SYSTEMS PROGRAMMING MIDTERM PROJECT REPORT

Fatih Selim YAKAR - 161044054

Systems Programming Midterm Project Report

Synchronization Problems Between the Supplier and the Cook

In supplier, I had a synchronization problem for the supplier and the cook to enter the kitchen at the same time and there was also the possibility that the kitchen was full or empty. I solved the problem this way:

I thought the supplier is producer and the cook is consumer, also the kitchen is mutual area. In this case, the supplier will put the food if there is free space in the kitchen and is not used by the cook, on the other hand, the cook will take the food if the kitchen has food and the supplier is not in the kitchen.

To ensure this, I prevented entering the kitchen at the same time with a mutex called kitchen_mutex. With the 2 full_kitchen and empty_kitchen semaphores, I have ensured that the supplier does not put food while the kitchen is full and that the cook does not get food when the kitchen is empty.

Synchronization Problems Between the Cook and the Student

The cook had to take food from the kitchen on the one hand, and on the other hand, if there was free space on the counter at the same time, he/she had to give it there. Already, he/she could not access cook kitchen at the same time as the supplier. I used a mutex named cook_mutex so that 2 cooks could not work at the same time to avoid confusion since it would do a lot of processing in itself later. In other words, the cook takes the import from the kitchen and gives the countera between cook_mutex. In other words, it performs the put and get operations from cook kitchen between cook_mutex.

Also, since Student will eat 3 foods at the same time and in this case deadlock may occur, I gave P, C and D in order, so that 3 foods would definitely come if he/she waited. It was thanks to 2*L*M+1 size, which was given as a parameter in order not to have any problems about to the give food. Thus, even if L*M times P, L*M times C and 1 desert were given by the supplier, I was able to get 3 meals if expected due to +1. So the cook was able to give it while getting it at the same time.

Then the same problem between supplier and cook happened between student and cook. In other words, when getting students, the counter should not be empty and there should be at least 1 of each food. And they should not be able to enter the counter at the same time to avoid confusion. On the other hand, while the cook was putting, they should not be counter full and not be able to enter at the same time.

To solve this, I used a mutex named counter_mutex to avoid entering it at the same time as cook and supplier, and also 2 full_counter, empty_counter named semaphores so that the cook does not put when the counter is full and the student does not get and eat it empty.

Synchronization Problems in Tables

If the table was just full, the student had to wait until it was empty. To achieve this, I used only a semaphore name full_table.

Used Functions

void print_error(char error_message[]): Prints error in STDERR by using write syscall.

void s_wait(sem_t *sem): Performs sem_wait with error control.

void s_post(sem_t *sem): Performs sem_post with error control.

int s_getval(sem_t* sem): Returns the integer value of the given semaphore as parameter.

ssize_t read_lock(int fd, void *buf, size_t count): Reads with locking.

void supplier_service_soup(): The supplier puts soup (posts the soup semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void supplier_service_mc(): The supplier puts main course (posts the main course semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void supplier_service_desert(): The supplier puts desert (posts the desert semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void cook_get_soup(int index): The cook gets soup (waits the soup semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void cook_get_mc(int index): The cook gets main course (waits the main course semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void cook_get_desert(int index): The cook gets desert (waits the desert semaphore) in the kitchen by using kitchen_mutex and empty_kitchen,full_kitchen semaphores.

void cook_give_soup(int index): The cook puts soup in the counter (posts soup semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void cook_give_mc(int index): The cook puts main course in the counter (posts main counter semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void cook_give_desert(int index): The cook puts desert in the counter (posts desert semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void student_eat_soup(int i,int round): The student gets desert in the counter (waits soup semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void student_eat_mc(int i,int round): The student gets main course in the counter (waits main course semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void student_eat_desert(int i,int round): The student gets desert in the counter (waits desert semaphore) in the counter by using counter_mutex and empty_counter,full_counter semaphores.

void supplier(int input_fd,char input_file_name[],int l,int m): The function that performs the job of the supplier process. It read 3*l*m character in the given input file and when he/she reads, he/she classifies the letter he/she reads as food and puts it in the kitchen.

void cook(int i,int l,int m): The function that performs the job of the cook process. Until the number of food he/she gives and receives by all the cooks reaches 3*L*M, he/she takes food from the kitchen in P, C, D order and puts the food he/she takes at the same time.

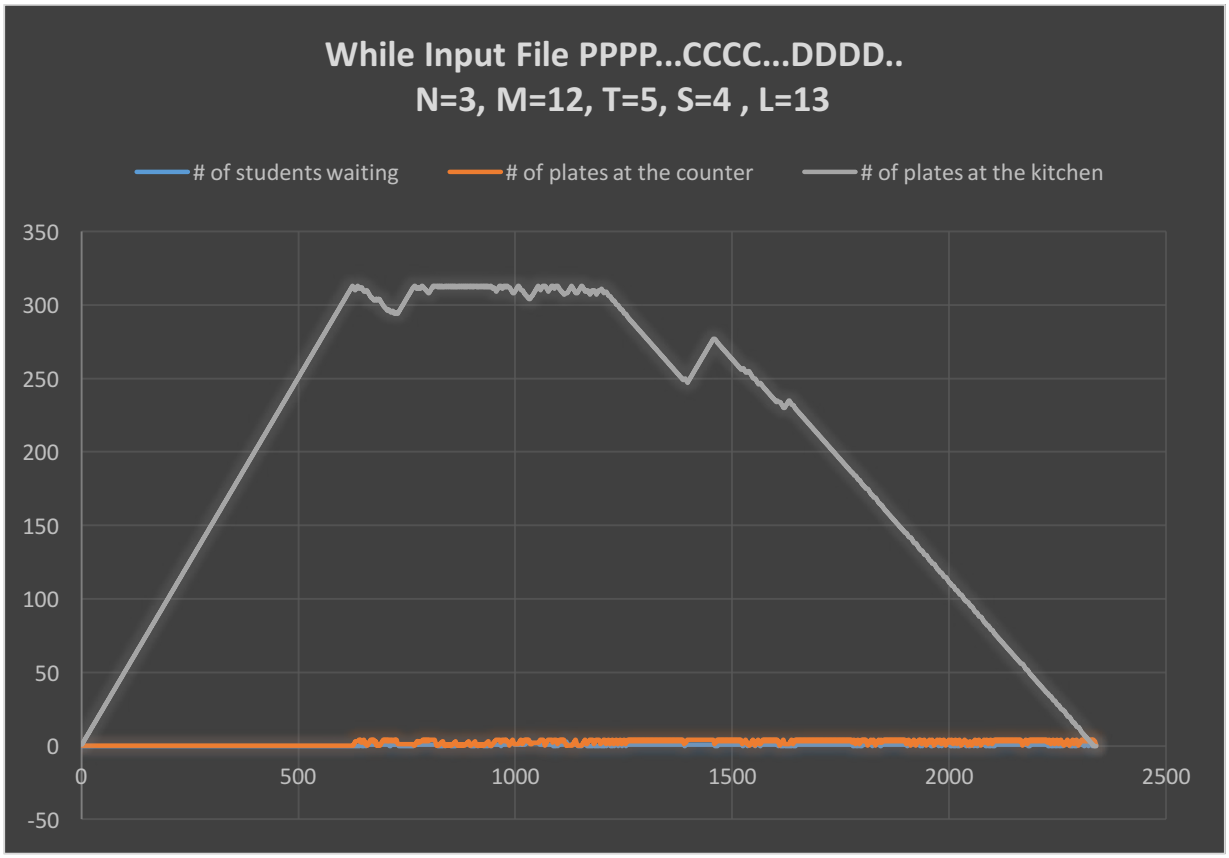
void student(int i,int l,int m,int t): The function that performs the job of the student process. Until all the food that students eat is 3*L*M, they take 3 of the three food (they reduce the semaphores on the counter), then they go to the table to eat their food (it reduces the table semaphore) and stays there. It does this L (the limit of 1 student's loop) times.

