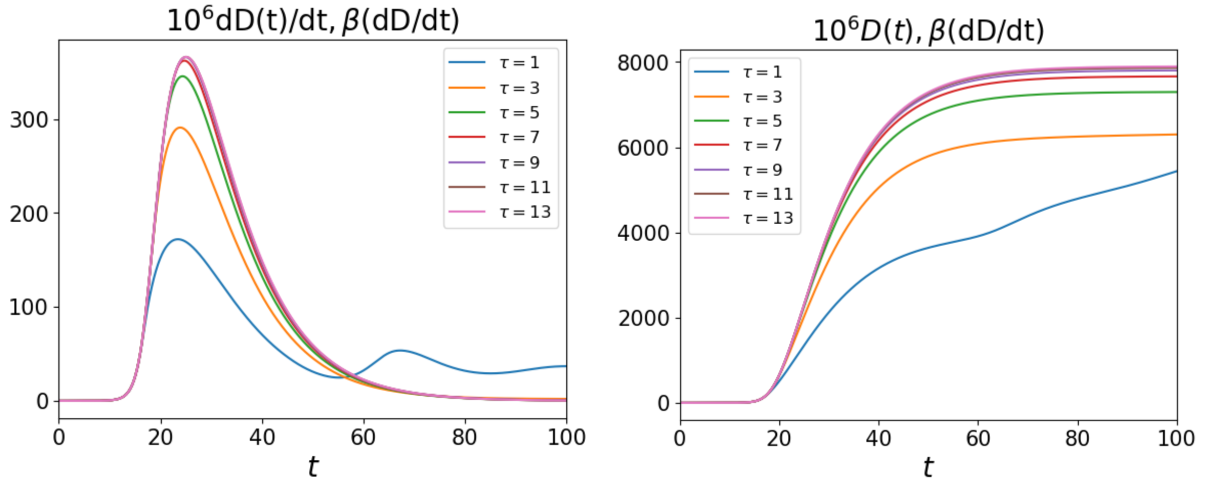


Figure 2.18: These show the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ when E_D is dependent on the death rate with $E_D \approx -7.2 \times 10^{12}$ corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t - \tau)$ given in units of days.

2.2.3 Tanh

Finally, we look at the tanh case. This shows the same characteristics as all other response functions. This suggests that so long as the response function is fairly reasonable, then the conclusion that people changing their behavior will lead to a lessening of deaths and possibly an oscillation in the infection or death rate.

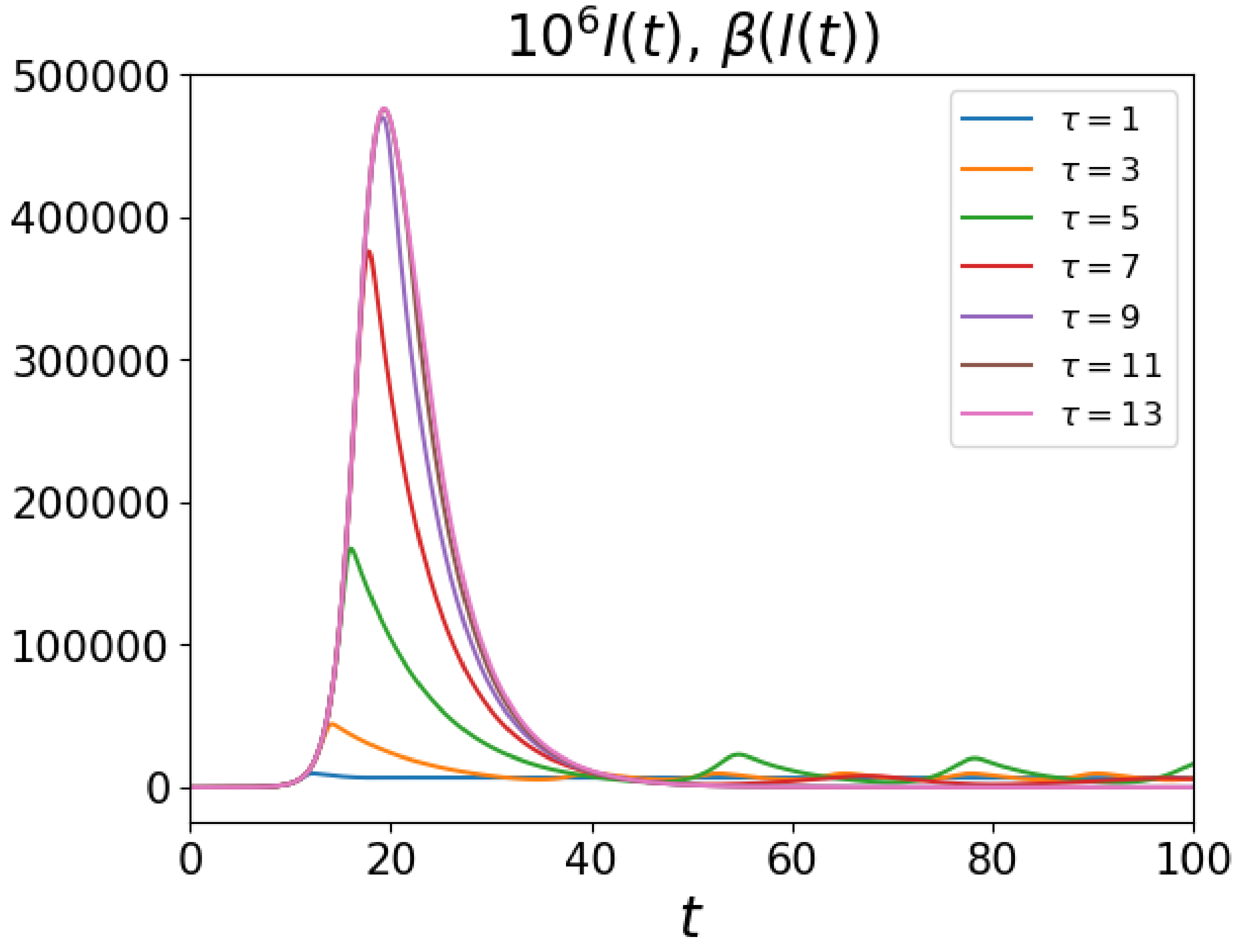


Figure 2.19: This shows the solution for $I(t)$ dependent on the infected population with $F_I \approx 530$ corresponding to $r \approx 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days. We see a linear response is fairly similar to our previous response model.

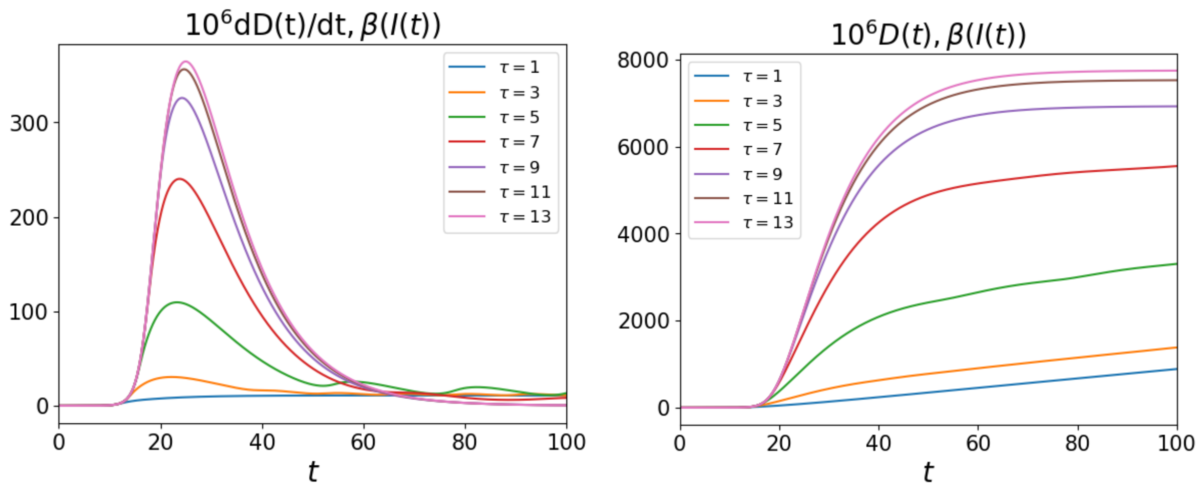


Figure 2.20: The solutions $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ dependent on the infected population with $F_I \approx 530$ corresponding to $r \approx 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t - \tau)$ given in units of days.

The tanh cases with β dependent on $\frac{\mathrm{d}D}{\mathrm{d}t}$ are shown in Figures 2.21 and 2.22.

