Modeling Using Delay Differential Equations

Alan Bruner, Evan Cochrane, Theodore Guetig, Ben Wilfong

1 Introduction

Real-world applications of differential equations often include delay differential equations (DDE), since they rely on a model's history instead of initial conditions. [2] The history improves the solution's alignment with the provided historical information. The history dependency by nature invokes a time delay since these equations are cumulative, or must process past information before determining a present value. This characteristic in DDEs makes it appropriate to represent many real-world models. To develop an understanding of DDE models we will look at epidemiology models, future data predictions, partial element circuits, and computational cloud environments.

2 Epidemiology Modeling

One common application of delay differential equations is to improve upon a basic Susceptible Infected Recovered (SIR) epidemiology model. The basic form of an SIR model represents the number of infected individuals using $S(t)$, the number of infected individuals using $I(t)$, and the number of recovered individuals using $R(t)$. A typical basic SIR model is

$$\begin{aligned} \frac{\mathrm{d}S}{\mathrm{d}t}(t) &= -\alpha S(t)I(t) \\ \frac{\mathrm{d}I}{\mathrm{d}t}(t) &= \alpha S(t)I(t) - \beta I(t) \\ \frac{\mathrm{d}R}{\mathrm{d}t}(t) &= \beta I(t) \end{aligned} \tag{1}$$

where α represents the percent of interactions between infected individuals and susceptible individuals that result in an infection, and β represents the percentage of infected individuals who recover per unit of time.

This model is mostly accurate, but it does not account for changing population sizes. In a practical example, a population will have some number of individuals entering and leaving. To account for this changing population size,

the model can be updated to be

$$\begin{aligned}
\frac{dS}{\mathrm{d}t}(t) &= \rho - \alpha S(t)\frac{I(t)}{N(t)} - \delta S(t) \\
\frac{dI}{\mathrm{d}t}(t) &= \alpha S(t)\frac{I(t)}{N(t)} - (\beta + \delta + \epsilon)I(t) \\
\frac{dR}{\mathrm{d}t}(t) &= \beta I(t) - \delta R(t) \\
N(t) &= S(t) + I(t) + R(t)
\end{aligned} \tag{2}$$

This new model adds a new expression, $N(t)$, which represents the total number of individuals in the system. This is then used to normalize the $S(t)I(t)$ interactions by the total population size. The new parameters to this model are

ρ: the recruitment rate to the population $\dfrac{\text{person}}{\text{time}}$

δ: the natural mortality rate $\dfrac{1}{\text{time}}$

ϵ: the mortality rate due to disease $\dfrac{1}{\text{time}}$

The parameters that remain from the previous model have largely unchanged meanings, with only slight unit changes to account for the normalization.

This model can be improved to account for vaccinations through the use of delay differential equations. When individuals are vaccinated, there is a delay between getting the vaccination and the immunity that comes from the vaccine. A standard differential equation is unable to account for this delay, so it becomes necessary to use a delay differential equation. This updated model becomes

$$\begin{aligned}
\frac{dS}{\mathrm{d}t}(t) &= \rho - \alpha S(t)\frac{I(t)}{N(t)} - \delta S(t) - uS(t-\tau) \\
\frac{dI}{\mathrm{d}t}(t) &= \alpha S(t)\frac{I(t)}{N(t)} - (\beta + \delta + \epsilon)I(t) \\
\frac{dR}{\mathrm{d}t}(t) &= \beta I(t) - \delta R(t) + uS(t-\tau) \\
N(t) &= S(t) + I(t) + R(t)
\end{aligned} \tag{3}$$

The new parameters to this model are u and τ. The parameter u is the percent of susceptible individuals the receive a vaccination at time $t-\tau$. The parameter τ represents the delay between an individual receiving a vaccination and them gaining immunity from the vaccination.

Another set of terms can be added to this model to show a more accurate measure of infected individuals. An assumption that can be made about the number of infected individuals is that

$$I(t) = \int_{t-\tau}^{t} \hat{I}(y)dy$$

where $\hat{I}(y)$ represents the rate of infection at time y. Adding this to the vaccination model changes the model to be

$$\begin{aligned}
\frac{dS}{\mathrm{d}t}(t) &= \rho - \alpha S(t)\frac{I(t)}{N(t)} - \delta S(t) - uS(t-\tau) \\
\frac{dI}{\mathrm{d}t}(t) &= \hat{I}(t) - \hat{I}(t-\tau) \\
\frac{dR}{\mathrm{d}t}(t) &= \beta I(t) - \delta R(t) + uS(t-\tau) \\
N(t) &= S(t) + I(t) + R(t) \\
\hat{I}(t) &= \alpha S(t)\frac{I(t)}{N(t)} - (\beta + \delta + \epsilon)I(t)
\end{aligned} \tag{4}$$

where all parameters remain unchanged from the previous model.

To look at these models, the real-world example of the H1N1 virus in Morocco was used. This example gives the parameters

$$\begin{aligned}
\tau &= 10 \\
\alpha &= 0.3095 \\
\beta &= 0.2 \\
\rho &= 1174.17 \\
\epsilon &= 0.0063 \\
d &= 3.9130 \times 10^{-5} \\
u &= -.0.02
\end{aligned} \tag{5}$$

Figure 1 shows the resulting graph from applying Model 3 using these parameters. Figure 2 shows the resulting graph from applying Model 4 using the same parameters. The second model is relatively stable during the first 100 time-steps, but near the time-step at $t = 120$, the solution begins to rapidly oscillate with increasing amplitude. This is likely due to the rate of infection becoming negative, which causes the total number of infected individuals to become inaccurate, and this inaccuracy then propagates into the other terms in the model.

