# MODELING HISTORICAL DEPENDENCY WITH DELAY DIFFERENTIAL EQUATIONS

Chuong Nguyen, Nicole Richardson, Ted Samore

Rose-Hulman Institute of Technology
{nguyenct, richarnj, samoretc}@rose-hulman.edu

**ABSTRACT**

Delay differential equations (DDEs) are used to model the ocean temperature in the El-Nino – La-Nina oscillation and partial element circuits. DDEs were also used in modeling a cloud computing system. In addition, two cloud computing models were developed.

***Keywords— Delayed Differential Equations, DDE, El-Nino – La-Nina, Cloud Computing, Processor***

## 1. INTRODUCTION

Delayed differential equations (DDEs) are ordinary differential equations (ODE) that have dependencies on historical values of the function and its derivative(s). An ODE is a DDE with zero time dependence, or lag. These equations can be used to model events such as ocean temperature in the El-Nino – La-Nina oscillation or partial element circuits as done in the first two parts of this paper. DDEs can also be used in modeling real-time systems with time delays, such as a cloud computing system.

## 2. METHODOLOGY

The procedures for this experiment began with gaining familiarity with the MATLAB package for DDEs as outlined in sections 2.1 and 2.2. This knowledge base was then applied to the design and testing of a model for a cloud computing system.

### 2.1 Two-delay of El-Nino – La-Nina

Experimentation was introduced with a simple retarded DDE (no dependence on historical values of T'(t)). The equation,

$$\frac{dT}{dt} = -\alpha * \tanh\big(\kappa * T(t-\tau_1)\big) + \beta * \tanh\big(\kappa * T(t-\tau_2)\big) + \gamma * cos(2\pi t),$$

shows two delay dependencies on the function, $\tau_1$ and $\tau_2$. The parameter values are given in Table 1.

**Table 1: Parameter values for each instance of the El-Nino – La-Nina oscillation. Each instance was repeated for the histories of h(t) = 0 and h(t) = 1.**

| Instance | α | β | κ | $\tau_1$ | $\tau_2$ |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 100 | 0.01 | 0 |
| 2 | 1 | 0 | 100 | 0.15 | 0 |
| 3 | 1 | 0 | 100 | 0.995 | 0 |
| 4 | 1 | 1 | 10 | 0.4 | 0.9 |
| 5 | 1.2 | 0.8 | 10 | 0.6 | 0.6 |

The five instances of the DDE model were repeated for both h(t) = 0 and h(t) = 1.

### 2.2 Partial Element Circuits

The experimentation was expanded to a neutral DDE (dependent on historical values of T'(t)). This included matrices as the parameters to the equation,

$$y'(t) = L * y(t) + M * y(t-\tau) + N * y'(t-\tau), t \geq 0,$$

which were,

$$L = 100\begin{bmatrix} -7 & 1 & 2 \\ 3 & -9 & 0 \\ 1 & 2 & -6 \end{bmatrix}, M = 100\begin{bmatrix} 1 & 0 & -3 \\ -1/2 & -1/2 & -1 \\ -1/2 & -3/2 & 0 \end{bmatrix},$$

$$\text{and } N = \frac{1}{72}\begin{bmatrix} -1 & 5 & 2 \\ 4 & 0 & 3 \\ -2 & 4 & 1 \end{bmatrix}.$$

For this model, the history function was,

$$h(t) = [\sin(t) \quad \sin(2t) \quad \sin(3t)].$$

### 2.3 Cloud Computing

A simplified cloud computing model was constructed first as a proof of concept prior to increasing the complexity and varying the parameters to experiment with the effect of lag time on performance. The simplified model consisted of two processors each of which could send information to the other. The model was a system of two delay differential equations that could be modified to explore different parameters, for example packets could be input to one or both processors and that input could be continuous or an impulse. The history functions, lag time, and computational speed of the processor can also be changed. The first model is dictated to by the differential equations,

$$\frac{dS_i}{dt} = -\alpha\big(S_i(t)\big) * S_i(t) + \Big(1 - \alpha\left(S_j(t-\tau)\right)\Big) * S_j(t-\tau) + f(t)$$

$$\alpha\big(S_i(t)\big) = 1 - \frac{S_i(t)}{S_i(t)+k}.$$

Each processor (i) has a differential equation of the above form that is measuring the rate of change of the size of the queue. The alpha function varies the percentage of time spent on sending out processes versus completing them. The value of alpha depends on the on the queue size and a parameter k. When alpha is one, the processor is not sending anything to the other processor. When alpha is zero, the processor is only sending tasks. The parameter k dictates how fast alpha will change with a change in queue size. This relationship is shown visually in Figure 1.