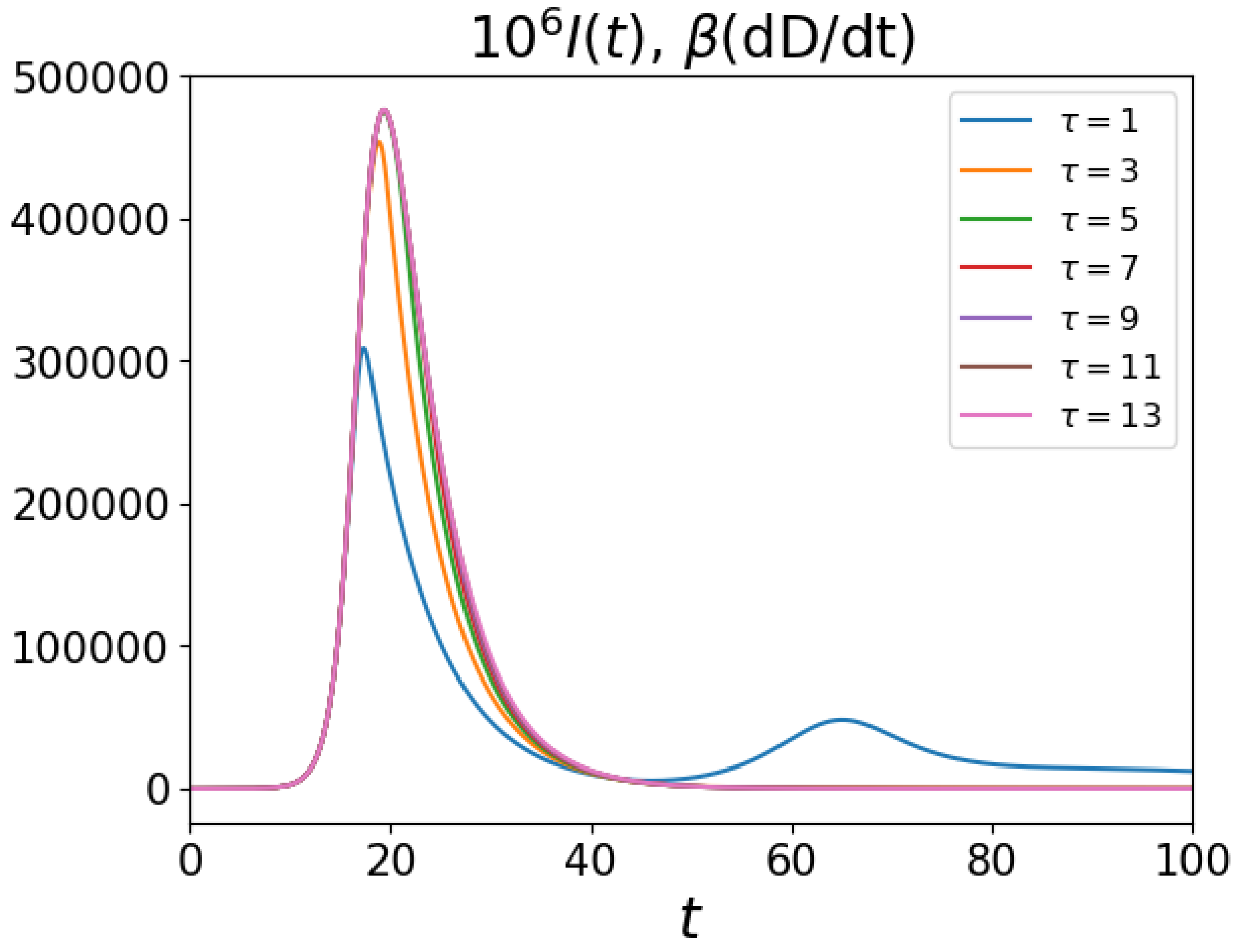


Figure 2.21: This shows the solution for $I(t)$ dependent on the death rate with $F_D \approx 53\,000$ corresponding to $r \approx 0.5$ for $I = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with a various delays τ for $\beta(t-\tau)$ given in units of days.

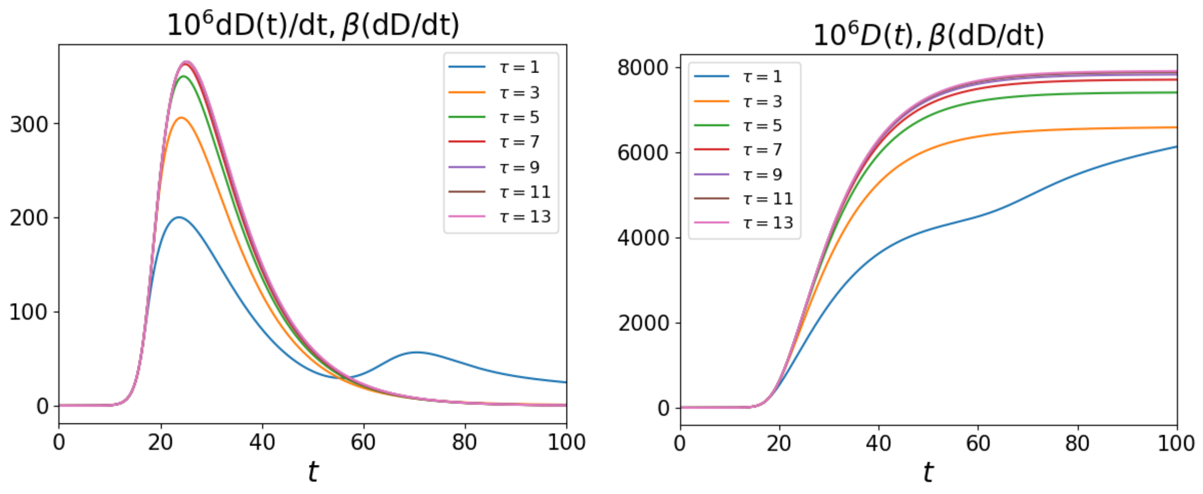


Figure 2.22: The solutions $\mathrm{d}Dt$ and $D(t)$ dependent on the death rate with $F_D \approx 53\,000$ corresponding to $r \approx 0.5$ for $I = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with various delays τ for $\beta(t-\tau)$ given in units of days.

2.2.4 Narrowness of Tanh

Let's consider the most oscillatory case $\tau = 5\,$day for the $I(t)$ dependent β case. We show the tanh functions themselves in Figure 2.26 for various $F = F_I$ values. We then scan the narrowing factor F_I from 530 to 5300 to see how this affects the epidemic breakout. We see the results in Figures 2.24 and 2.25.

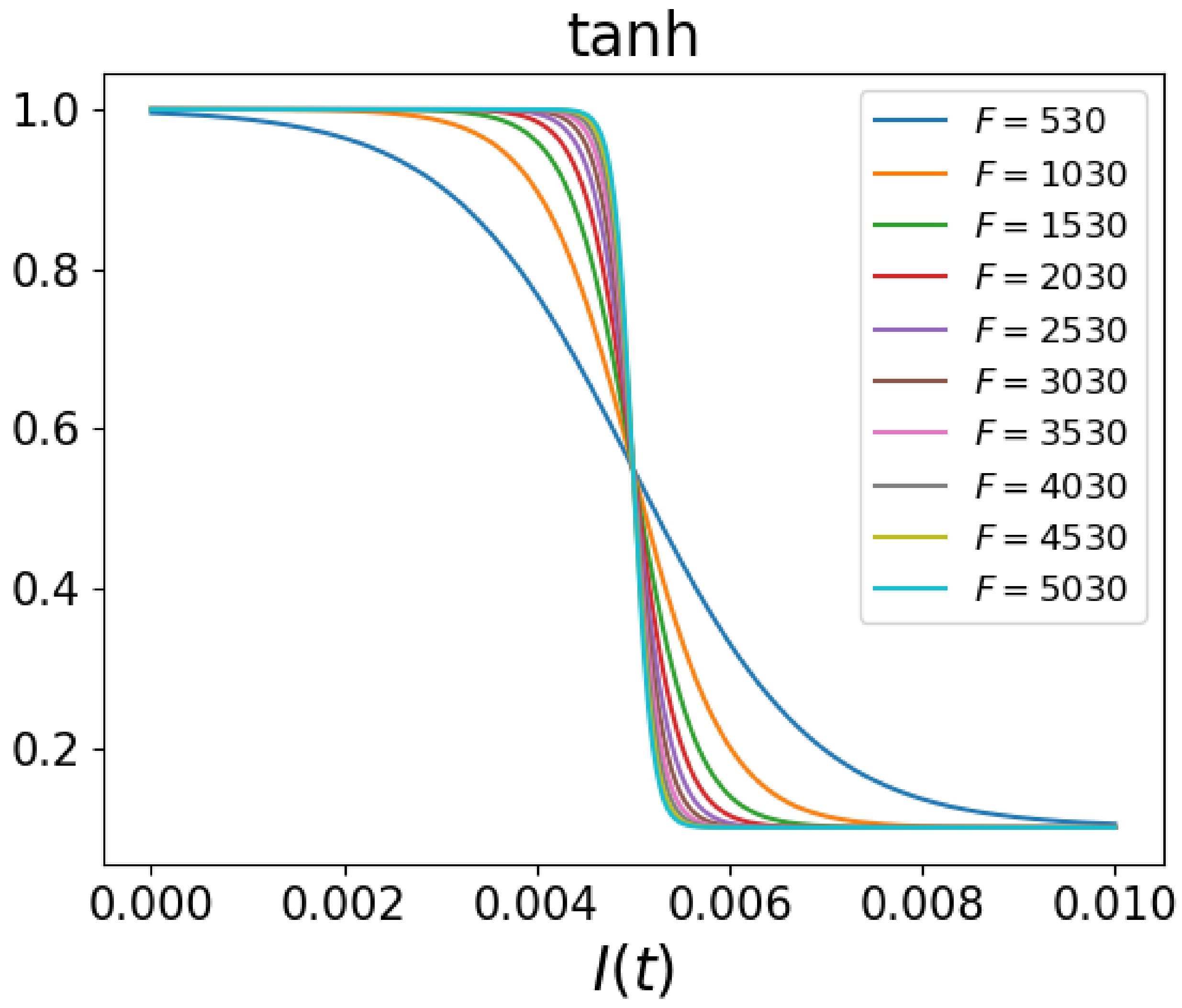


Figure 2.23: These show the tanh profiles for cases with β dependent on $I(t)$ and with $\tau = 5\,\mathrm{day}$.