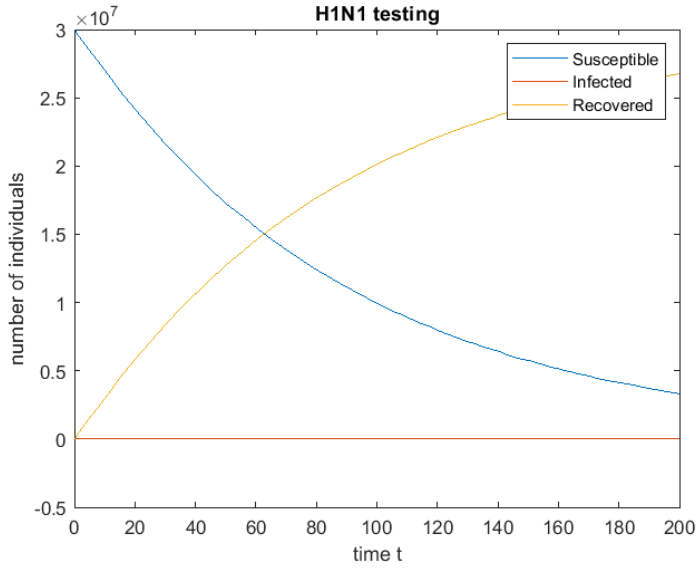


Figure 1: Number of Susceptible, infected, and recovered individuals using Model 3

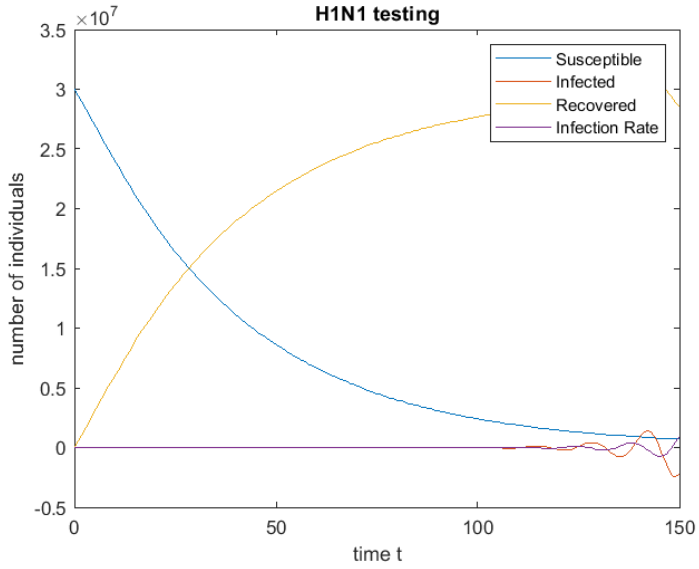


Figure 2: Number of Susceptible, infected, and recovered individuals, and the rate of infection using Model 4

3 El-Niño La-Niña Modeling

Delay differential equation are also used to predict the temperature fluctuations associated with the El-Niño La-Niña cycle. We do this using both the single delay model

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\alpha T(t-\tau) + T - T^3 \tag{6}$$

and the two delay model

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\alpha \tanh(\kappa T(t-\tau_1)) + \beta \tanh(\kappa T(t-\tau_2)) + \gamma \cos(2\pi t) \tag{7}$$

with all constants α, β, γ, κ, τ_1, and τ_2 greater than zero. We begin our exploration by solving Model (7) with the parameters in Table 1 and constant histories $T(t) = 1$ and $T(t) = 0$ for $t \leq 0$. The results of this exploration are shown in Figure 3. The interesting conclusion that can be drawn from this exploration is that for these simple histories, the steady-state solution for each instance remains unchanged once a short transient period is completed. It is also helpful to note that changes in the parameters yield vastly different solutions.

Table 1: Model Exploration Parameters

Instance	α	β	γ	κ	τ_1	τ_2
1	1	0	1	100	0.01	0
2	1	0	1	100	0.15	0
3	1	0	1	100	0.995	0
4	1	1	1	10	0.9	0.1
5	1.2	0.8	1	10	0.6	0.6

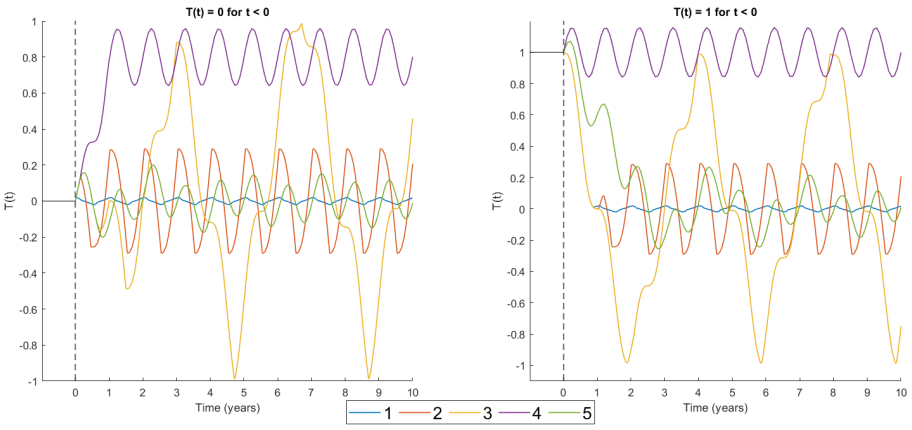


Figure 3: Parameter Exploration Results

These result of applying these parameters to the historical data shown in Figure 4 at several different start dates is shown in Figure 5