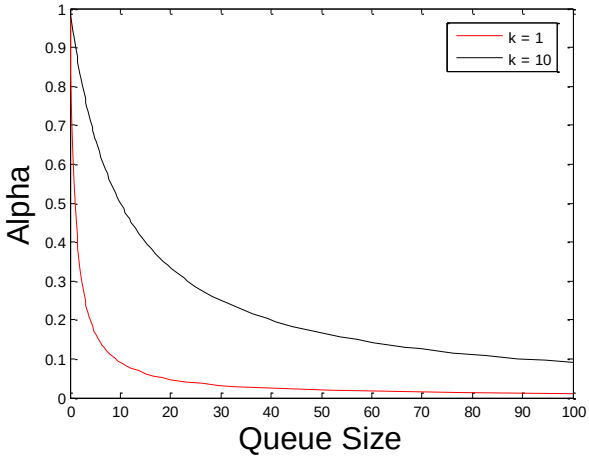



**Figure 1: Comparison of the change in alpha value with respect to queue size as the parameter k is varied.**

A similar set-up was used for the second model which is dictated by the differential equations

$$\frac{dS_i}{dt} = -k_1 - \alpha\big(S_i(t)\big) + \alpha\left(S_j(t-\tau)\right)$$

$$\alpha\big(S_i(t)\big) = \frac{S_i(t)}{S_i(t)+k_2}.$$

A constant processing power was assumed, and message transfer rate dictated by the function alpha. The relationship between alpha and queue size can be seen in Figure 2.

## 3. RESULTS AND DISCUSSION

The results of the first two experimental models are simply presented while the results from work with the cloud computing models are discussed throughout the body.

### 3.1 Two-delay of El-Nino – La-Nina

The model was solved using the MATLAB command ddensd, twice, for each of the instances in Table 1. The resulting plots are shown from t=0-3 in Figures 2-6.