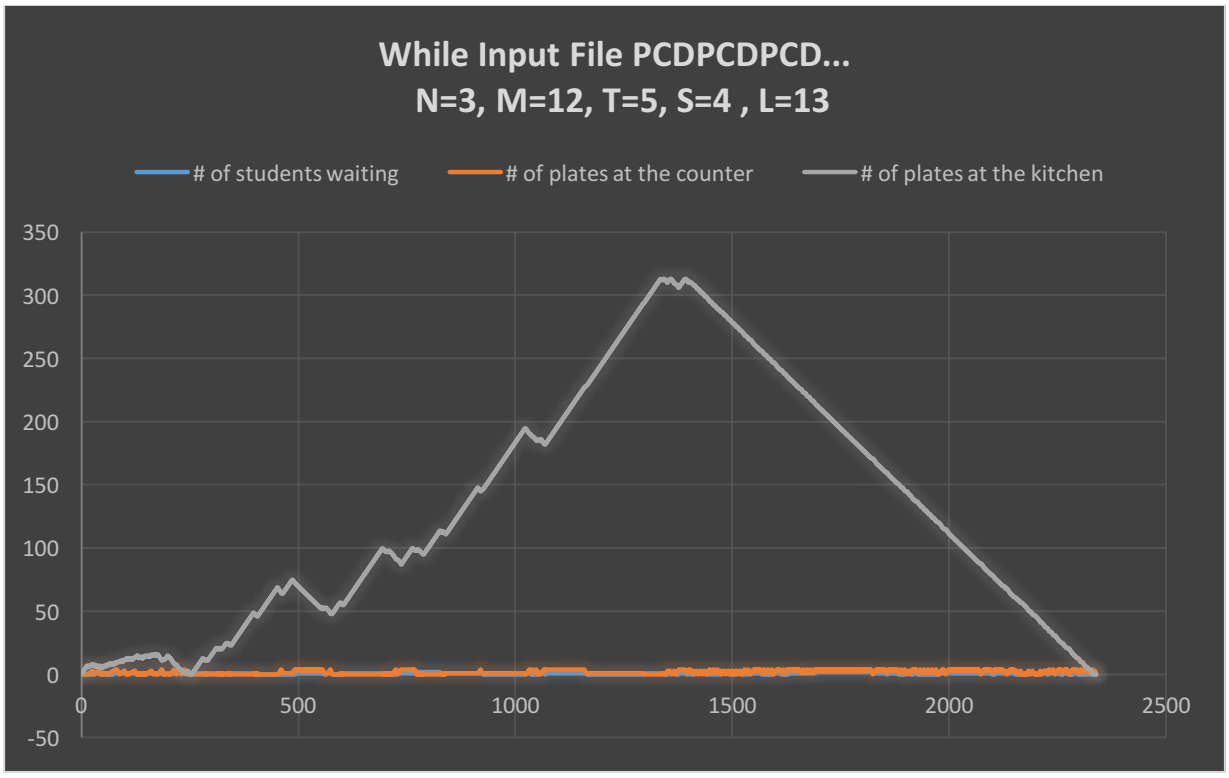
void sigint_handler(int signum): When the signal is caunght, it exits by freeing all sources.

int main(int argc,char *argv[]): Controls the command line arguments and initialize the all resources with these, creates the Supplier, Cook and Student process'es by using given parameters, frees resources and exits.

NOTE: You should run this code with “sudo” command.