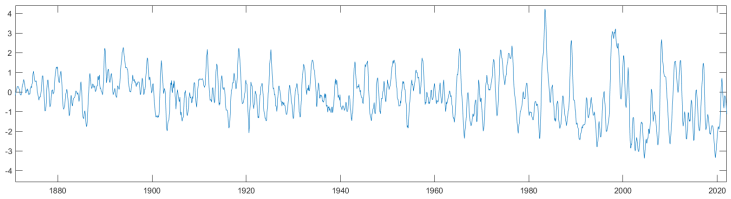


Figure 4: NOAA Data [3]

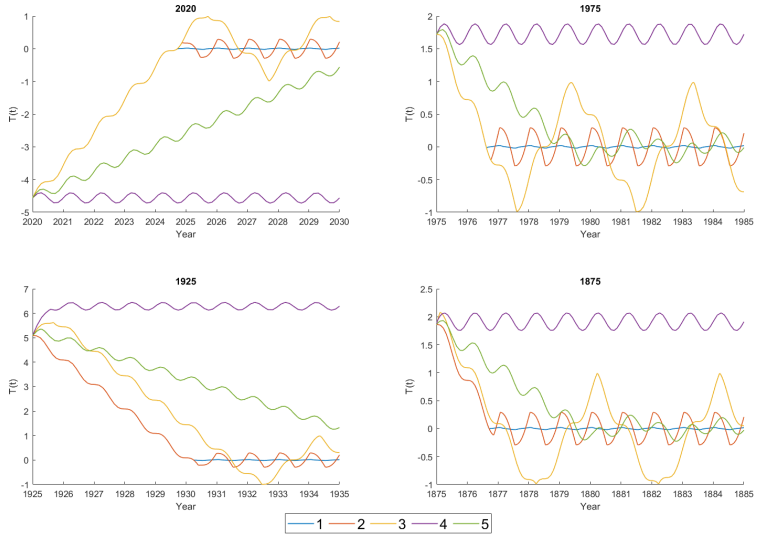


Figure 5: Parameter Exploration With Real History

In order to best predict future oscillations, the model parameters was optimized to best fit the historical data. This optimization was performed over all parameters $\alpha,\ \beta,\ \gamma,\ \kappa,\ \tau_1$, and τ_2. The residual function was determined by

$$R = \sum_{i=1}^{n} \left|(T - T_{\text{real}})\right|_{\infty} \left(t_{\text{predict}} - t_{\text{max error}} + 1\right)^2$$

where T is the predicted solution, T_{real} is the exact solution, t_{predict} is how long of a prediction is made, and $t_{\text{max error}}$ is the future time at which the error is maximum. This approach punishes errors near the initial condition with the hope of getting the best short term predictions. T_{Real} was determined by using

an initial condition at year t_o prior to $2022-t_{\text{predict}}$. The historical data between t_0 and t_0+t_{predict} is then used as the exact solution. This shift was performed for all possible integer shifts in hopes of finding the best possible fit. Some results of this optimization for Model 7 using MATLABs constrained optimization solver are provided in Table 2 and Figures 6 through 9.

Table 2: Model 7 Optimized Parameters

	Initial Conditions						Optimized Parameters						
Instance	τ_1	τ_2	α	β	γ	κ	τ_1	τ_2	α	β	γ	κ	R
1	1	1	1	1	1	1	2.21	2.05	1.05	0.95	0.99	1.00	2.73e3
2	10	15	1	1	0.1	5	14.23	9.78	0.95	0.97	0.097	49.91	2.70e3
3	15	10	0.1	0.1	0.1	5	14.32	9.69	0.54	0.12	0.0004	9.23	2.65e3
4	15	15	2.5	2.5	2.5	2.5	14.97	15.21	2.54	2.46	2.48	2.48	2.75e3