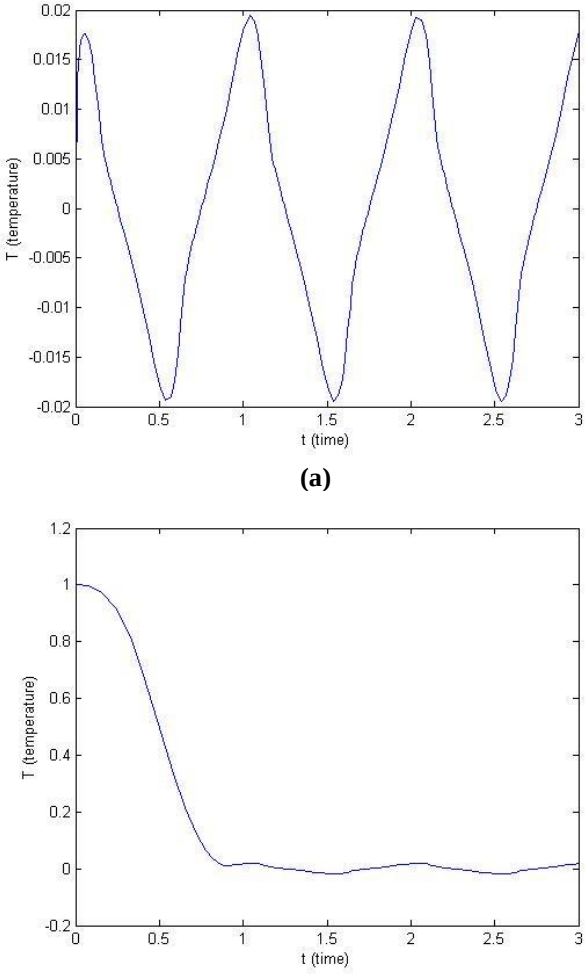
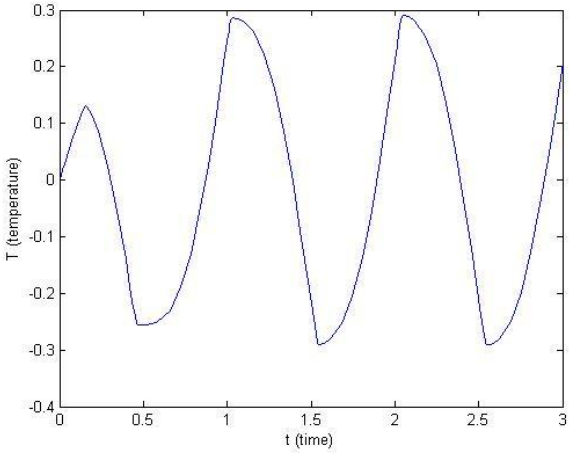
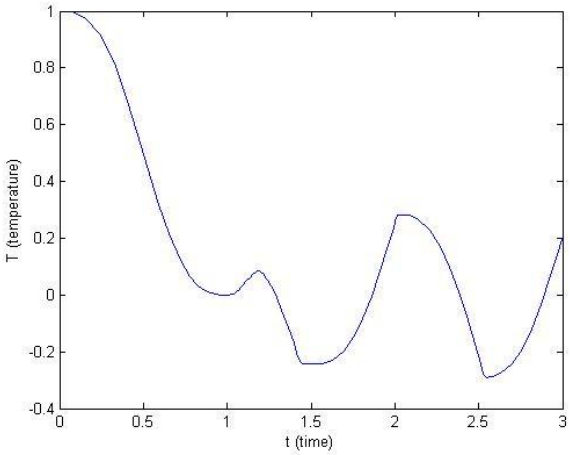



**Figure 2: Instance 1 of the El-Nino – La-Nina model with (a) showing the results of h(t) = 0 and (b) showing the results for h(t) = 1.**

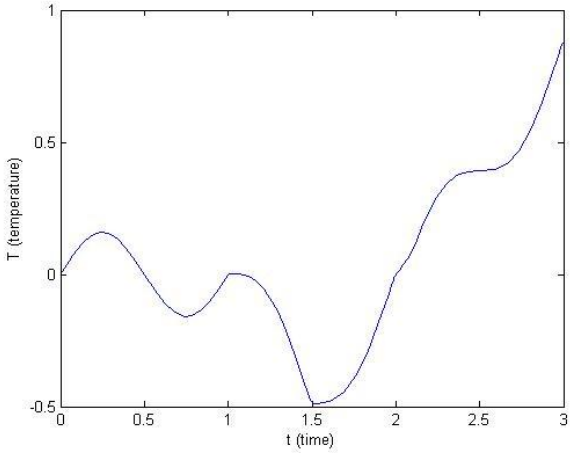



**(a)**

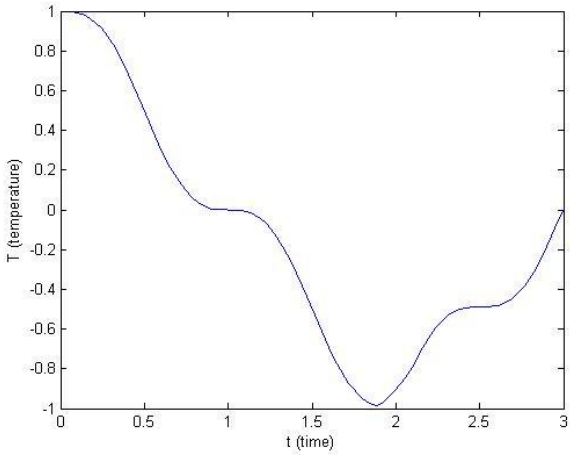



**(b)**

**Figure 3: Instance 2 of the El-Nino – La-Nina model with (a) showing the results of h(t) = 0 and (b) showing the results for h(t) = 1.**