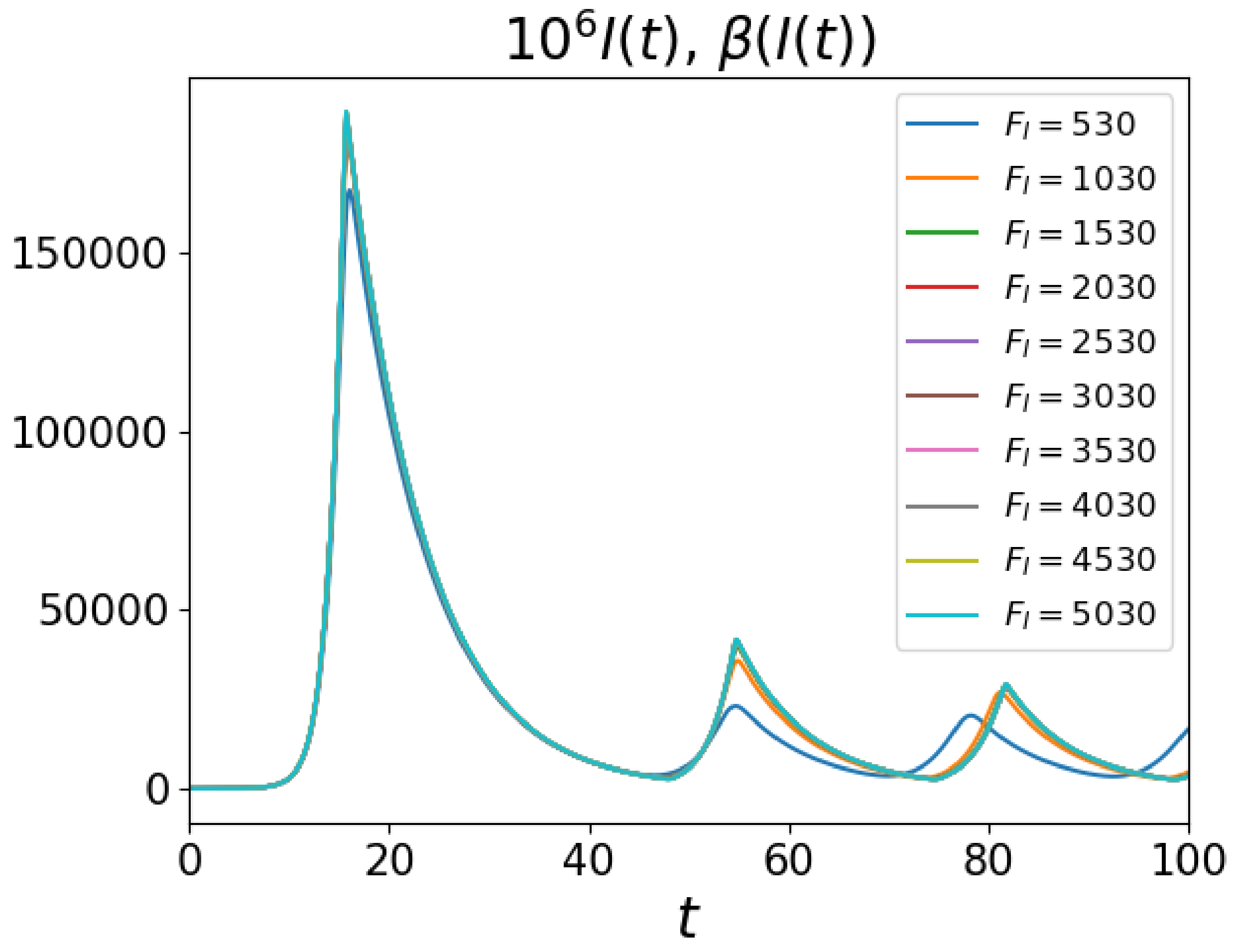


Figure 2.24: The solution $I(t)$ dependent on the infected population with $F = F_I$ varied corresponding to $r \approx 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with $\tau = 5\,\text{day}$. We see that the narrower the tanh (the greater F), the more infected in each cycle.

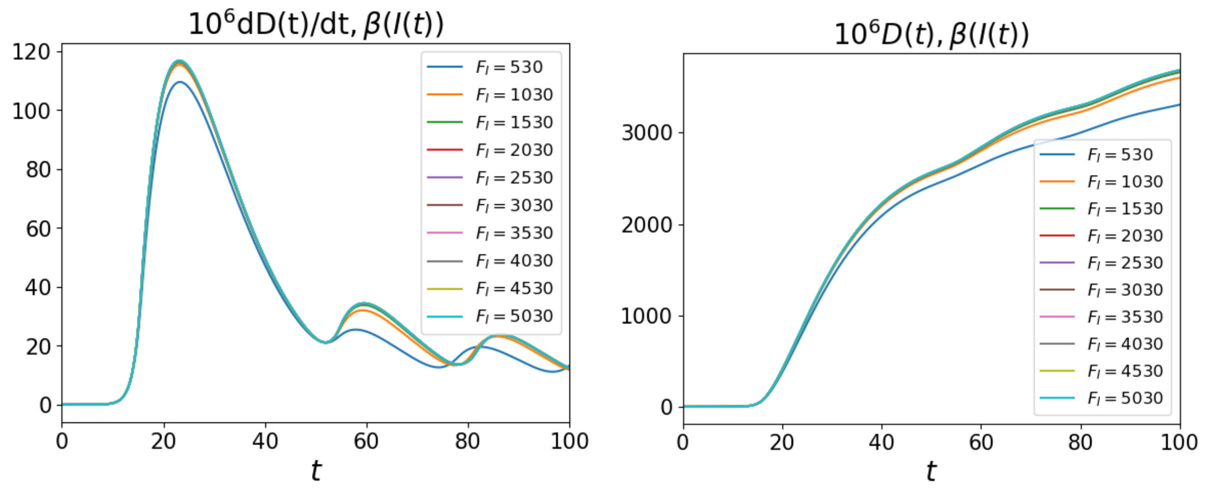


Figure 2.25: This shows the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ dependent on the infected population with $F = F_I$ varying but corresponding to $r \approx 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ for $\tau = 5\,\text{day}$. So we find that the narrower the tanh (the greater F), the more deaths there are though not substantially different.

Let's consider the most oscillatory case we have previously looked at, $\tau = 1\,\text{day}$ for the $\frac{\mathrm{d}D}{\mathrm{d}t}$ dependent β case. We then scan the narrowing factor F_D from 53 000 to 530 000 to see how this affects the epidemic breakout. We can see the profiles in Figure 2.26. We see the results in Figures 2.27 and 2.28.