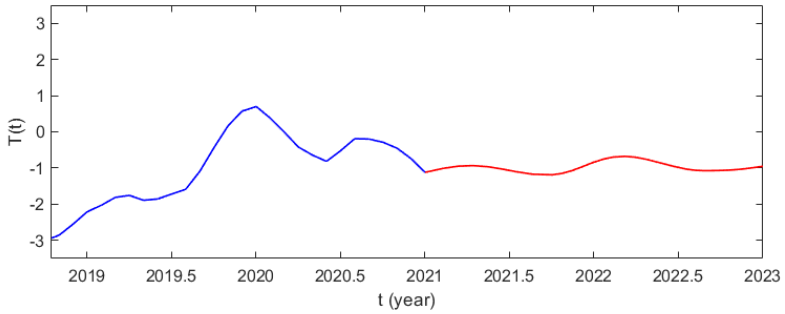


Figure 6: Model Two Instance 1 - History is in blue, prediction in red

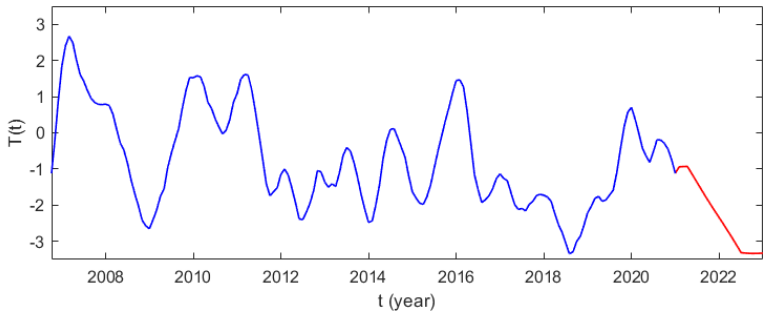


Figure 7: Model Two Instance 2 - History is in blue, prediction in red

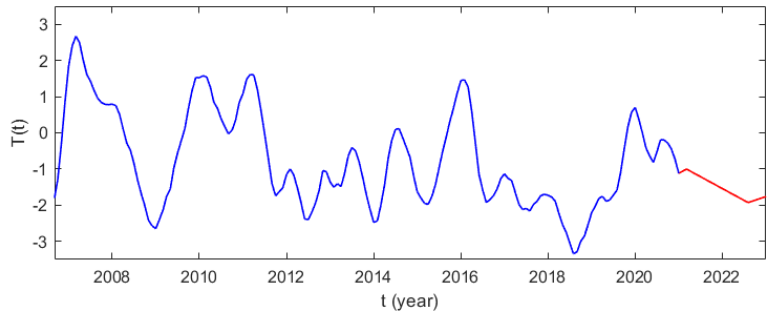


Figure 8: Model Two Instance 3 - History is in blue, prediction in red

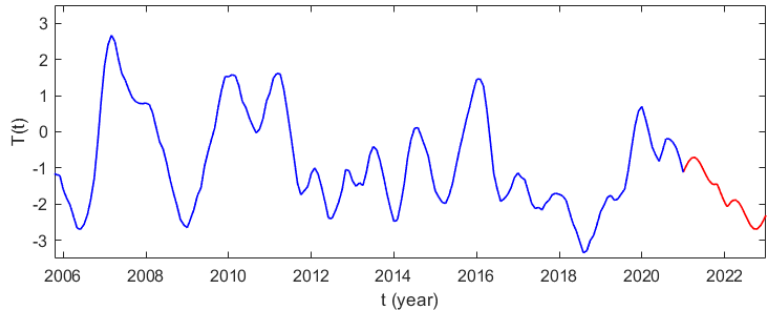


Figure 9: Model Two Instance 4 - History is in blue, prediction in red

Unfortunately, none of these predictions are particularly good, nor were the others when over 200 initial conditions were searched. Attempts were made using MATLABs simulated annealing and surrogate optimization tools, but results were similar for all three tools. The same optimization approach was used to fit model A to the history data. Results for this optimization using various initial conditions are provided in Table 3 and figures 10 through 13.

Table 3: Model A Optimization Parameters

	Initial		Optimized		
Instance	τ	α	τ	α	R
1	2	2.5	2.99	1.00	3.25e3
2	1	3	3.365	1.00	3.16e3
3	0.2	0.2	0.24	1.46	2.45e3
4	1	0.5	0.0043	1.79	2.49e3

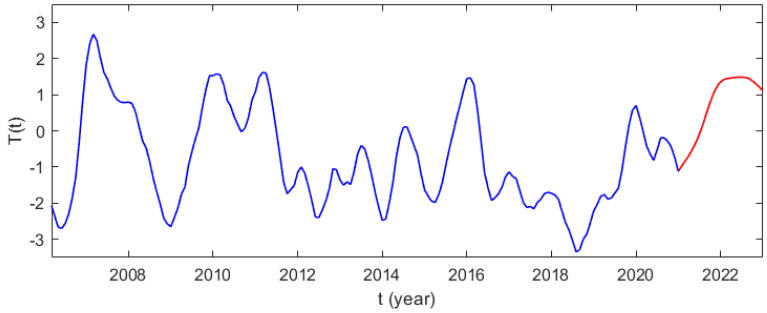


Figure 10: Model One Instance 1 - History is in blue, prediction in red

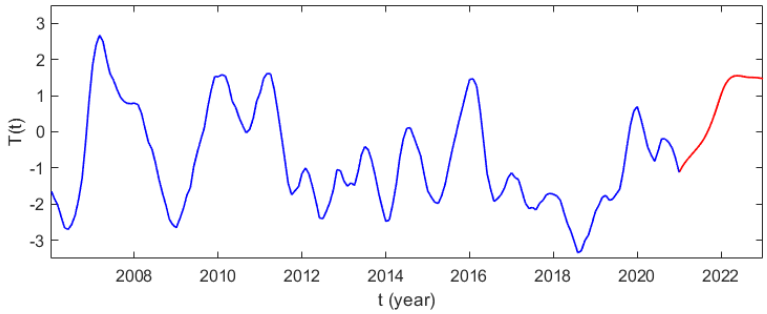


Figure 11: Model One Instance 2 - History is in blue, prediction in red

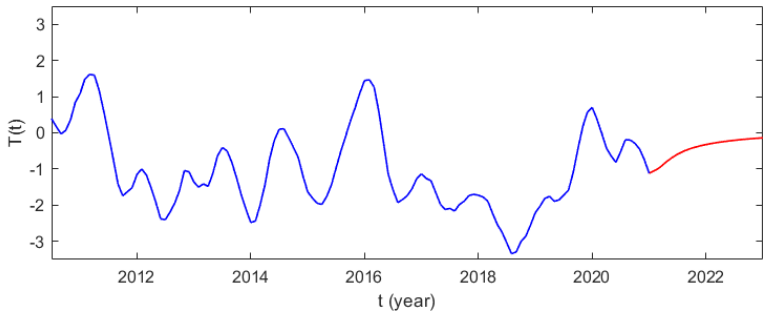


Figure 12: Model One Instance 3 - History is in blue, prediction in red

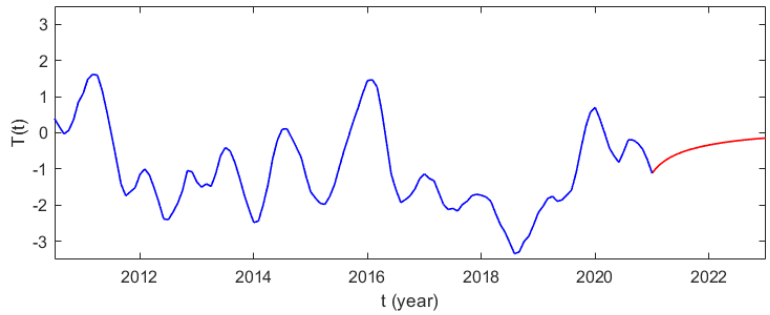


Figure 13: Model One Instance 4 - History is in blue, prediction in red

This optimization process also failed to produce reasonable predictions. Again, sampling at more initial conditions did not improve the output. Based on the residual values found, model A yields a better fit, but this conclusion is suspect given the qualitatively poor predictions of both models.