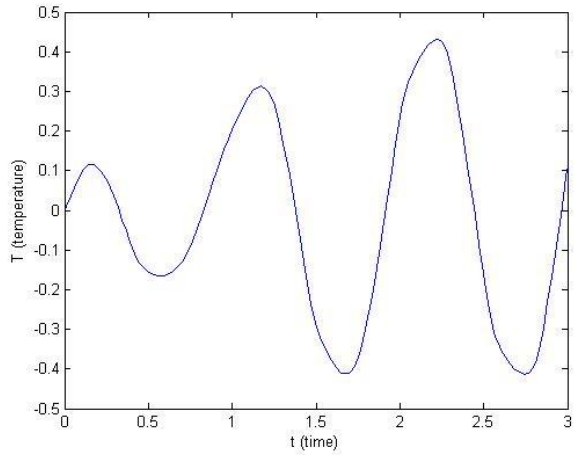



**(a)**

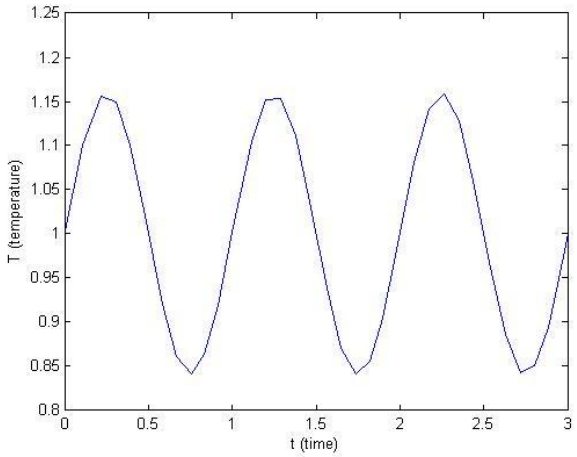



**(b)**

**Figure 4: Instance 3 of the El-Nino – La-Nina model with (a) showing the results of h(t) = 0 and (b) showing the results for h(t) = 1.**

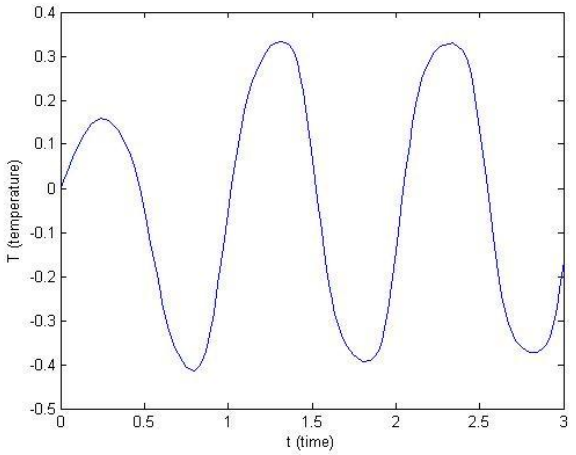



**(a)**

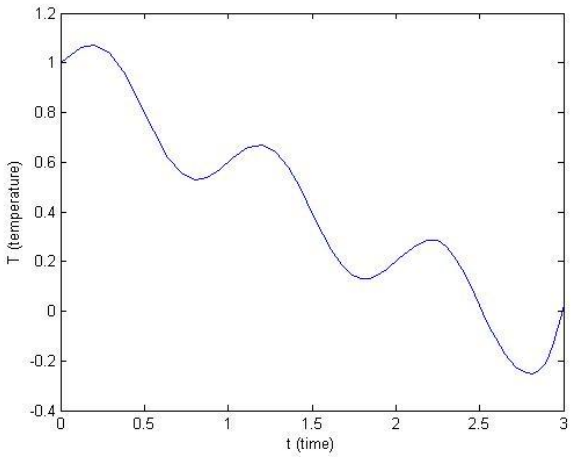



**(b)**

**Figure 5: Instance 4 of the El-Nino – La-Nina model with (a) showing the results of h(t) = 0 and (b) showing the results for h(t) = 1.**




**(a)**




**(b)**

**Figure 6: Instance 5 of the El-Nino – La-Nina model with (a) showing the results of h(t) = 0 and (b) showing the results for h(t) = 1.**

### 3.2 Partial Element Circuits

The model was solved using the MATLAB command ddensd. The input lag was 10 time units and he resulting plot is shown in Figure 7.