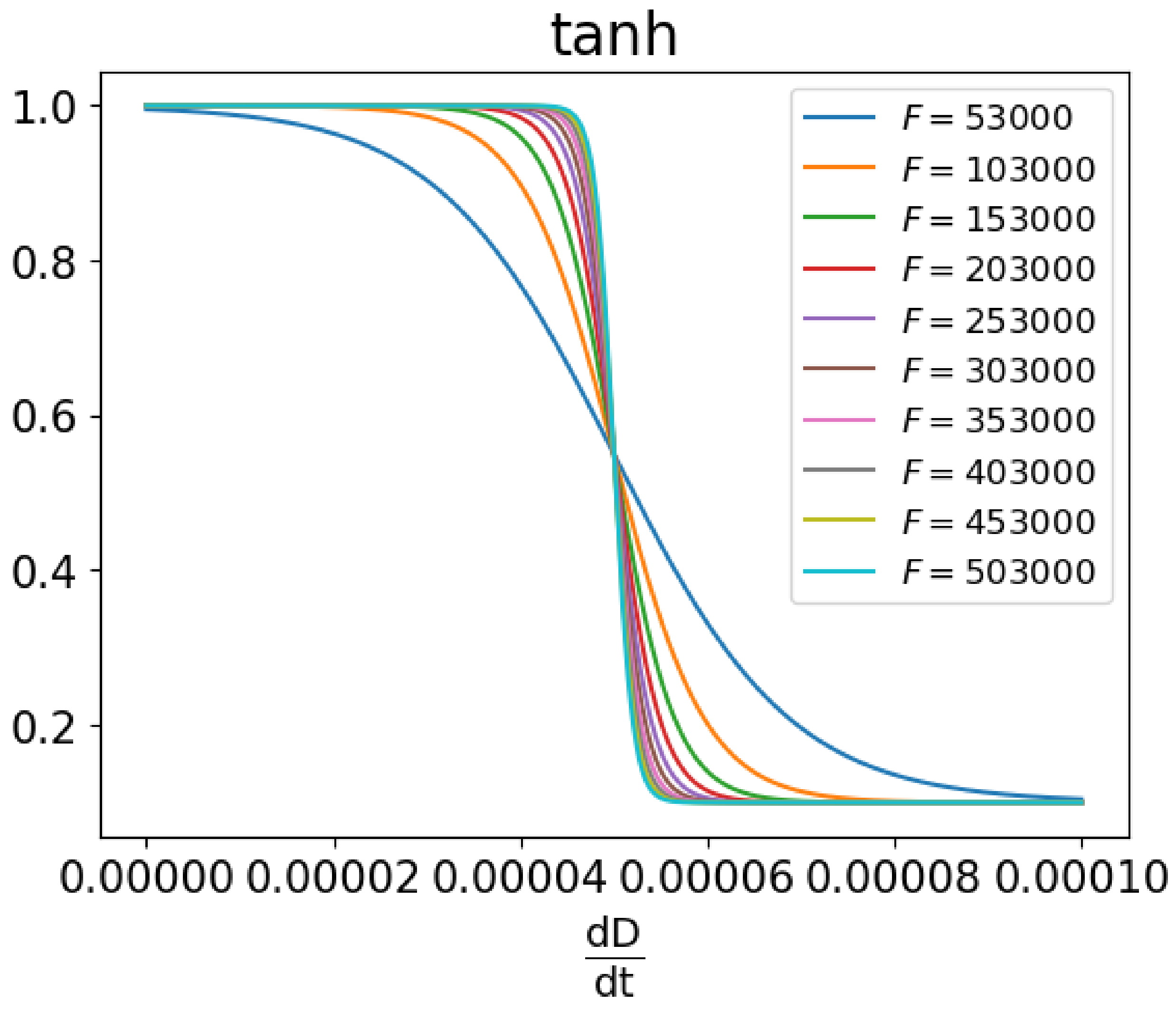


Figure 2.26: These show the tanh profiles using $\frac{\mathrm{d}D}{\mathrm{d}t}$ for $\tau = 1\,\text{day}$.

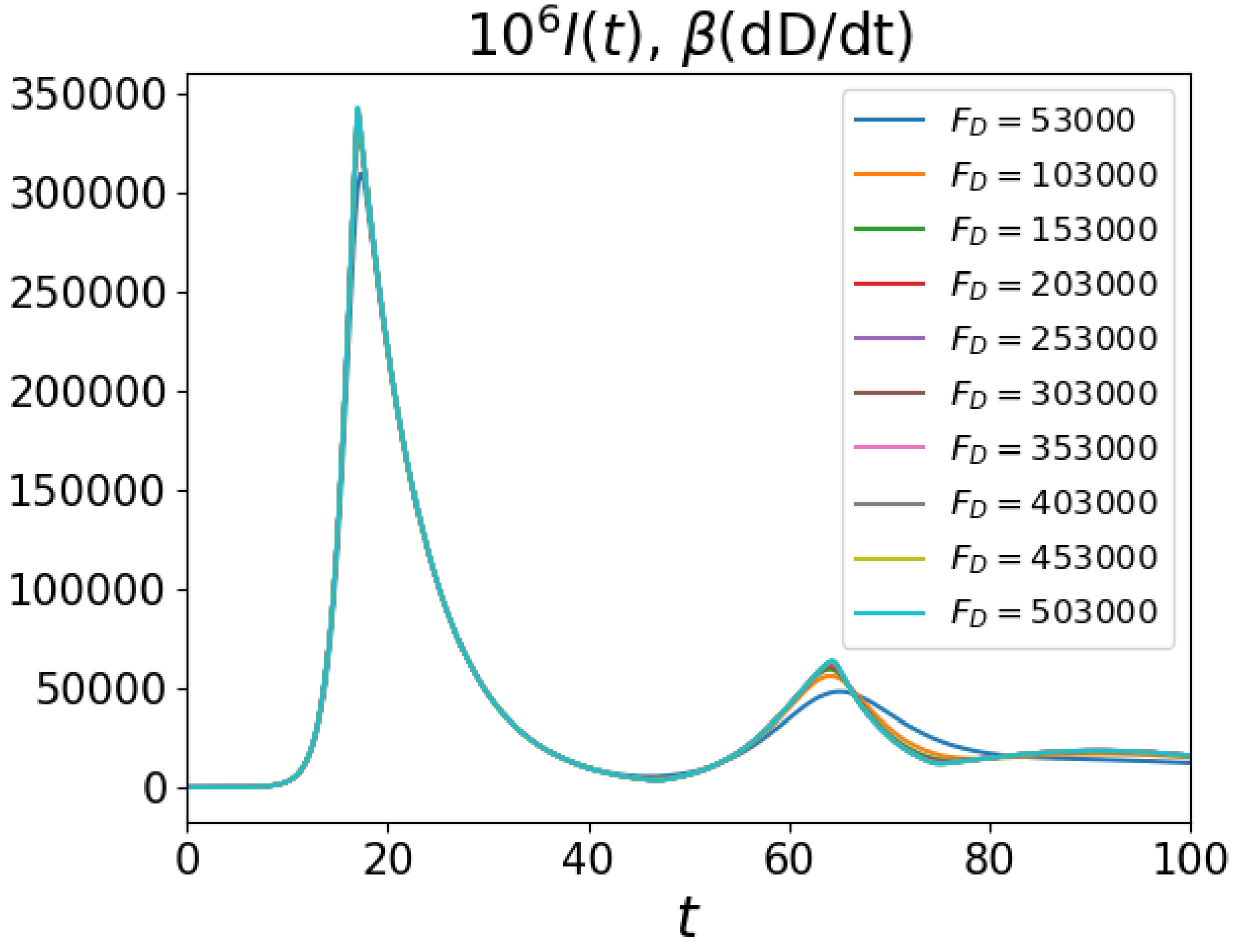


Figure 2.27: This shows $I(t)$ dependent on the death rate with $F = F_D$ varied corresponding to $r \approx 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with $\tau = 1\,\mathrm{day}$. Again, we find narrower tanh (greater F) corresponds to more infected.

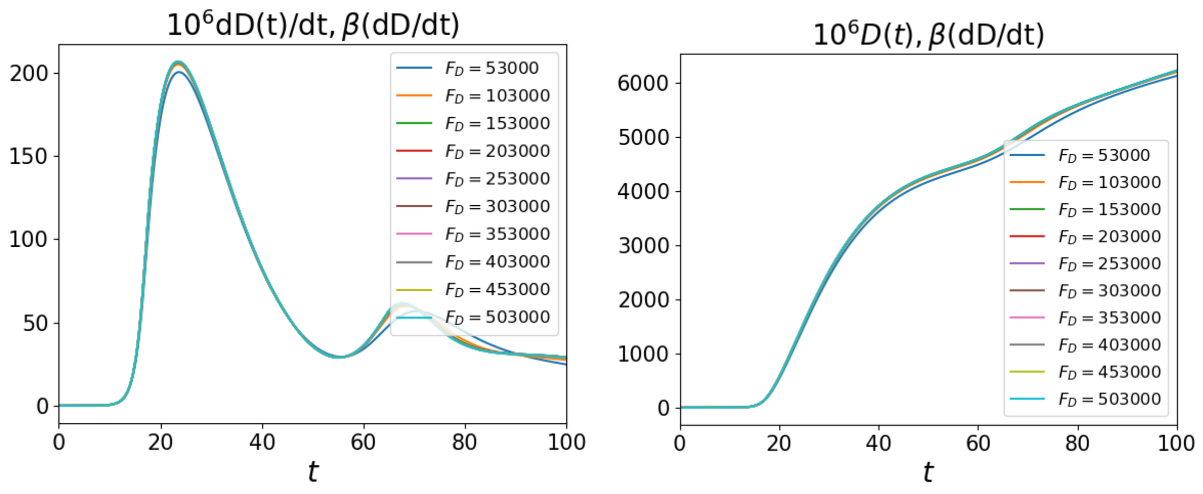


Figure 2.28: The solutions $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ dependent on the infected population with $F = F_D$ varied corresponding to $r \approx 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with $\tau = 1\,\mathrm{day}$ are shown. Once again, narrower tanh (greater F) indicates more overall deaths.

2.2.5 Cross Comparison

Let's now compare the different response functions for I with time delay $\tau = 5$ day. We find that the less gradual the response function (for similar parameters), the smaller the death rate and so number of deaths.

We can start by looking at what responses I have allowed. These are shown in Figure 2.29. The solutions are shown in Figures 2.30 and 2.31.