4 Partial Element Circuits

In circuit analysis, DDE plays a role in properly modeling partial element circuits. The inherit delay in DDE is to represent the electrical delay experienced in real-world circuits. The following linear DDE model will be used to examine the impact of delays.

$$y'(t) = Ly(t) + My(t-\tau) + Ny'(t-\tau), t \geq 0 \tag{8}$$

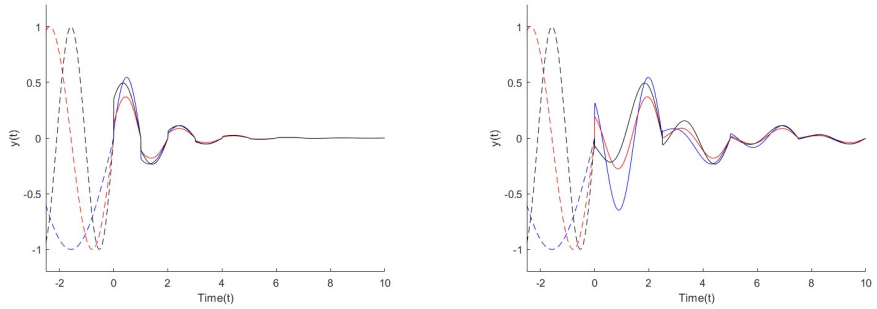
where,

$$L = 100\begin{bmatrix} -7 & 1 & 2 \\ 3 & -9 & 0 \\ 1 & 2 & -6 \end{bmatrix}, \quad M = 100\begin{bmatrix} 1 & 0 & -3 \\ -1/2 & -1/2 & -1 \\ -1/2 & -3/2 & 0 \end{bmatrix},$$
$$\text{and } N = \frac{1}{72}\begin{bmatrix} -1 & 5 & 2 \\ 4 & 0 & 3 \\ -2 & 4 & 1 \end{bmatrix}.$$

The DDE model has two delays with one involving y and another on the derivative of y. A delay on a derivative classifies the DDE as neutral and required a different solver to properly apply both delays. MATLAB's "ddensd" was used to properly handle both variations of the delay. The history of the model is denoted by

$$h(t) = (sin(t), sin(2t), sin(3t)) \tag{9}$$

The delay, τ, was solved at various values to observe its impact on the model. Figure 14 shows the results for selected values of delay, τ.



(a) $\tau = 1$ (b) $\tau = 2.5$

(c) $\tau = 5$ (d) $\tau = 10$

Figure 14: Potential model results for various values of τ. The blue lines represent y(t) with history $sin(t)$, the red lines represent y(t) with history $sin(2t)$, and the black lines represent y(t) with history $sin(3t)$. The dashed lines are the histories of their respected colors when $t < 0$.

The delayed derivative results in a solution that decays over time. The delay adjust how quickly the solution reaches a value of zero. At $\tau = 1$, the results show that all histories settle to zero around $t = 5$. When $\tau = 2.5$, the behavior of each individual history can be seen better. The frequency of $sin(t) and sin(2t)$ history align together while $sin(3t)$ lags behind. At $\tau = 5$, the individual histories become clearly represented. After t = 0, the results of $sin(3t)$ history exhibits a different path than the other histories. A similar characteristic occurs after t = 5 but impacts the results of $sin(t)$ history. When $\tau = 10$, the same pattern continues but for a longer duration before decaying.

5 Distributed Cloud Computing Modeling

We also use delay differential equations to model a network of cloud computers that can send packets to one another for processing. The delay will be modeled as the time it takes to send data from one computer to another.

Let c index the set of computers C. We model the network as a weighted

directed graph $N = (C, E)$. The weights represent a computer's relative propensity to send packets to each of the other computers if the edge is not a loop, or to process packets itself if the edge is a loop. This information is stored in a nonnegative weight matrix W for a network N. Weights of 0 for non-loops correspond to the directed edge not being present, so no data transfer can occur. Weights of 0 for loops correspond to a computer at which no data can be processed. Each computer also has a limit on the amount of data that can be processed, stored in a vector $\mathbf{m}_c$.

Let $P_c(t)$ model the number of packets at computer c at time t. The rate at which computer c processes data $\frac{dP_c^*}{dt}$ is proportional to the amount of data it has, but limited by the $\mathbf{m}_c$ value,

$$\frac{dP_c^*}{dt} = -\alpha \min\{P_c(t), \mathbf{m}_c\}.$$

The rate at which computer c transfers data to computer $d \neq c$, $\frac{dP_c^d}{dt}$, is proportional to the difference in the packets in c and the packets in d,

$$\frac{dP_c^d}{dt} = -\alpha(P_c(t) - P_d(t - \tau_{cd})).$$

The proportionality constants α are determined by the weight matrix W.