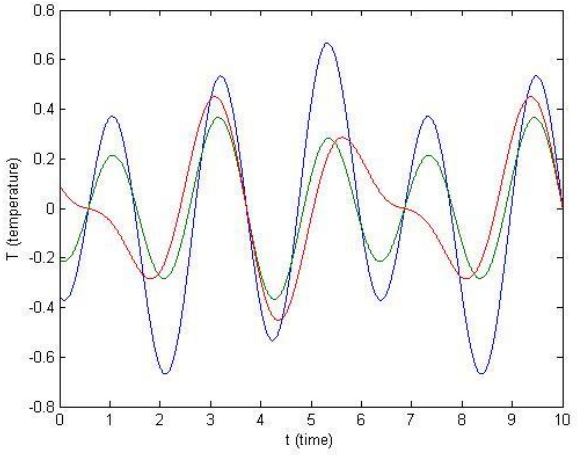
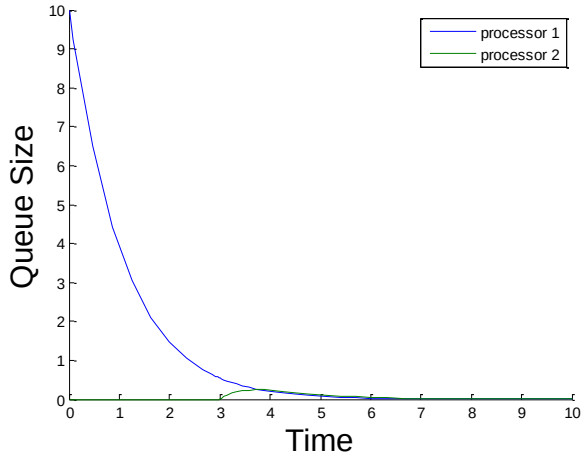



**Figure 7: Partial element circuit response with a sinusoidal history function and a time delay of 10.**

### 3.3 Cloud Computing

In our initial model, the rates of sending, receiving, and performing tasks depends on the value of the function alpha.

In figure 9 at time t = 0 processor one's queue was set to be 10, and the time delay was three units. Since k is larger in figure 8b, more packets are sent to processor two. In this scenario, it is not advantageous for the processor to send packets.




**(a)**

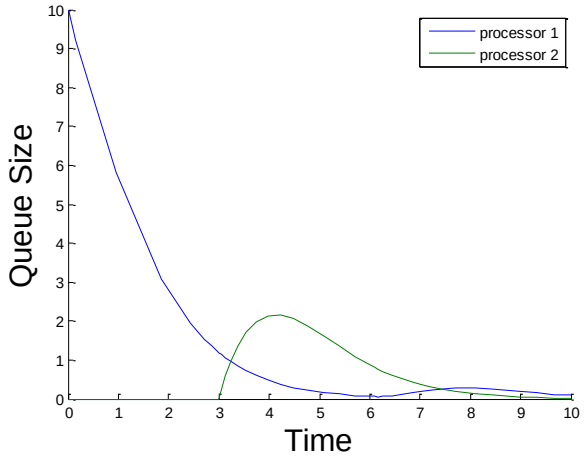
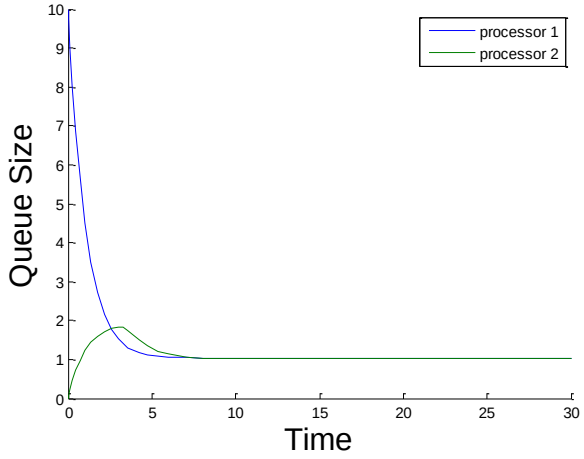
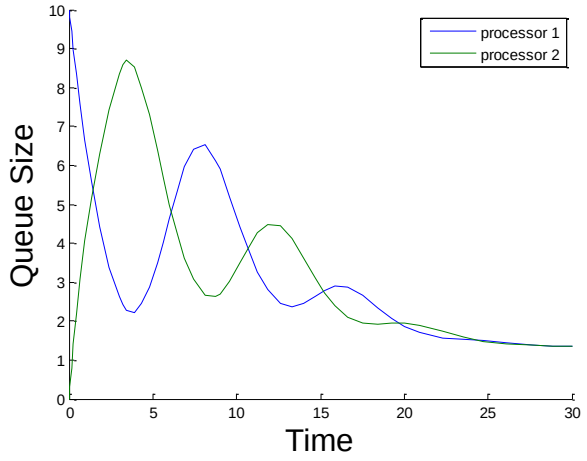