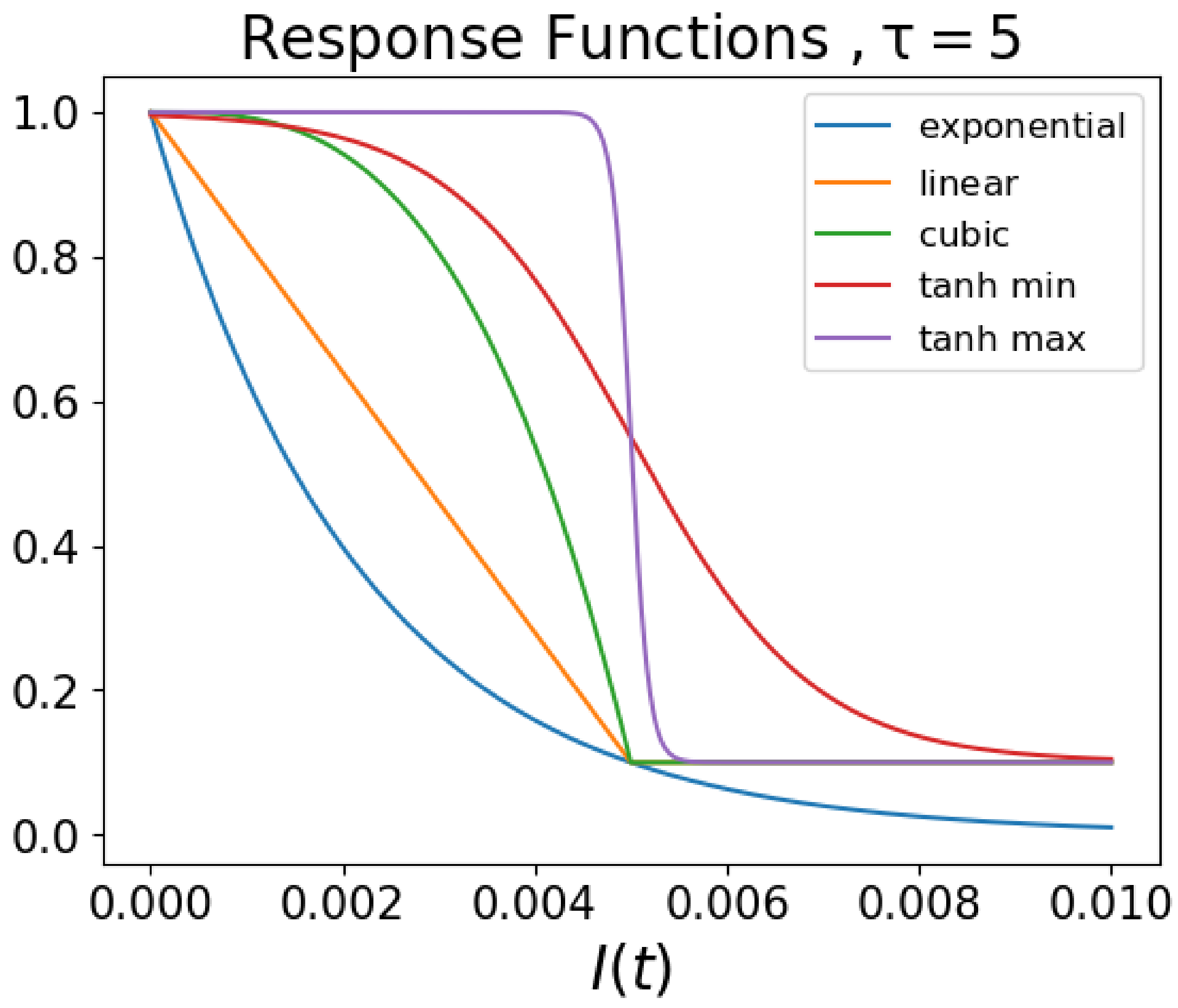


Figure 2.29: These show the tanh profiles using $I(t)$ for $\tau = 5$ day for various response functions.

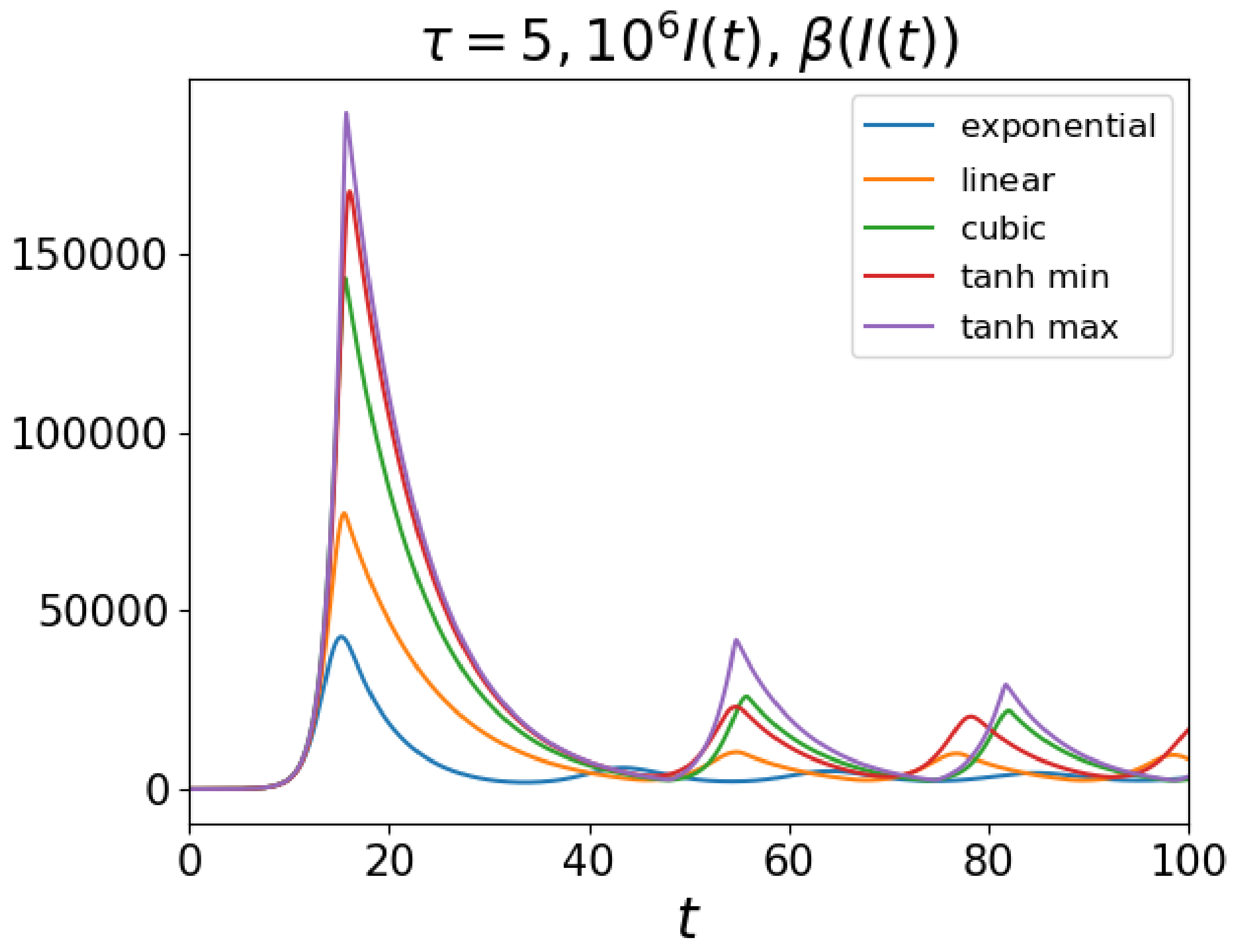


Figure 2.30: This shows the solution $I(t)$ dependent on the infected population for all the various response models corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with $\tau = 5\,\text{day}$ and the tanh max corresponding to $F = 5300$ and tanh min corresponding to $F = 350$. We see the more gradual the response function, the fewer infected and dead.

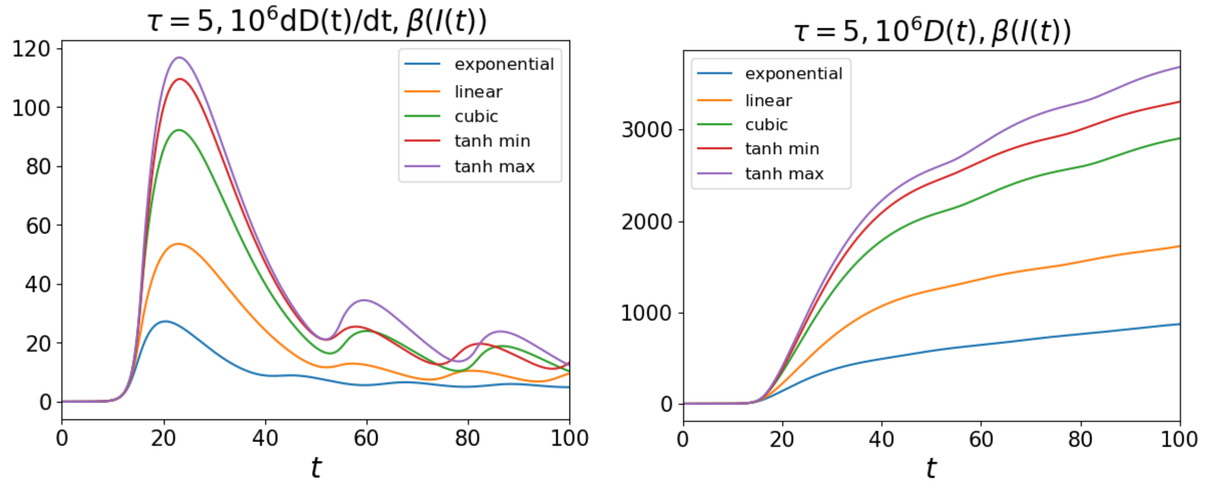


Figure 2.31: This shows the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ dependent on the infected population for all the various response models corresponding to $r = 0.5$ for $I = 5000/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with $\tau = 5\,\text{day}$ and the tanh max corresponding to $F = 5300$ and tanh min corresponding to $F = 350$. We see the more gradual the response function, the fewer infected and dead.

Similar conclusions for when based on death rate. The response functions are shown in Figure 2.32. The solutions are shown in Figures 2.33 and 2.34.