We also include a final node f that solely keeps track of all the packets processed by all computers, and let $\mathbf{I}(t)$ denote the packets being input into the system at time. The complete system of differential equations is then

$$\frac{dP_c}{dt} = \mathbf{I}_c(t) - W_{cc} \min\{P_c(t), \mathbf{m}_c\} - \sum_{\substack{d \in C \\ d \neq c}} W_{cd}(P_c(t) - P_d(t - \tau_{cd})) \quad \forall c \in C$$

$$\frac{dP_f}{dt} = \sum_{c \in C} W_{cc} \min\{P_c(t - \tau_{cf}), \mathbf{m}_c\}.$$

In our models, we only consider input packet rates $\mathbf{I}(t)$ that are non-zero for one computer, which then must distribute packets to other computers.

A baseline instantiation of this model uses weight matrix

$$W = \begin{bmatrix} 1 & 0.2 & 0.2 \\ 0.2 & 1 & 0.2 \\ 0.2 & 0.2 & 1 \end{bmatrix},$$

processing vector

$$\mathbf{m} = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix},$$

input rate

$$\mathbf{I}_1(t) = \begin{cases} 10 & 0 \leq t \leq 5 \\ 0 & t > 5 \end{cases},$$

and all delays equal to 2 seconds. This topology is summarized in the network graph:

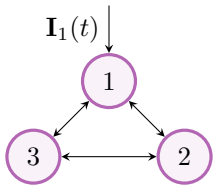


Figure 15: Baseline network N_1

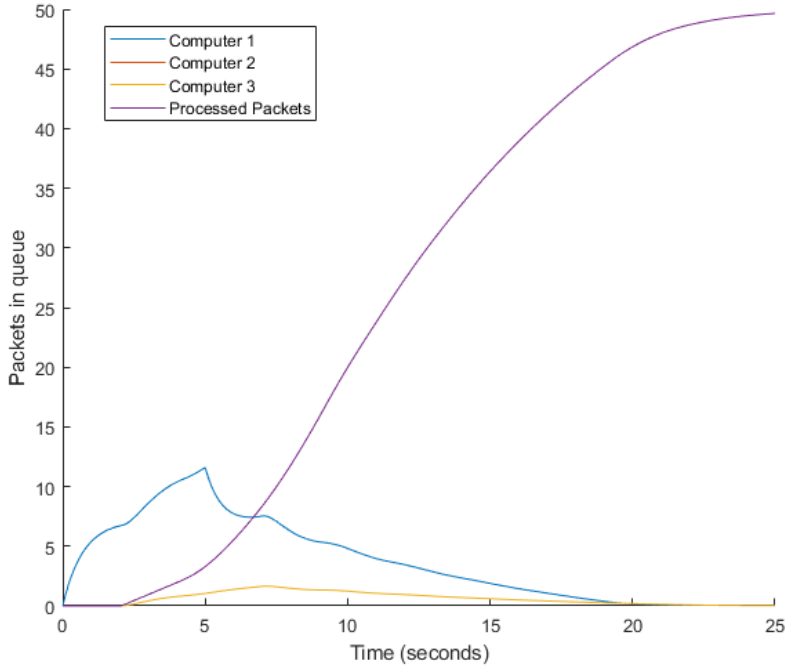


Figure 16: Packets present in each computer v. time for network N_1

The solution to this instantiation using Matlab's `ddesd` solver for 50 seconds is shown in Figure 16.

Computers 2 and 3 are completely symmetric, so the curves for computer 2 and computer 3 are identical. The total number of processed packets also remains at zero for the first 2 seconds due to the delay. The total number of processed packets approaches the total number of packets input into the system.

This number is the integral over the sum of the values of the input function $I(t)$,

$$\int_0^\infty \sum_{c \in C} \mathbf{I}_c(t)\, dt,$$

which in the case of N_1 is 50.

Removing the directed edge from computer 1 to 3 from network N_1 creates the network N_1^A and results in the following weight matrix and topology shown in Figure 17:

$$W = \begin{bmatrix} 1 & 0.2 & 0 \\ 0.2 & 1 & 0.2 \\ 0.2 & 0.2 & 1 \end{bmatrix}.$$

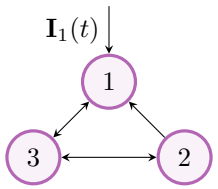


Figure 17: Baseline network N_1^A

The solution to Network N_1^A is shown in Figure 18.

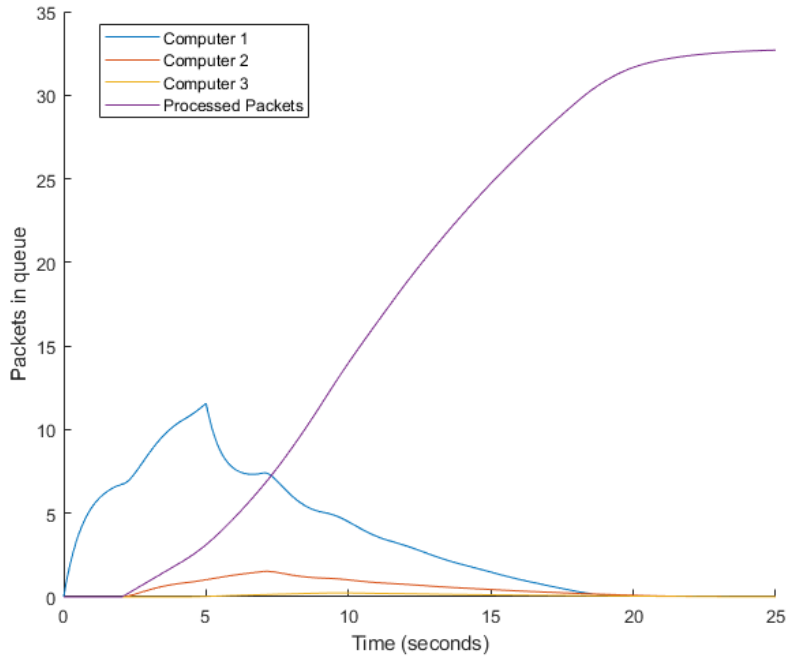


Figure 18: Packets present in each computer v. time for network N_1^A

Computers 2 and 3 no longer have the same number of packets at every time, but unfortunately, the total number of packets processed is not the total number of packets input, so packets are lost. Packets are only lost if the weight matrix is not symmetric, $W_{cd} \neq W_{dc}$ for some $c, d \in C$, so we use a symmetric W in the remaining network simulations. This effectively limits our network topology to undirected graphs with loops.