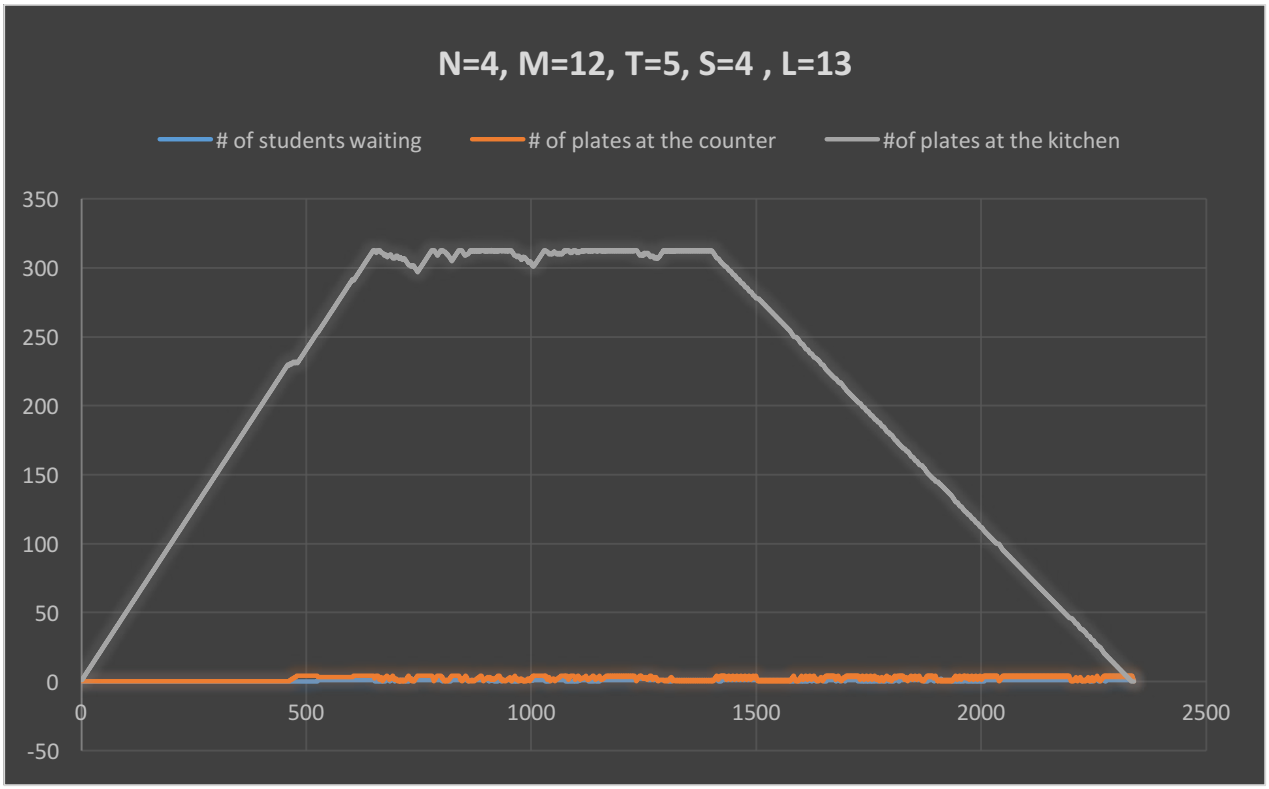
Plots

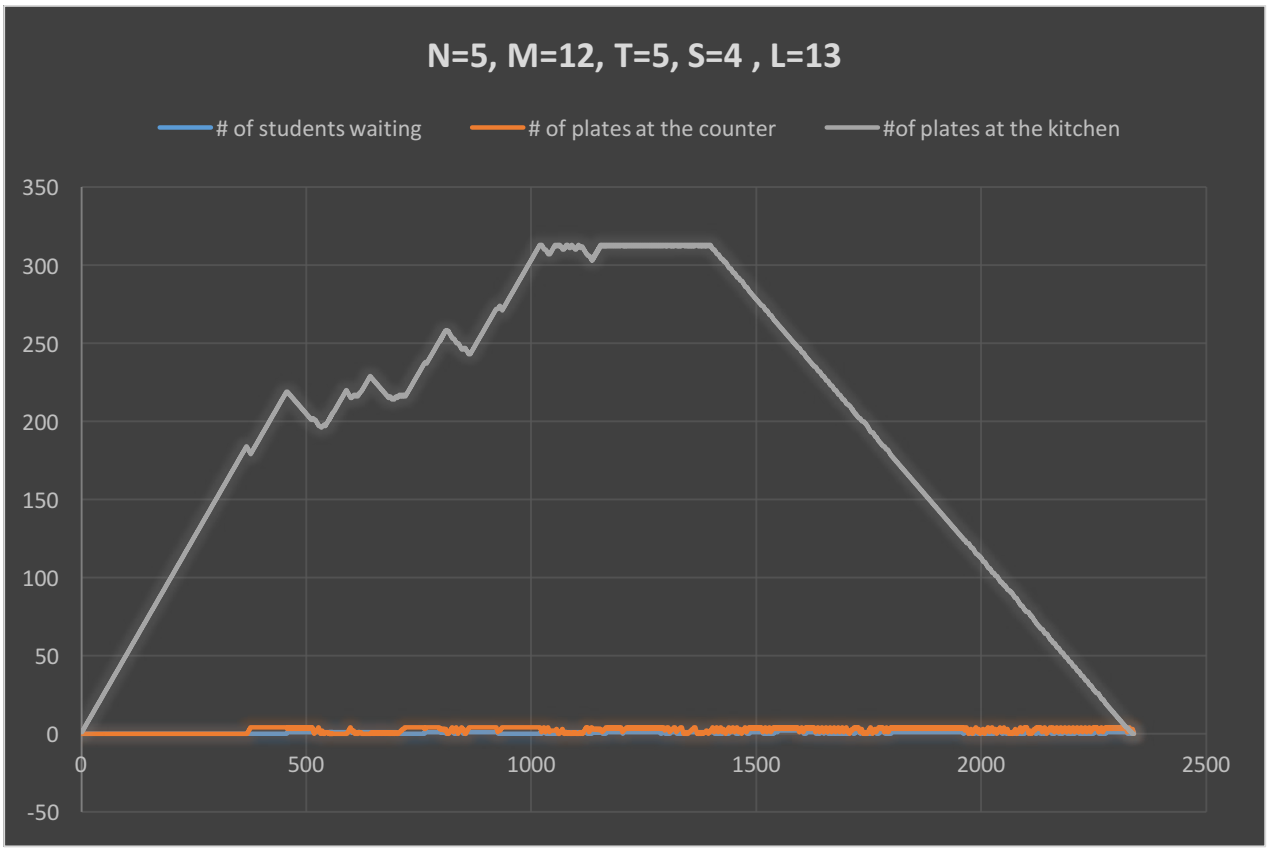


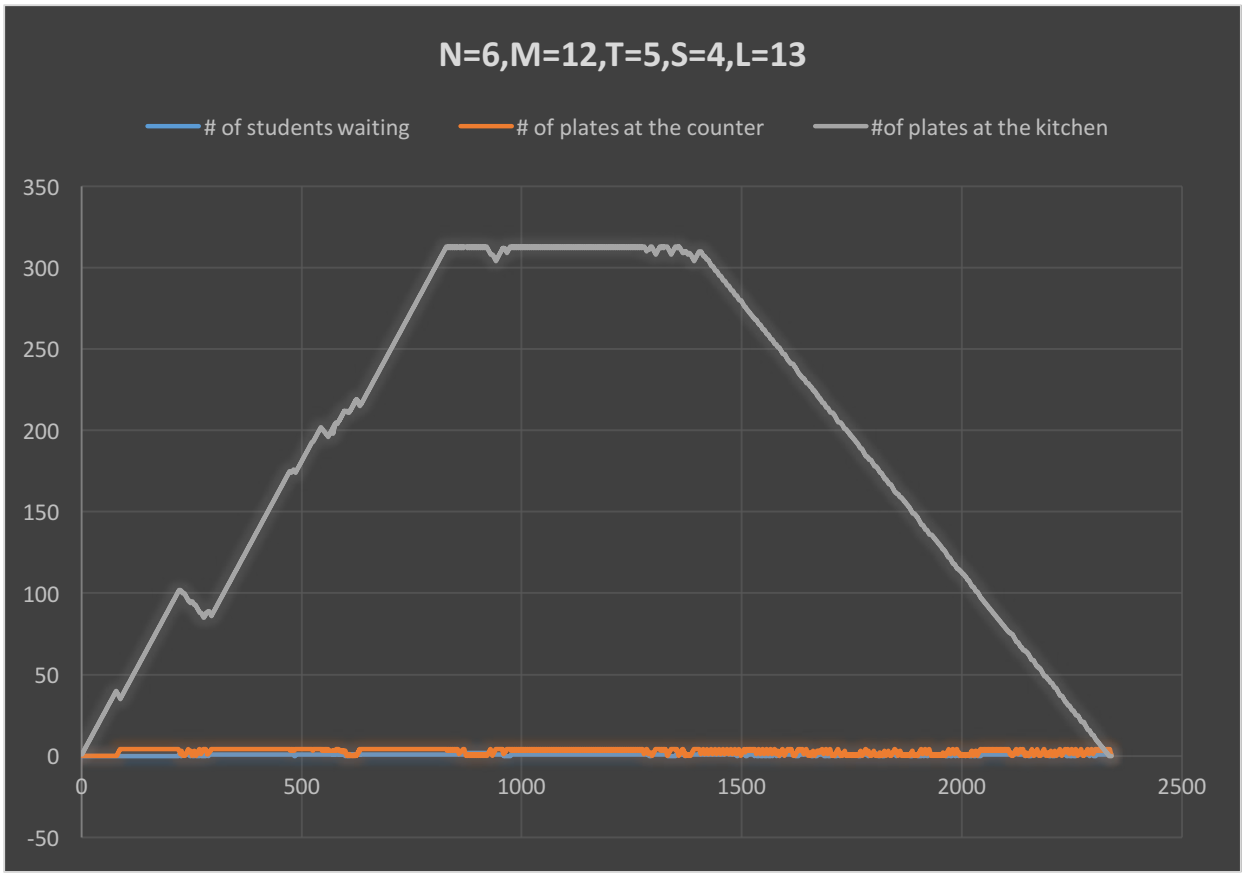