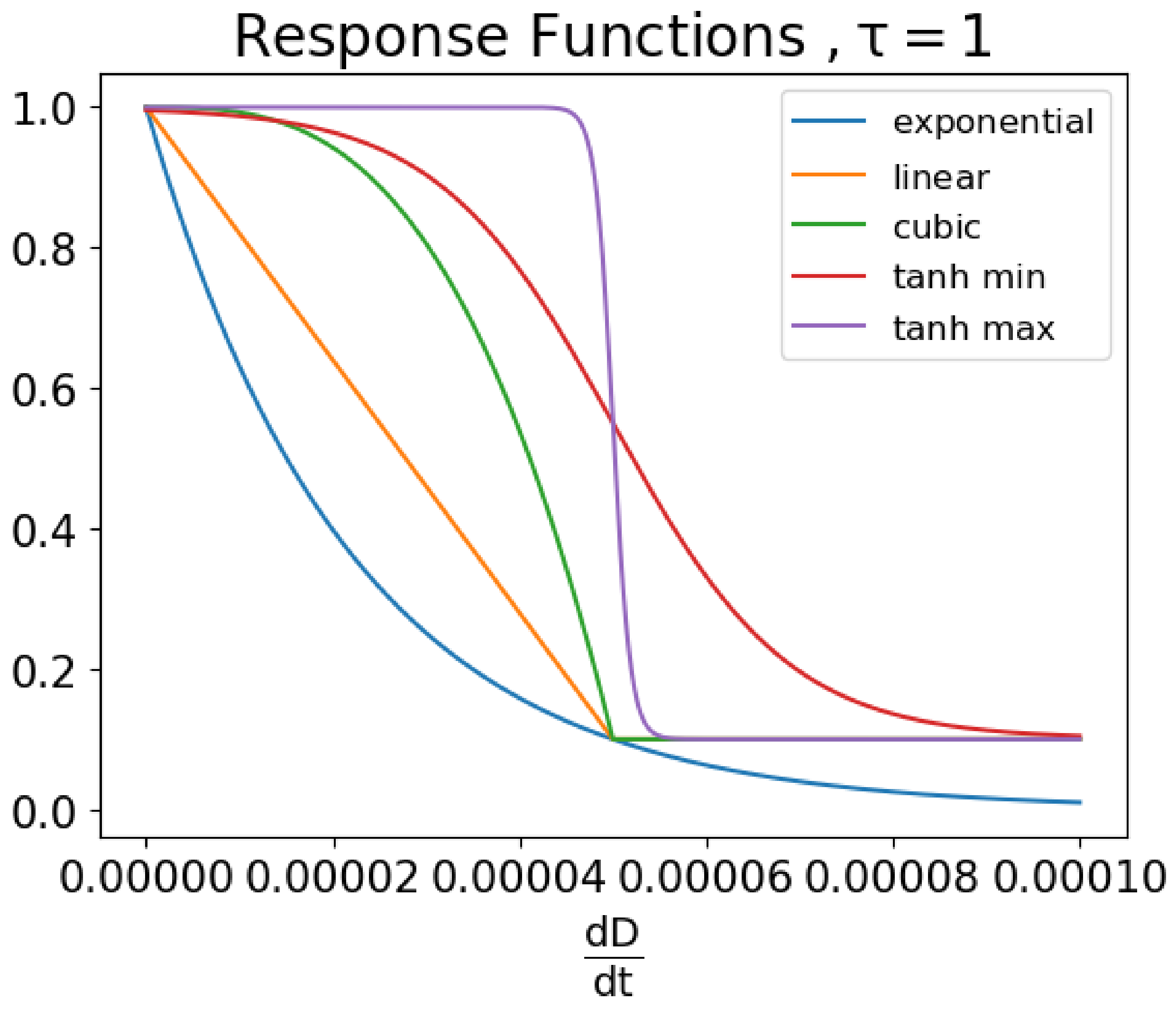


Figure 2.32: These show the tanh profiles using $\frac{\mathrm{d}D}{\mathrm{d}t}$ for $\tau = 1\,$day for various response functions.

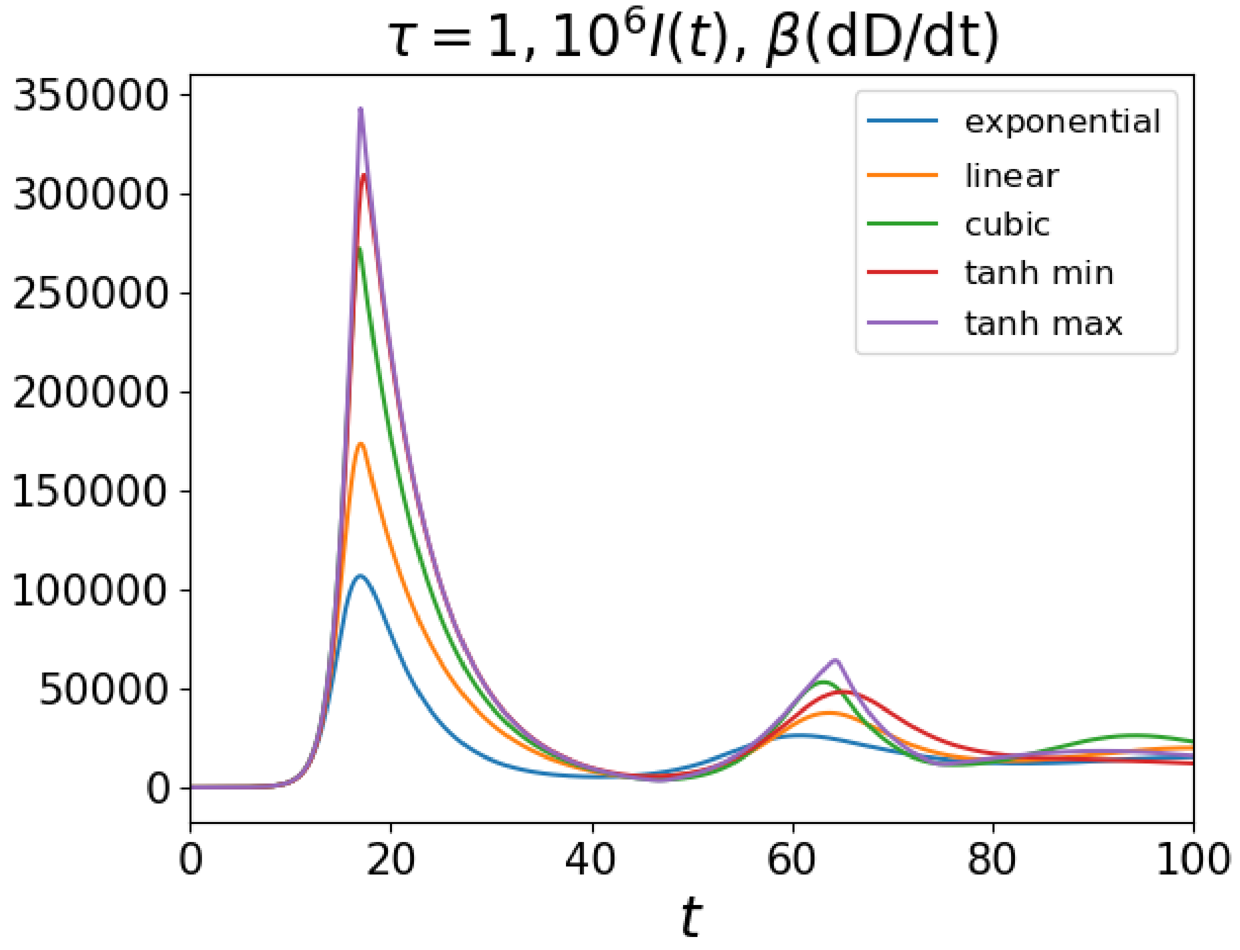


Figure 2.33: This shows the solution $I(t)$ dependent on the death rate for all the various response models corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\mathrm{day}^{-1}$, $\gamma = 0.2\,\mathrm{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\mathrm{day}^{-1}$ with $\tau = 1\,\mathrm{day}$ and the tanh max corresponding to $F = 530\,000$ and tanh min corresponding to $F = 35\,000$.