We next instantiate the model with 4 new networks on 4 nodes with all non-loop weights being either 0 or 0.5, all loop-weights being 0, processing vector

$$\mathbf{m} = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 4 \end{bmatrix},$$

input rate

$$\mathbf{I}_1(t) = \begin{cases} 1000 & 0 \leq t \leq 0.1 \\ 0 & t > 0.1 \end{cases},$$

and all delays equal to 2 seconds. The input rate is meant to resemble a 'spike' in data being input into the network.

The topologies of the four networks are the complete graph K_4 on 4 vertices, the cycle graph C_4 on 4 vertices, the path graph P_4 on 4 vertices, and the star

graph S_4 on 4 vertices. (see Figure 19). Solutions are shown in Figures 20 through 23.

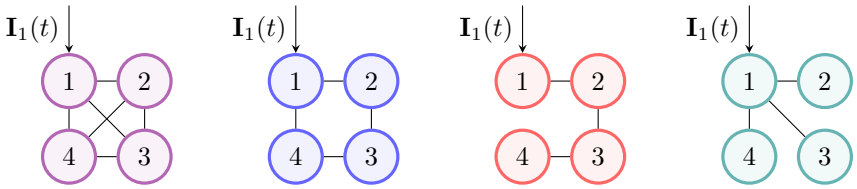


Figure 19: From Left to Right: Networks $N_2^K, N_2^C, N_2^p, N_2^S$

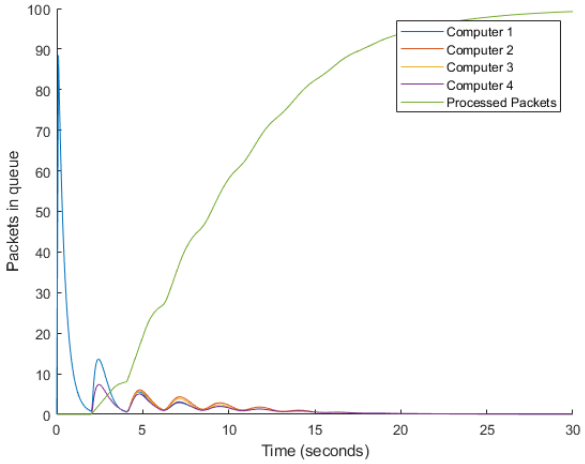


Figure 20: Packets present in each computer v. time for network N_2^K

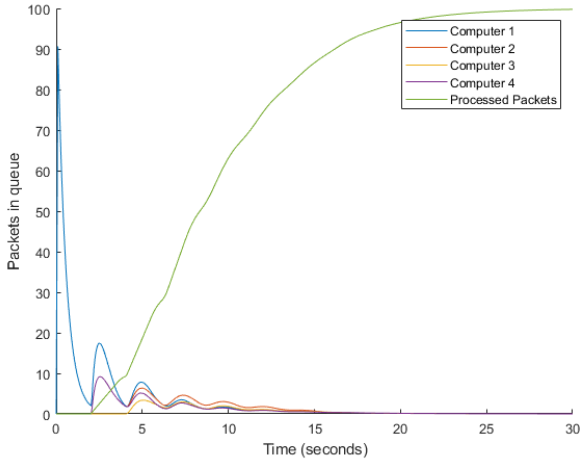


Figure 21: Packets present in each computer v. time for network N_2^C

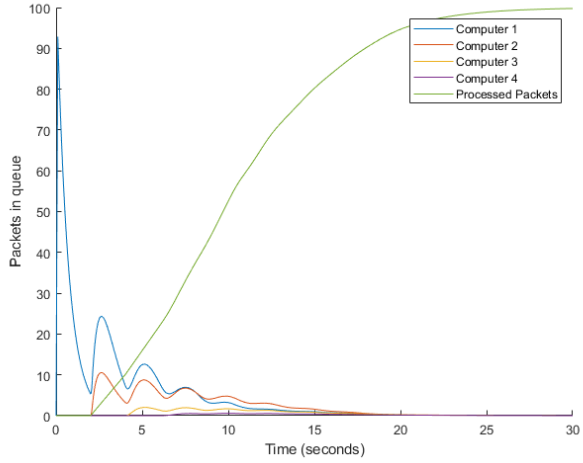


Figure 22: Packets present in each computer v. time for network N_2^P

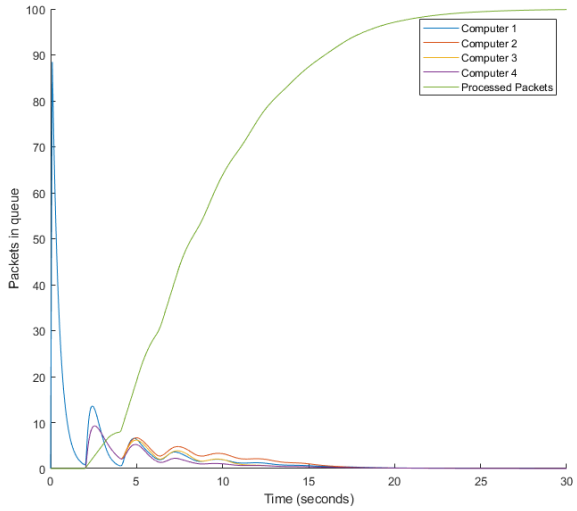


Figure 23: Packets present in each computer v. time for network N_2^S

The total number of packets processed approaches 100 as time increases for all networks. The topologies alone have an impact on the way packets are distributed and processed. The delays in transferring packets ensures that some computers need to wait longer than others before any packets at all can be received. This is most visible in N_2^P, since packets take 2 seconds to reach computer 2, 4 to reach computer 3, and 6 to reach computer 4. The more connected the graph is, the closer the four computers appear in the number of packets in waiting. It also appears like the more connected graphs N_2^K and N_2^C take longer to process their packet queues to near zero than the less connected graphs N_2^P and N_2^S.

To quantify this, we look at when the total number of packets left to be processed is less than half a packet. In this case we look for the first time at which the number of processed packets is greater than 99.5. These results are tabulated below:

Network	$\min t$ s.t.$P_f(t) > \int_0^\infty \sum_{c \in C} \mathbf{I}_c(t)\, dt - 0.5$
N_2^K	31.9331 seconds
N_2^C	27.3289 seconds
N_2^P	27.6019 seconds
N_2^S	25.7609 seconds

Interestingly, the less connected graphs processed the data faster than the connected ones, despite emptying their packet queues faster. The star topology S_4 gave the best results, being over 6 seconds or almost 20 percent faster than the

complete topology K_4. The probable reason for this is that the more connected topologies spend more time transferring packets between one another and less time actually processing the data. This behavior can be seen in Figure 20, as packets are sent out from all computers simultaneously and then reappear.

These results do not seem unreasonable. Many real-world parallel and distributed computing environments use star graphs, or more general trees with a height greater than 1. [1] The input node distributes tasks in an orderly fashion to each of its children, and each child sends its results to its parent. This is referred to as a tree network.

As an extension, we looked at all possible symmetric graphs on 5 vertices and found the approximate processing times using the same method. The delays were once again 2 seconds, the processing rates of all computers were 2, and the $I(t)$ function and possible values in the W matrix were the same as in the previous problem. There are 1024 symmetric graphs on 5 vertices, and, even taking isomorphism and disconnected graphs into account, there are too many to list every single topology and its resulting time.