**Figure 8: Cloud computing response with queue size of 10 at t = 0 for processor 1. In Figure 8a k = 100, and in figure 8b k = 10. The time delay was three units of time.**

Another scenario involving this model is depicted in Figure 9. The queue size for processor 1 was 10 for $t \leq 0$, and the queue size for processor 2 was 0 for $t \leq 0$. A forcing function of $f(t) = 1$ was in both differential equations. In (a) k =100 and in (b) k =10. Both reach the same steady state.




**(a)**




**(b)**

**Figure 9: The forcing functions for the processors was f(t) = 1. In (a) k = 100, and in (b) k = 10.**

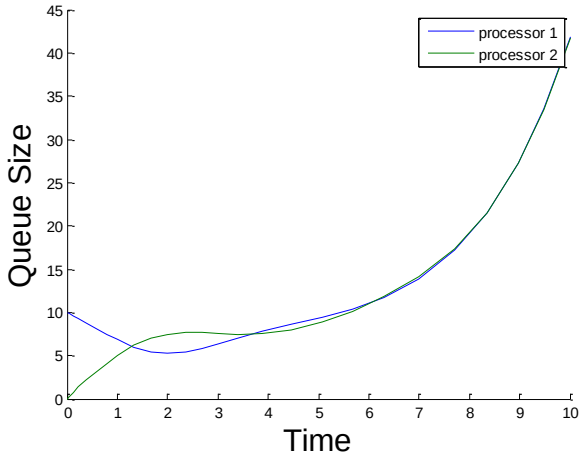



**Figure 10: Artificial packets are created and the queues increase exponentially.**

As in the previous example, steady state was reached faster when sending fewer packets.

In this model, there is never a scenario where sending packets speeds up the reduction in queue size, and reaches steady states faster. Tests involving additional processors, different forcing functions, different rates of processing speed, and different time lags were performed. None were able to demonstrate any improvement in processing time by sending packets. The best way to reduce the queue size was always to have as high value of k as possible, so that the node would process the packets itself.

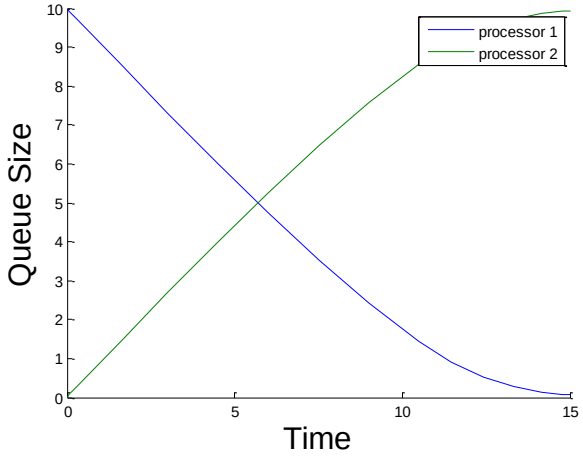
Moreover, this model creates new, artificial packets when sending. To illustrate this, consider the scenario when alpha is zero. The processor's own queue size is not being reduced, even though packets are appearing on the other processor af ter a time lag (at an exponential rate).