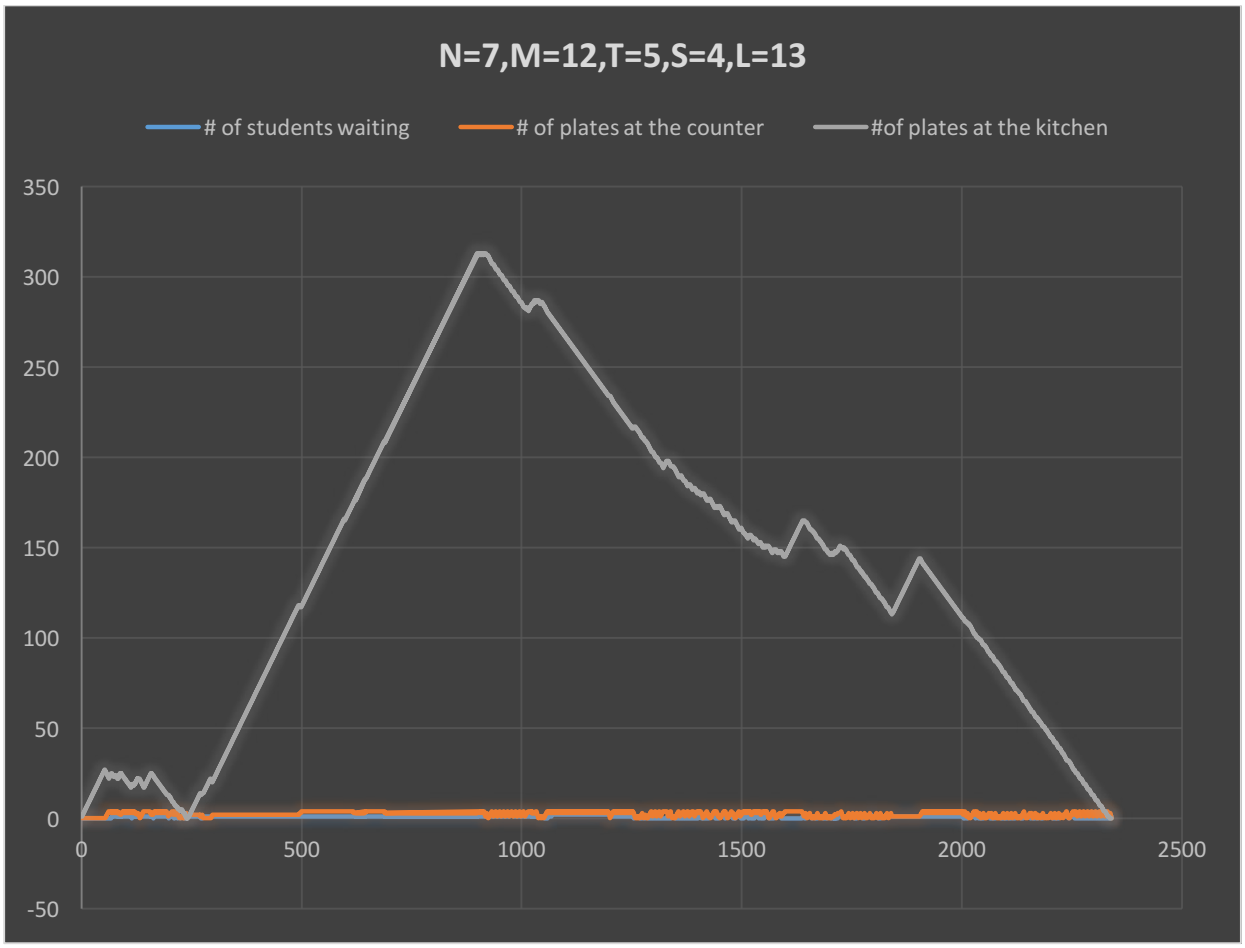
All the plots below were drawn according to the PCDPCD .. input.