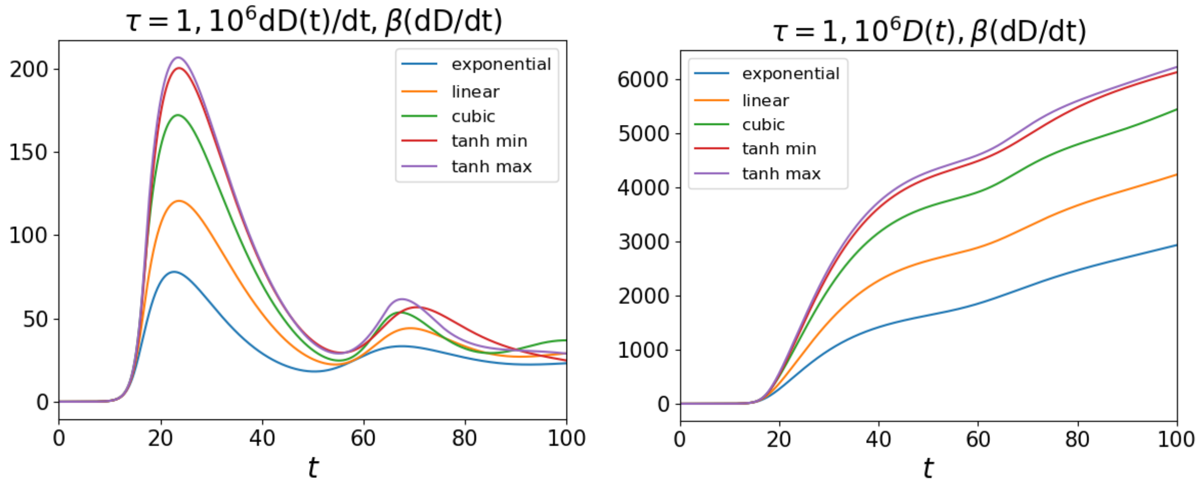


Figure 2.34: This shows the solutions for $\frac{\mathrm{d}D}{\mathrm{d}t}$ and $D(t)$ dependent on the death rate for all the various response models corresponding to $r = 0.5$ for $\frac{\mathrm{d}D}{\mathrm{d}t} = 50/10^6$, $\beta_0 = 1\,\text{day}^{-1}$, $\gamma = 0.2\,\text{day}^{-1}$, $r_0 = 5$, $\theta = 0.1\,\text{day}^{-1}$ with $\tau = 1$ day and the tanh max corresponding to $F = 530\,000$ and tanh min corresponding to $F = 35\,000$. We see the more gradual the response function, the fewer infected and dead.

Chapter 3

Conclusions

The main takeaways from these investigations is that the response function can make a difference with the more sensitive people are to a specific death rate leading to fewer deaths. If it is essentially a switch, then a great deal more deaths will occur. We also see that the larger the delay in the information, the more people get infected and die. For behavior based on the perceived number of infected, it is important that the delay not be much beyond 5 day or else there is a much larger number of deaths, before hitting an equilibrium. For behavior based on the death rate, every day of delay in response rapidly increases the number that die. Thus, if one is using the death rate, the more up-to-date the information, the better the response in this model.

There are a number of limitations. The assumption that everyone in a population of S, I, R, D, C act exactly the same is completely unrealistic. It is known that super spreaders may be the largest problem for disease spread. One can argue with whether the response functions chosen are the best to compare against, but that can be easily fixed by implementing any response function suggested. Finally, using continuous variables for a discrete problem will create some errors, but probably won't strongly affect the model so long as the population being modeled has over hundred thousand people or so. In addition, the code used to model everything has not been extensively benchmarked. It is converging with expected characteristics, but there could still be a bug somewhere in the code.

I will emphasize again that this is simply a toy model that helps us understand how possible different behavioral responses could affect the spread of a disease. It is a very simplistic model and so any difficult to understand results should be tested against more advanced models.