In Figure 10, k=5, processors one's queue is 10 initially, and processor two's queue is empty. The forcing function is zero so that no new packets are added. However, the differential equation reaches a point where it is receiving more packets than it can send and receive. Both of the processor's begin to have a large enough queue so that they are only trying to send. Since no packets are added to the system, packets are being artificially created in this model.

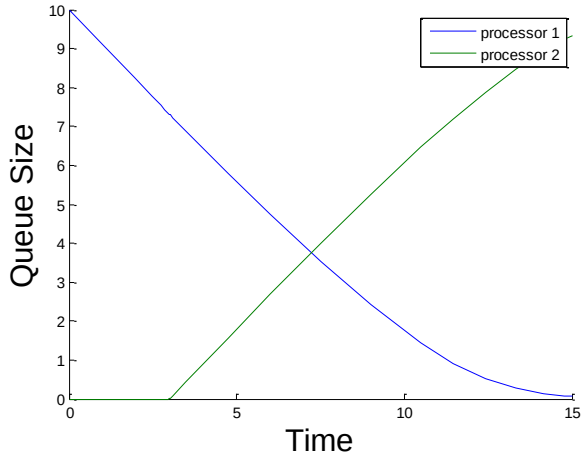
In the second model, the same alpha function was used for the amount of packets sent:

$$\alpha(S_i(t)) = \frac{S_i(t)}{S_i(t) + k_2},.$$

In Figure 11, the queue size of processor 1 is 10, and the queue size of processor 2 is 0. Packets cannot be processed, and they can only be sent from processor 1 to processor 2. In figure 11b, a delay was added in receiving packets. These graphs prove that packets are not being created in this model.

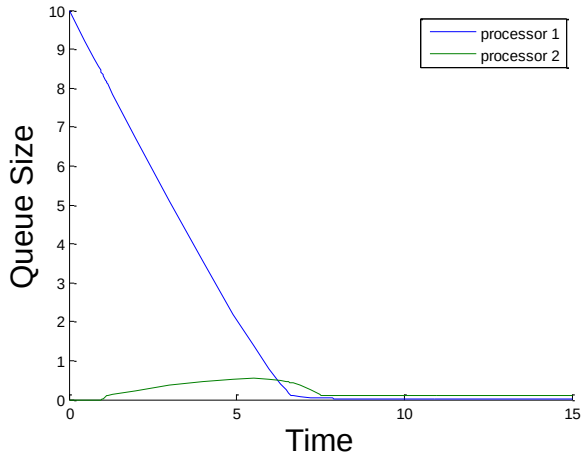



**(a)**

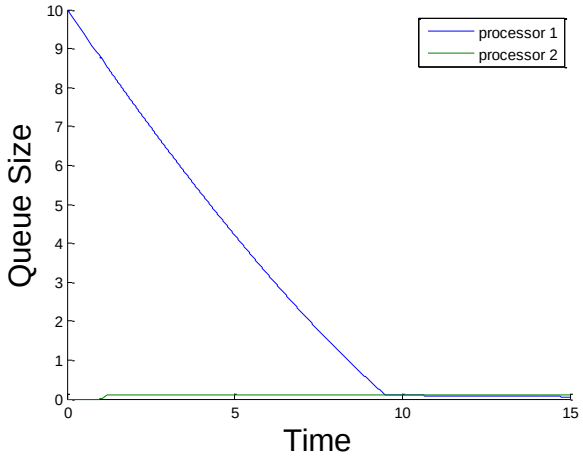



**(b)**

**Figure 11: (a) k = 1, b k =10**




**(a)**




**(b)**

**Figure 12: (a) k = 1. (b) k =10**

Figure 12 demonstrates a simulation of the second model at differing k values. We now can see the evidence of sending packets improving the run time. The queue sizes approaches zero faster at a higher value of k for the first time. This means that sending tasks improves run time.

.