N is variable and M,T,S,L are constant

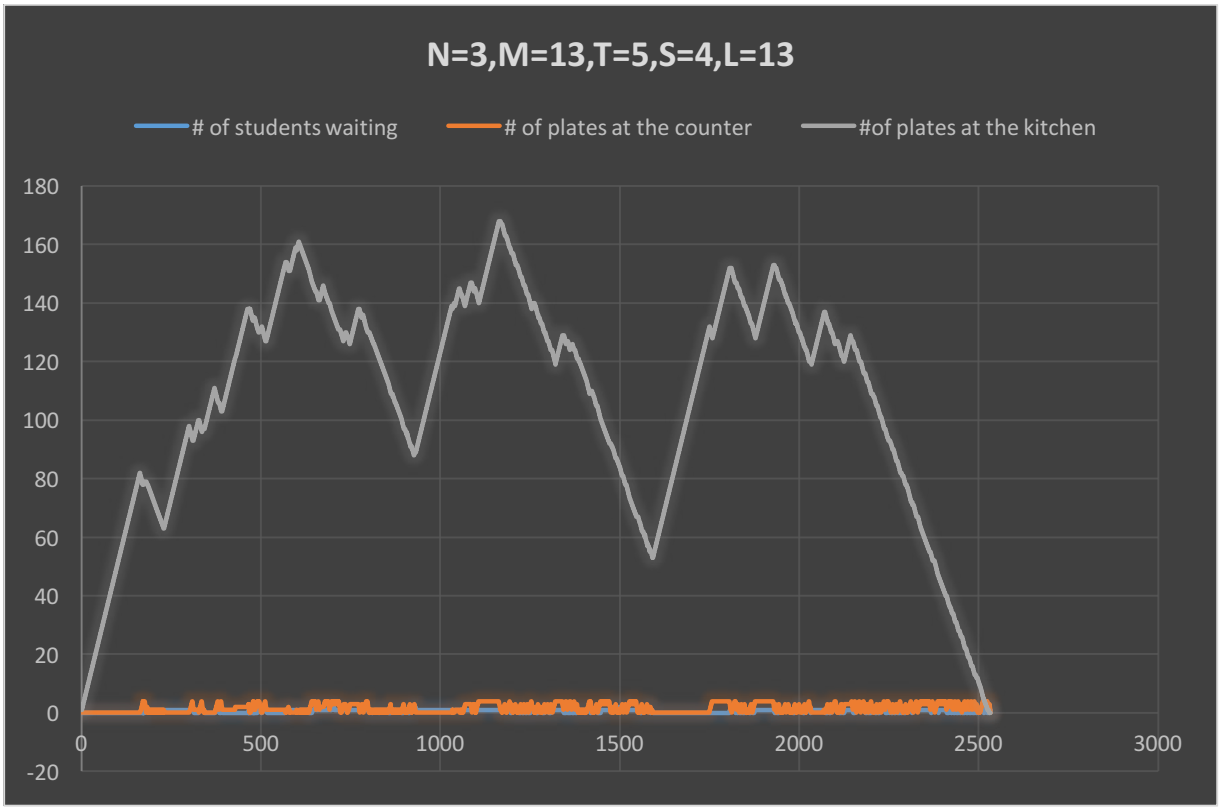


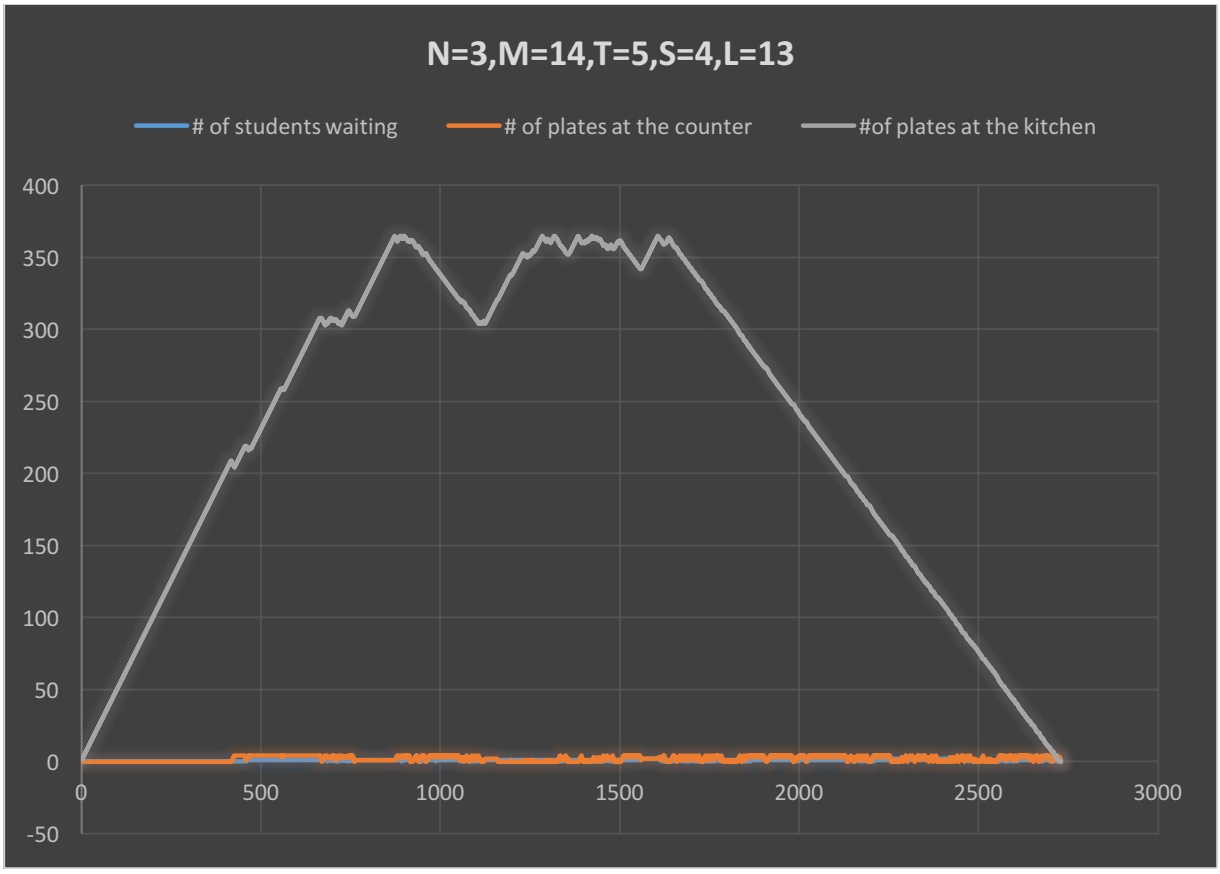


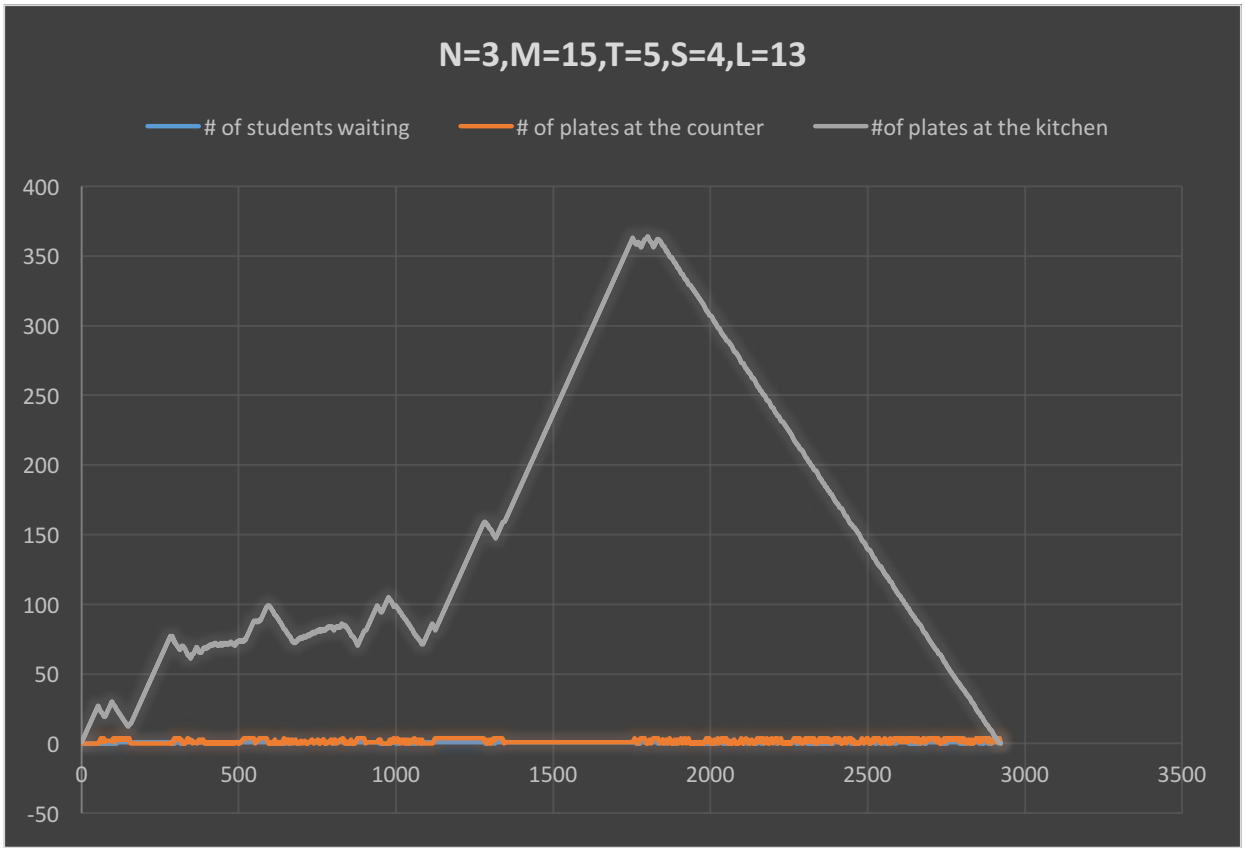

N=6,M=12,T=5,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500

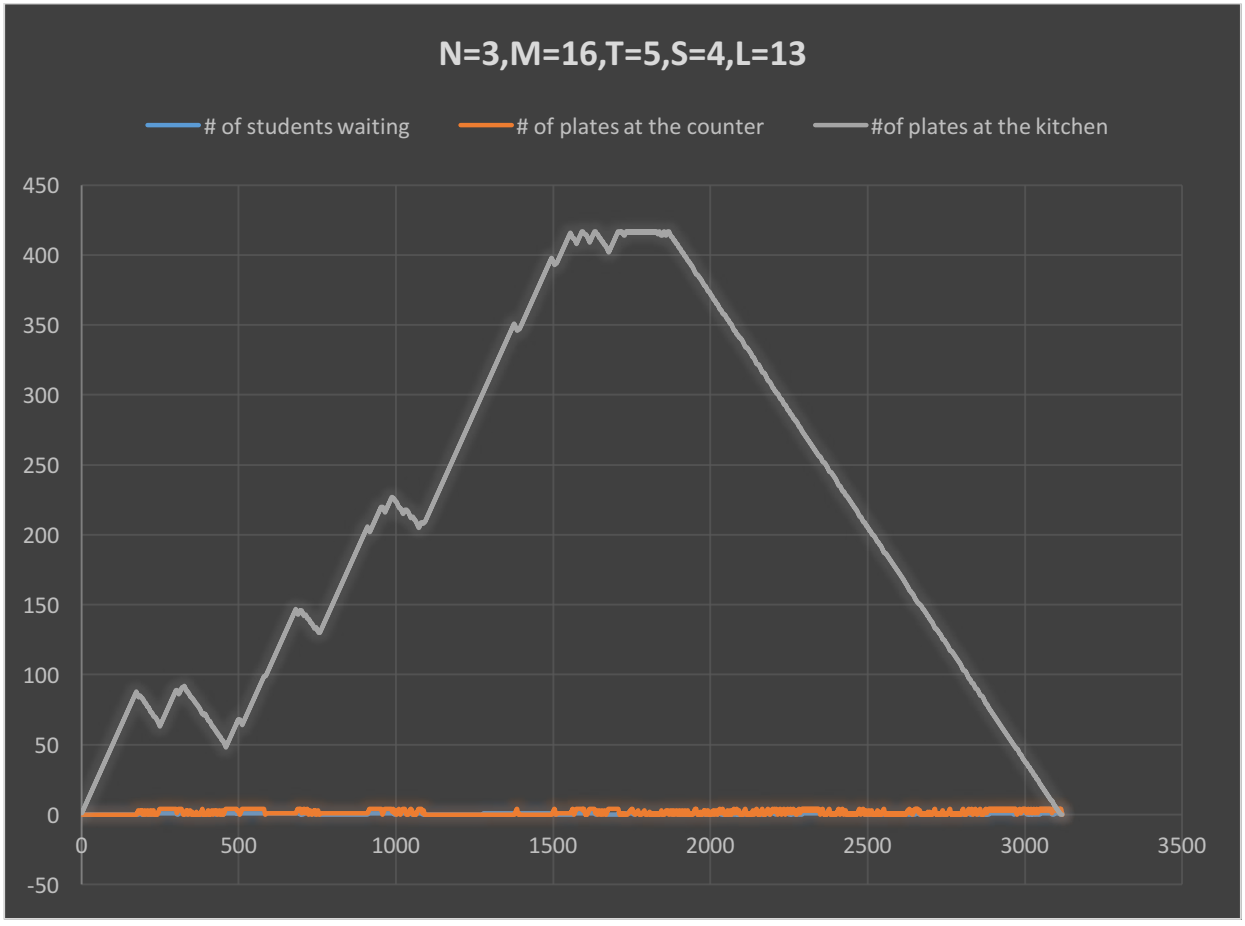