Chapter 4

Python Code

SIRDC DDE Solver

```
#/bin/env python3

import numpy as np
import scipy.special as scsp
import matplotlib.pyplot as plt
import scipy.interpolate as scin
import numpy.random as npr

# set seed to be reproducible
npr.seed(1)
############################################################
# lin_interp function:  linear interpolation
#     input :
#       y    : (two reals) pair of values of solution
#                 (to be interpolated)
#       t    : (two reals) times for both y values
#       tdes : (real) time to be interpolated to
#     output : (real) interpolated value
############################################################
def lin_interp(y,t,tdes):
  y0=y[0]
  y1=y[1]
  t0=t[0]
  t1=t[1]
  return (y0*(t1-tdes)+y1*(tdes-t0))/(t1-t0)

############################################################
# cubic_spline_interp function:  spline interpolation
#     input :
#       y    : (two reals) pair of values of solution
#                 (to be interpolated)
#       t    : (two reals) times for both y values
#       k1   : (real) derivative of y[0]
#       k2   : (real) derivative of y[1]
#       tdes : (real) time to be interpolated to
#     output : (real) interpolated value
############################################################
def cubic_spline_interp(y,t,k1,k2,tdes):
  y0=y[0]
  y1=y[1]
  t0=t[0]
  t1=t[1]
  s=(tdes-t0)/(t1-t0)
  a=k1*(t1-t0)-(y1-y0)
  b=-k1*(t1-t0)+(y1-y0)
  return (1-s)*y0+s*y1+s*(1-s)*(a*(1-s)+b*s)

############################################################
```

```
# herm_interp function:  general hermite interpolation
#     input :
#       c    : (list) list of points where we want
#                 interpolated values
#       t    : (list) list of data points we have
#       y    :  (list) derivative of y[0] (same size as t)
#       yp   : (list) derivative of y[1] (same size as t)
#     output : (list) interpolated values
###################################################################
def herm_interp(c,t,y,yp):
  n=len(t)          # number of interpolating points
  k=len(c)          # number of discrete data points
  li=np.ones((n,k))         # Lagrange basis polynomials
  a=np.zeros((n,k))         # basis polynomials alpha(x)
  b=np.zeros((n,k))         # basis polynomials beta(x)
  H=np.zeros((1,k))         # Hermite interpolation polynomial H(x)
  for i in range(n):
    dl=0;             # derivative of Lagrange basis
    for j in range(n):
      if j!=i:
        dl=dl+1/(t[i]-t[j])
        li[i,:]=li[i,:]*(c-t[j])/(t[i]-t[j]);
    l2=li[i,:]**2
    b[i,:]=(c-t[i])*l2           # basis polynomial alpha(x)
    a[i,:]=(1.-2*(c-t[i])*dl)*l2  # basis polynomial beta(x)
    H=H+a[i,:]*y[i]+b[i,:]*yp[i]  # Hermite polynomial H(x)
  return H

###################################################################
# rhs function: right hand side of equation
#     input :
#       t      : time
#       coeffs : array of coefficients for DDE
#                 beta0=coeffs[0]
#                 gamma=coeffs[1]
#                 theta=coeffs[2]
#                 delta=coeffs[3]
#       S      : non delayed Solutions list
#       Sd     : delayed Solutions list
#       ad=0   : coefficient for various beta_models
#       infected=True : determines if beta depends on
#                        infected population or death rate
#       randomR0=None : whether to add random noise to the
#                        model
#       spread=0 : coefficient for tanh beta_model
#       trate=0 : coefficient for tanh beta_model
#       beta_model : determines how beta is determined
#                    see  RK4_ddesolve comments
#       betastar=0.1 : coefficient for beta_model
###################################################################
def rhs(t,coeffs,S,Sd,ad=0,infected=True,randomR0=None,spread=0,trate=0,beta_model=None,betastar
    =0.1):
  # SIRDC equations
  # dS/dt=-beta*I*S/N
  # dI/dt=beta*I*S/N-gamma*I
  # dR/dt=gamma*I-theta*R
  # dD/dt=delta*theta*R
  # dC/dt=(1-delta)*theta*R
  # S is number susceptible,
  # I is number infected
  # R is number resolving/infectious
  # D is number of dead
  # C is number of recovered and immune
  # N is sum of these
  # I normalize all by N, which is constant
  # R0=beta/gamma is basic reproduction number
  # my beta**-1 is the typical time per contact
  # my gamma**-1 is the typical time infetious
  # my theta**-1 is the typical time before death or non-infectious
  # my delta is the death rate
  S,I,R,D,C=S
  Ss,Is,Rs,Ds,Cs=Sd
  beta0=coeffs[0]
  gamma=coeffs[1]
  theta=coeffs[2]
  delta=coeffs[3]
  # for infections use infected
  # for death rate use DE to get death rate
  if infected: der=Is
  else: der=delta*theta*Rs
  # balanced around R0
#  if randomR0==None or randomR0==0: randfac=0
#  else: randfac=max(2*(npr.random()-0.5)*(randomR0),-1)
  # unbalanced around R0
  if randomR0==None or randomR0==0: randfac=0
  else: randfac=npr.random()*randomR0
  if beta_model==None:
    beta=beta0
  elif beta_model=='exponential': # set new beta, exponential
    beta=(1+randfac)*np.exp(np.log(beta0)-ad*der)
  elif beta_model=='linear': # linear
    beta=max(beta0*(1+ad*der),betastar)
  elif beta_model=='cubic':# cubic
    beta=max(beta0*(1+ad*der**3),betastar)
  elif beta_model=='tanh': # tanh
    beta=beta0*(1+ad*(1+np.tanh(spread*(der-trate))))
  else: # default to None model
    beta=beta0
#  print(f'spread={spread:.2e},beta={beta},der={der:.2e}')
  c1=-beta*I*S
  c2=beta*I*S-gamma*I
  c3=gamma*I-theta*R
  c4=delta*theta*R
  c5=(1.-delta)*theta*R
#  c1=np.maximum(c1,0)
#  c2=np.maximum(c2,0)
#  c3=np.maximum(c3,0)
  return np.array([c1,c2,c3,c4,c5])

#Use RK4 to solve
################################################################################
# RK4_ddesolve function: Runge-Kutta 4th order DDE solver
#     input :
#       histfunc : (numpy array) the "initial condition" for a DDE
#       delay : (real) the delay in time units
#       stpdelay : (integer) the delay in number of positions in an array
#         # Thus dt=delay/stpdelay always for this
#       tmin    : (real) what time to start the calculation from
#       tmax    : (real) when to end the calculation
#       coeffs : (list) list of necessary coefficients for rhs
#       delayon=True : (boolean) if True a DDE, if False, changes to
#             an ODE  for delayon=False, dt=delay/stpdelay and the
#             histfunc should be a numpy array with the initial
#             conditions repeated twice (so if initial condition is y0,
#             then numpy.array([y0,y0]) should be the histfunc input)
#       irate=0 : (real) this sets where the sensitivity is for the
#                   beta_model (so behavior based off or switches
#                   when near this value)
#       infection_beta=True : (boolean) This determines if the beta_model is
#                               dependent on the infected population (true)
#                               or the death rate (false)
#       randomR0=None : whether to add random noise to the model. If it is
#                         None then no random noise. If it is a real number
#                         it adds a random value between 0 and the real given
#       Ffactor=0 : (real) coefficient for tanh beta_model, determines the
#                             width of the tanh function
#       beta_model=None : (str) determines which model of beta to use
#                possibilities include X(t) is either I(t) or dD/dt(t):
#                None        : beta is not time dependent
#                           beta=beta0=coeffs[0]
#                'linear'    : beta depends linearly on X(t), but can only go
#                               as low as betastar
#                           beta=max(beta0*(1+ad*der),betastar)
#                           ad=(betastar/beta-1)/(irate*sol[0,1])
#                'exponential': beta depends exponentially on X(t)
#                           beta=np.exp(np.log(beta0)-ad*X(t))
#                           ad=(-np.log(betastar/beta))/(irate*sol[0,1])
#                'cubic': beta depends cubically on X(t)
#                           beta=max(beta0*(1+ad*der**3),betastar)
#                           ad=(betastar/beta-1)/(irate*sol[0,1])**3
#                'tanh': beta depends on X(t) via the hyperbolic tangent
#                           beta=beta0*(1+ad*(1+np.tanh(spread*(der-trate))))
#                           ad=(betastar-beta)/2
#                            spread=np.arctanh(0.99)/(irate*sol[0,1])
#                              or
#                            spread=Ffactor if Ffactor is not None
#                           trate=irate*sol[0,1]
#       betastar=0.1 : coefficient for various beta_model
#     output :
#        list of numpy arrays, (time,solution[0],solution[1],etc.)
################################################################################
def RK4_ddesolve(histfunc,delay,stpdelay,tmin,tmax,coeffs,delayon=True,irate=0,infection_beta=True,randomR0=None,Ffactor=None,beta_model=None,betastar=0.1):
  nparams=len(coeffs)
  nvars=nparams
  print("Starting␣with␣history␣function␣of␣size",histfunc.shape)
  if (histfunc.shape[0]!=stpdelay+1):
    print("history␣function␣must␣be␣"+str(stpdelay+1)+"␣long␣array.")
    return -1
  if (histfunc.shape[1]!=nvars):
    print("history␣function␣must␣have␣{}␣components.".format(nvars))
    return -1
  beta=coeffs[0]
  gamma=coeffs[1]
  theta=coeffs[2]
  delta=coeffs[3]

  # set up
  dt=delay/stpdelay
  solsize=np.ceil((tmax-tmin)/dt)
  totalsize=int(stpdelay+1+solsize)
  print("total␣size␣of␣array",totalsize)
  sol=np.zeros([totalsize,nvars])
  # assume histfunc has initial time at tmin and tmin-dt
  sol[0:stpdelay+1,:]=histfunc
  t=np.linspace(tmin-delay,tmax,totalsize)
  bdelay=stpdelay+1
  # for infections and death rate calculations
  if beta_model=='exponential':
    ddrate=(-np.log(betastar/beta))/(irate*sol[0,1])
    ddrate2=0.
    ddrate3=0.
  elif beta_model=='linear':
    ddrate=(betastar/beta-1)/(irate*sol[0,1])
    ddrate2=0.
    ddrate3=0.
  elif beta_model=='cubic':
    ddrate=(betastar/beta-1)/(irate*sol[0,1])**3
    ddrate2=0.
    ddrate3=0.
  elif beta_model=='tanh':
    # out of order so we can reuse ddrate3
    ddrate=(betastar-beta)/2
    ddrate3=irate*sol[0,1]
    if Ffactor==None: ddrate2=np.arctanh(0.99)/ddrate3
    else: ddrate2=Ffactor
  else: # default to exponential model
    ddrate=(-np.log(betastar/beta))/(irate*sol[0,1])
    ddrate2=0.
    ddrate3=0.
  # we first go through the histfunc dependent part
  for i in range(bdelay,bdelay+stpdelay):
    if (delayon):
```

```python
      S1=[sol[i-1-stpdelay,j] for j in range(nvars)] # set up delay
    else:
      S1=[sol[i-1,j] for j in range(nvars)] #remove delay on S
    k1=dt*rhs(t[i-1],coeffs,sol[i-1,:],S1,ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    # only use linear interpolation on our constant history function
    if (delayon):
      Ss=[lin_interp([sol[i-stpdelay-1,j],sol[i-stpdelay,j]],[t[i-stpdelay-1],t[i-stpdelay]],t[i-1]+0.5*dt-delay) for j in range(nvars)]
      # set up delays for evaluation
      Ss1=Ss
      Ss2=Ss
    else:
      Ss1=[sol[i-1,j]+0.5*k1[j] for j in range(nvars)] # remove delay on S
    k2=dt*rhs(t[i-1]+0.5*dt,coeffs,sol[i-1,:]+0.5*k1,Ss1,ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    if (not delayon):
      Ss2=[sol[i-1,j]+0.5*k2[j]  for j in range(nvars)] # remove delay on S
    k3=dt*rhs(t[i-1]+0.5*dt,coeffs,sol[i-1,:]+0.5*k2,Ss2,ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    # set up delays
    if (delayon):
      Ss3=[sol[i-stpdelay,j] for j in range(nvars)]
    else:
      Ss3=[sol[i-1,j]+k3[j] for j in range(nvars)] # remove delay on S
    k4=dt*rhs(t[i-1]+dt,coeffs,sol[i-1,:]+k3,Ss3,ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    sol[i,:]=sol[i-1,:]+1/6.*(k1+2.*(k2+k3)+k4)
    if (np.min((sol[i,:]))<0):
      # Force all negative values to zero
      masker=sol[i,:]<0
      sol[i][masker]=0
  # now use values from histfunc or solution
  for i in range(bdelay+stpdelay,totalsize):
    # set up delays
    if (delayon):
      S1=[sol[i-1-stpdelay,j] for j in range(nvars)]
    else:
      S1=[sol[i-1,j] for j in range(nvars)] # remove delay on S
    k1=dt*rhs(t[i-1],coeffs,sol[i-1,:],S1,ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    # set up coefficients for Hermite interpolation
    if (delayon):
      # support is the number of data points to be supplied
      # for the Hermit cubic interpolation support=3
      support=3
      ISs=[np.ones(support) for j in range(nvars)]
      Ssp=[np.ones(support) for j in range(nvars)]
      tees=np.ones(support)
      offset=1
      for j in range(support):
        Sspt=rhs(t[i+j-offset-stpdelay],coeffs,sol[i+j-offset-stpdelay,:],sol[i+j-offset-2*stpdelay,:],ad=ddrate,infected=infection_beta,randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar) #for delays on S
        for k in range(nvars):
          Ssp[k][j]=Sspt[k]
          ISs[k][j]=sol[i+j-offset-stpdelay,k]
        tees[j]=t[i+j-stpdelay-offset]
      tdes=np.array([t[i-1]+0.5*dt-delay])
      Ss=[herm_interp(tdes,tees,ISs[j],Ssp[j])[0][0] for j in range(nvars)]
#    # could also use linear interpolation
#     if (delayon):
#        Ds=lin_interp([sol[i-stpdelay-1,0],sol[i-stpdelay,0]],[t[i-stpdelay-1],t[i-stpdelay]],t[i-1]+0.5*dt-delay)
#        Ss=lin_interp([sol[i-stpdelay-1,1],sol[i-stpdelay,1]],[t[i-stpdelay-1],t[i-stpdelay]],t[i-1]+0.5*dt-delay)
#   set up delays
    if (delayon):
      Ss1=Ss
      Ss2=Ss
    else:
      Ss1=[sol[i-1,j]+0.5*k1[j] for j in range(nvars)]  # remove delay on S
    k2=dt*rhs(t[i-1]+0.5*dt,coeffs,sol[i-1,:]+0.5*k1,Ss1,ad=ddrate,infected=infection_beta,
    randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    if (not delayon):
      Ss2=[sol[i-1,j]+0.5*k2[j] for j in range(nvars)] # remove delay on S
    k3=dt*rhs(t[i-1]+0.5*dt,coeffs,sol[i-1,:]+0.5*k2,Ss2,ad=ddrate,infected=infection_beta,
    randomR0=randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    if (delayon):
      Ss3=[sol[i-stpdelay,j] for j in range(nvars)]
    else:
      Ss3=[sol[i-1,j]+k3[j] for j in range(nvars)] # remove delay on S
    k4=dt*rhs(t[i-1]+dt,coeffs,sol[i-1,:]+k3,Ss3,ad=ddrate,infected=infection_beta,randomR0=
    randomR0,spread=ddrate2,trate=ddrate3,beta_model=beta_model,betastar=betastar)
    sol[i,:]=sol[i-1,:]+1/6.*(k1+2.*(k2+k3)+k4)
    # change to zero if we get negative numbers
    if (np.min((sol[i,:]))<0):
      masker=sol[i,:]<0
      sol[i][masker]=0
  return [t]+[sol[:,j] for j in range(nvars)]
```