The longest processing time is 52.7110 seconds, and this occurred for any topology for which computer 1, the computer at which all data is input, is not connected to any other computer. The complete graph K_5 has a middling performance of 36.9434 seconds. The star graph S_5 with computer 1 as its center is once again the best performer out of any computed topology, with a processing time of 25.1350 seconds, although by a narrower margin than last time. The next best graph is any graph isomorphic to S_5 with one edge deleted, with a processing time of 25.8199 seconds.

References

[1] Barry Wilkinson & Michael Allen. *Parallel Programming*. Upper Saddle River, NJ: Pearson, 2005.

[2] Allen Holder and Joseph Eichholz. *An Introduction To Computational Science*. International Series in Operations Research & Management Science. Cham, Switzerland: Springer, 2019.

[3] Kevin Trenberth. *TNI Index*. https://psl.noaa.gov/gcos$_w$*gsp*/*Timeseries*/*Data*/*tni.long.data*.

A Summary

We explored several models that can be represented as delay differential equations such as epidemiology, future data predictions, partial element circuits, and computational cloud environments. Each model used delayed differential equations in their solution, but varied on the application towards the problem.

For the partial element circuit, the DDE included a derivative on the delay which is known as a neutral DDE. Unlike the other DDE analyzed, the MATLAB solver "ddensd" had to be used to account for the delay derivative. The different values for delay, τ, impacted the solution's rate of decay regardless of history. At larger values of τ we could clearly see the effects of the different histories based on frequency and behavior.

We also explored and attempted to optimize two models of the El-Niño La-Niña oscillation. The models are

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\alpha T(t-\tau) + T - T^3$$

and

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\alpha \tanh(\kappa T(t-\tau_1)) + \beta \tanh(\kappa T(t-\tau_2)) + \gamma \cos(2\pi t).$$

The first of these is a simple one delay model while the second is a more complicated two delay model. Unfortunately, our attempts to optimize the model for accurate predictions did not lead to any insightful results as no quality fit was obtained. MATLAB's implementations of the interior-point method, simulated annealing, and surrogate optimization failed to produce quality predictions. This is likely due to the formulation of multi-dimensional global problem we were tried to solve.

Finally, we used delay differential equations to model a distributed cloud computing environment, where computers can send and receive packets of data to each other, or choose to process the data themselves. The rate of flow from one computer to another is dependent on the difference between the current number of packets stored in the two computers, however, this information must also be sent between computers, and has a delay between sending and receiving, so DDEs were a natural way of modeling this. In particular, computer networks were modeled as graphs, and we searched various graph topologies across a range of vertex numbers, processing speeds, and input data rates. In every case we looked at, the complete graph (the topology of all computers being able to send or receive data from all other computers) was not the most efficient, since data was sent back in forth between computers in a loop. With 5 computers, the complete graph took 36.9 seconds to completely process 100 packets of data. The star graph (the topology of all computers save one being connected to only a central computer that received all data) proved the best candidate in our model. With 5 computers, the star graph took 25.8 seconds to process all of the data.