N=7,M=12,T=5,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500

M is variable and N,T,S,L are constant

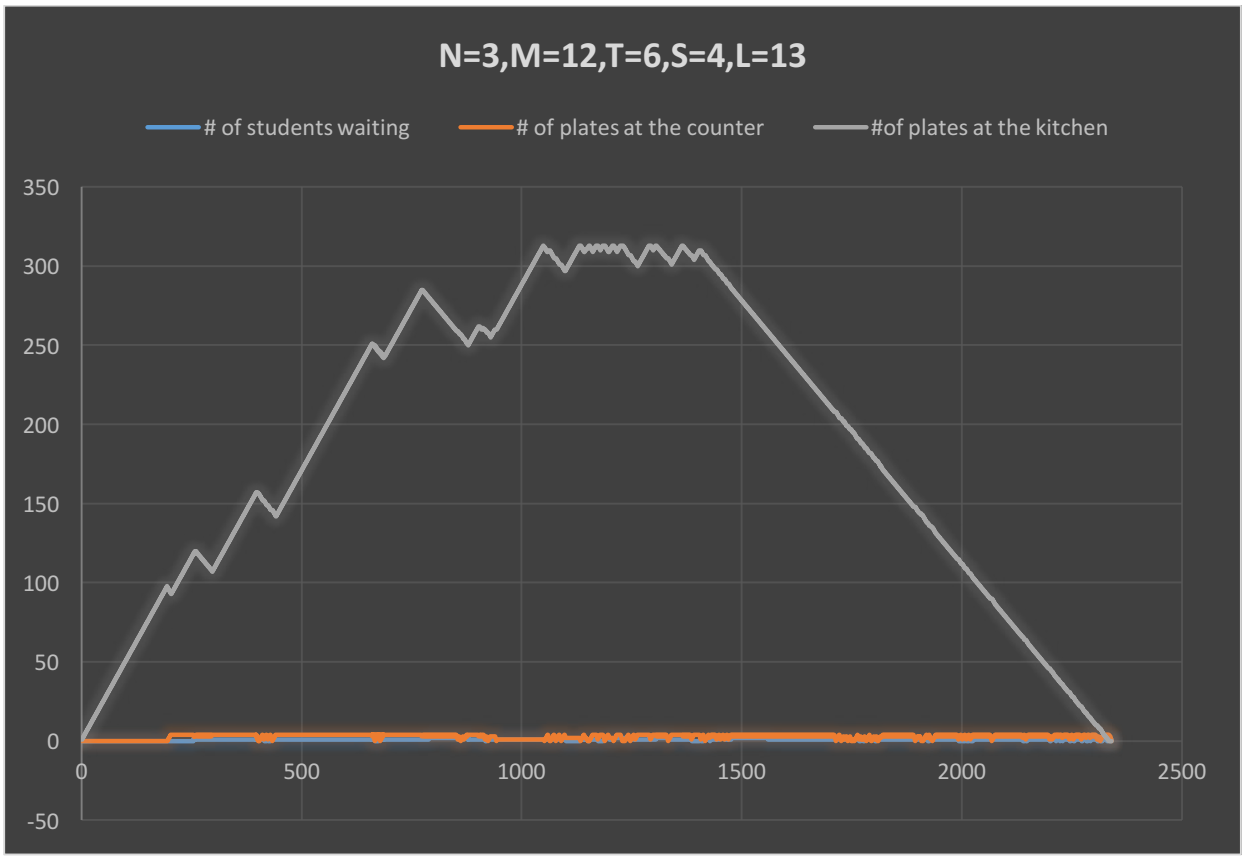


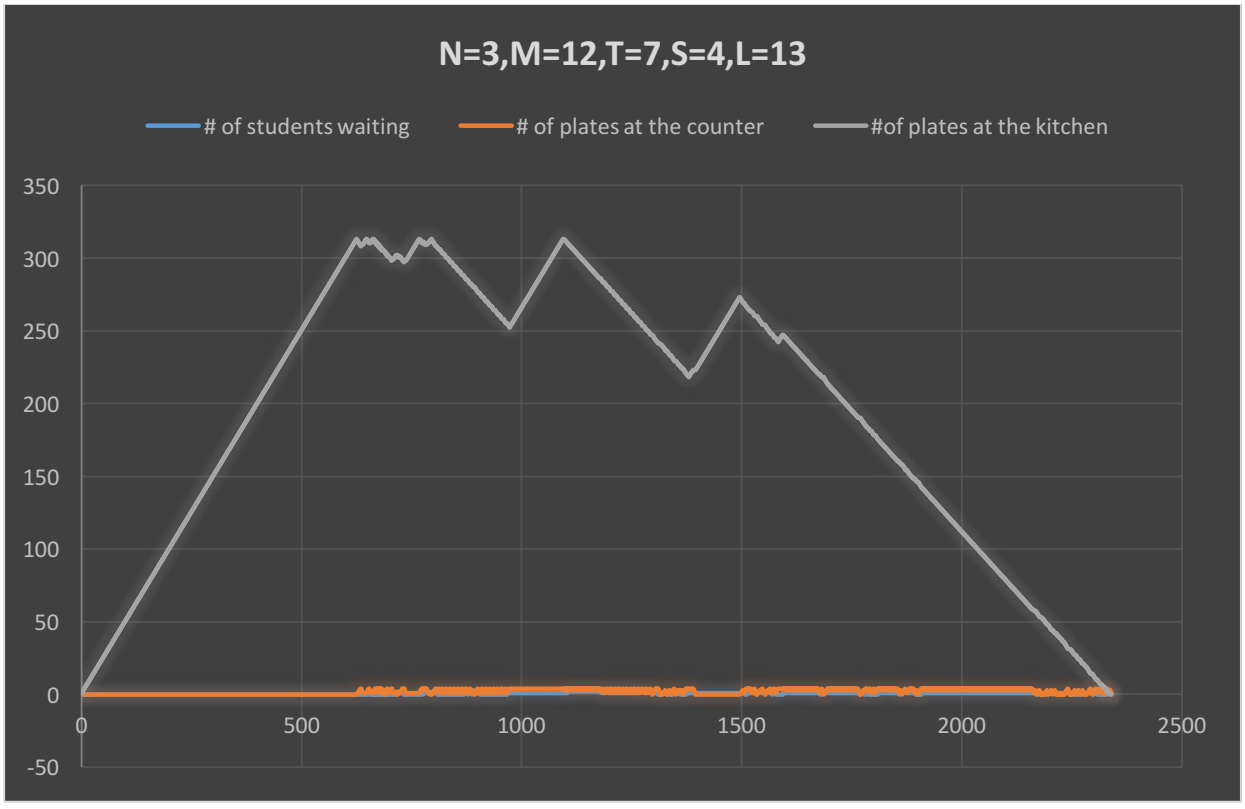


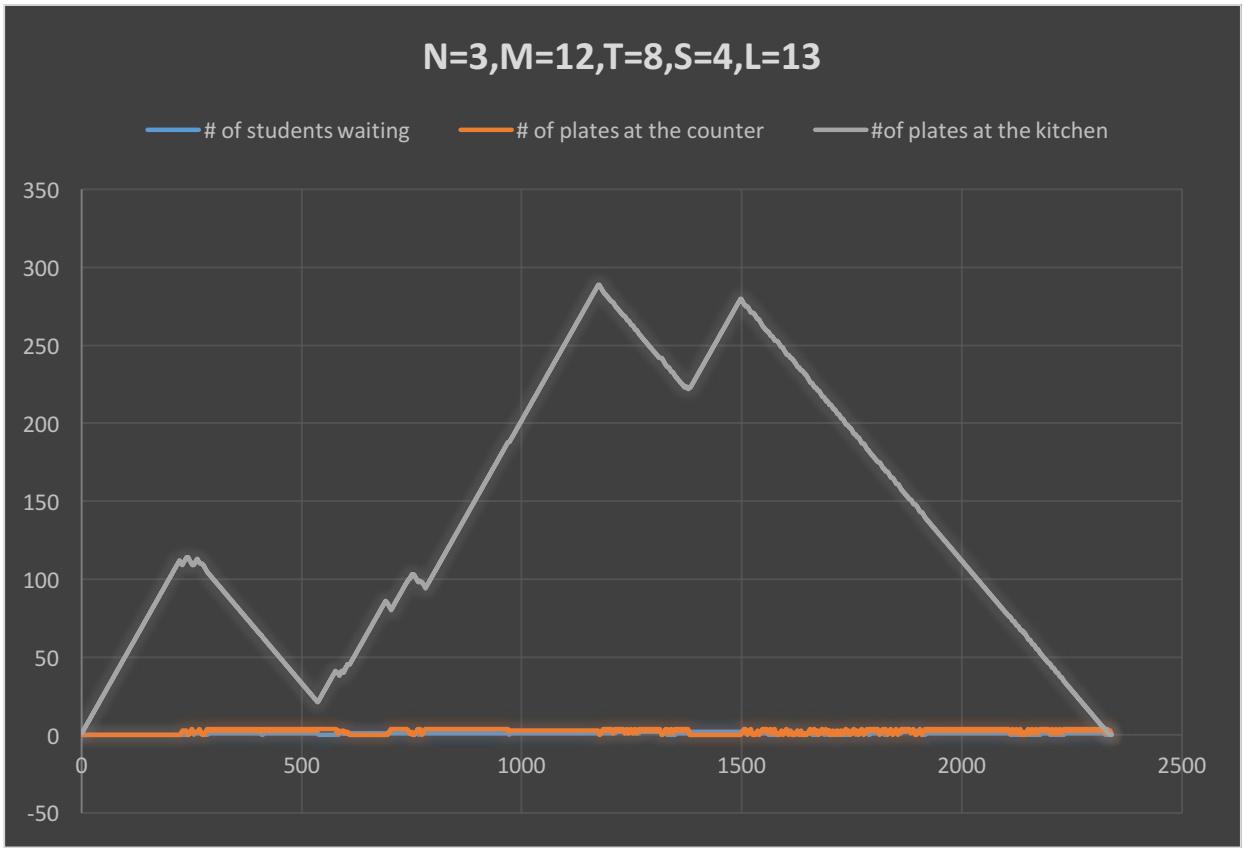

N=3,M=15,T=5,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
400
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500
3000
3500

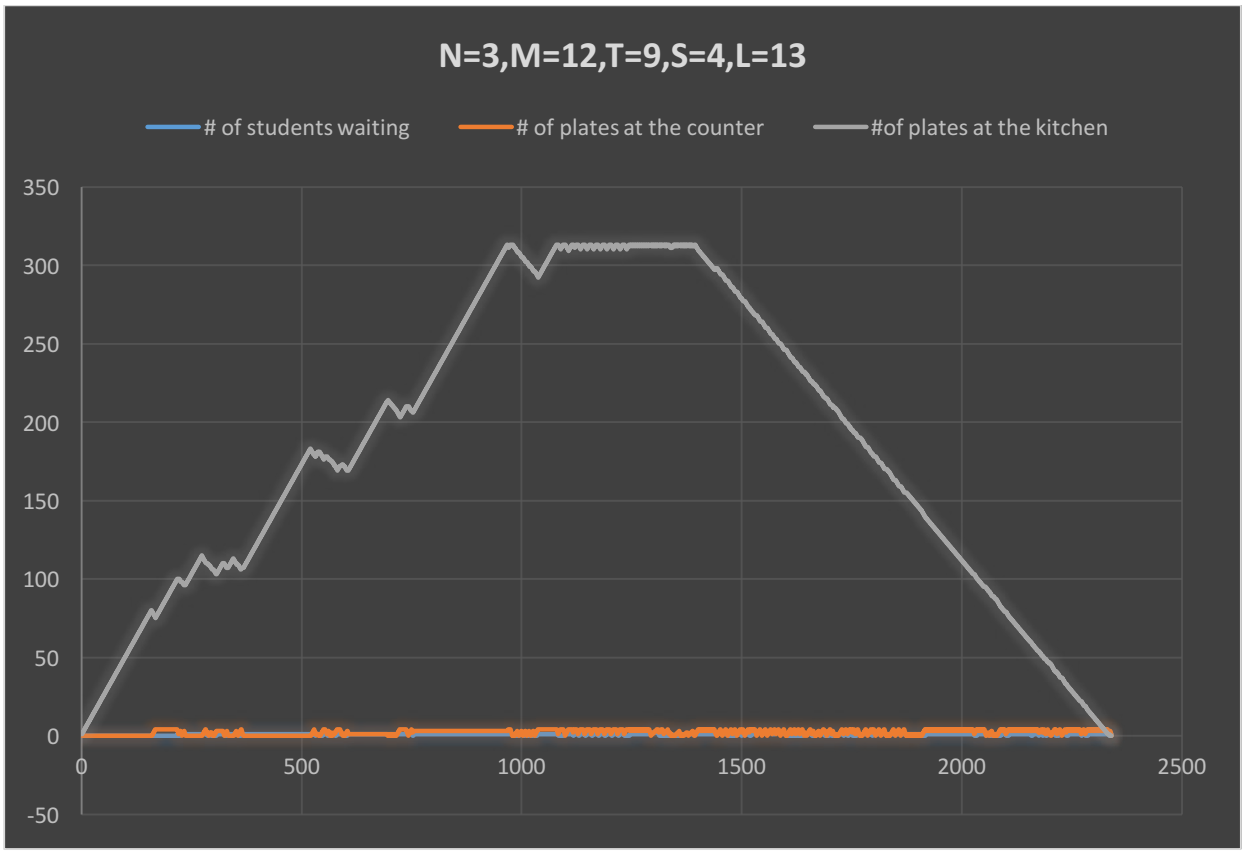

N=3,M=16,T=5,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
450
400
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500
3000
3500

T is variable and M,N,S,L are constant

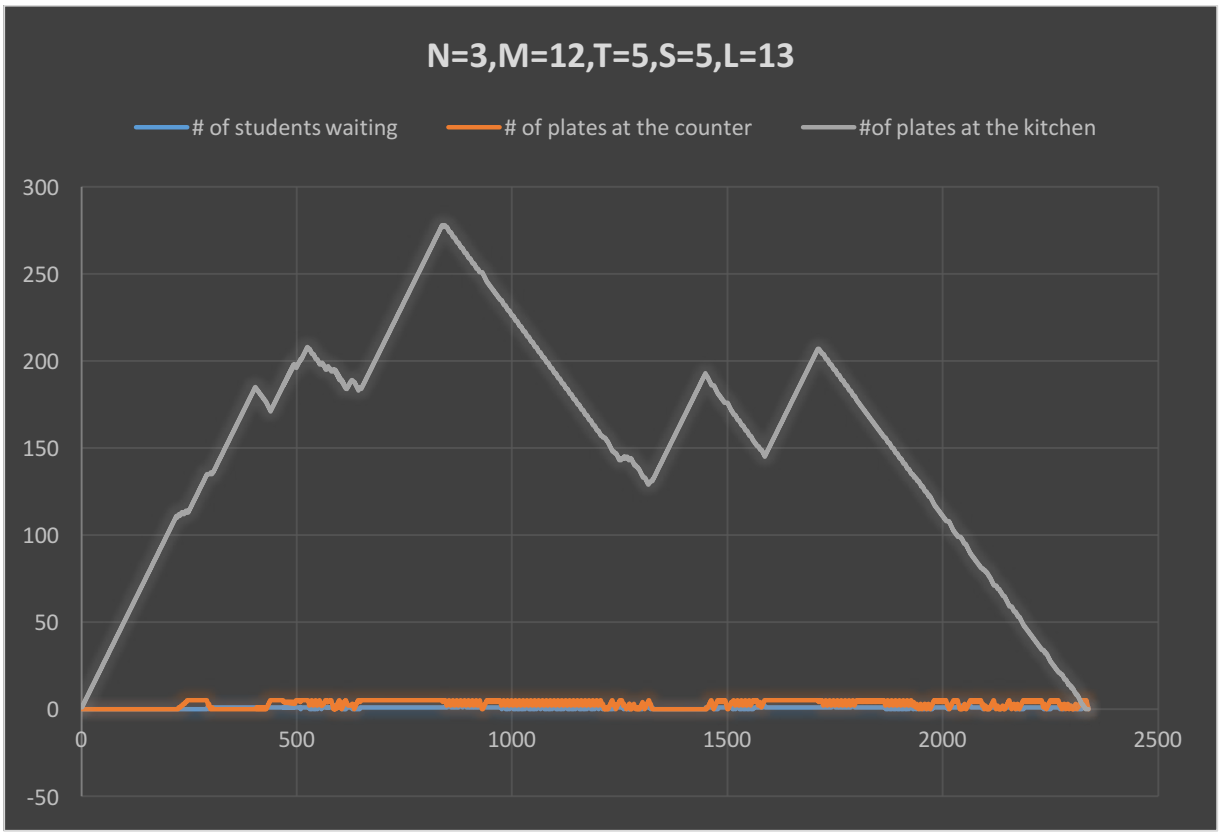


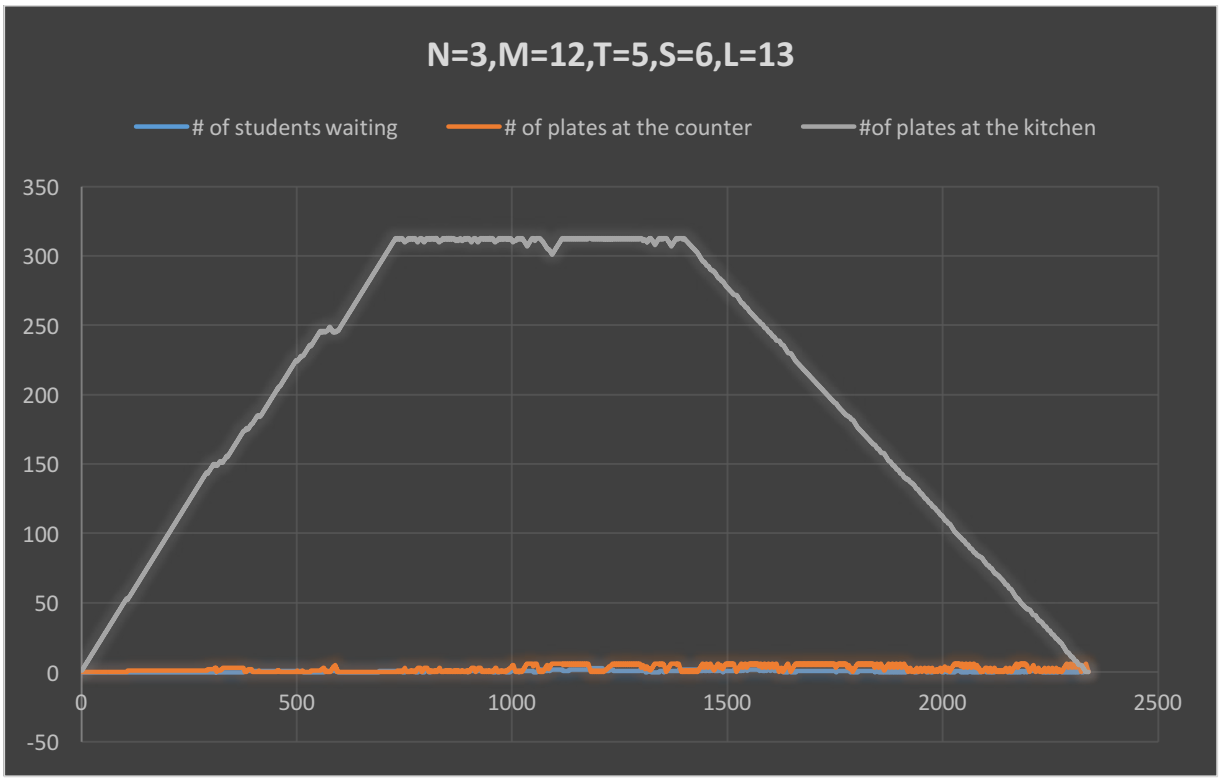


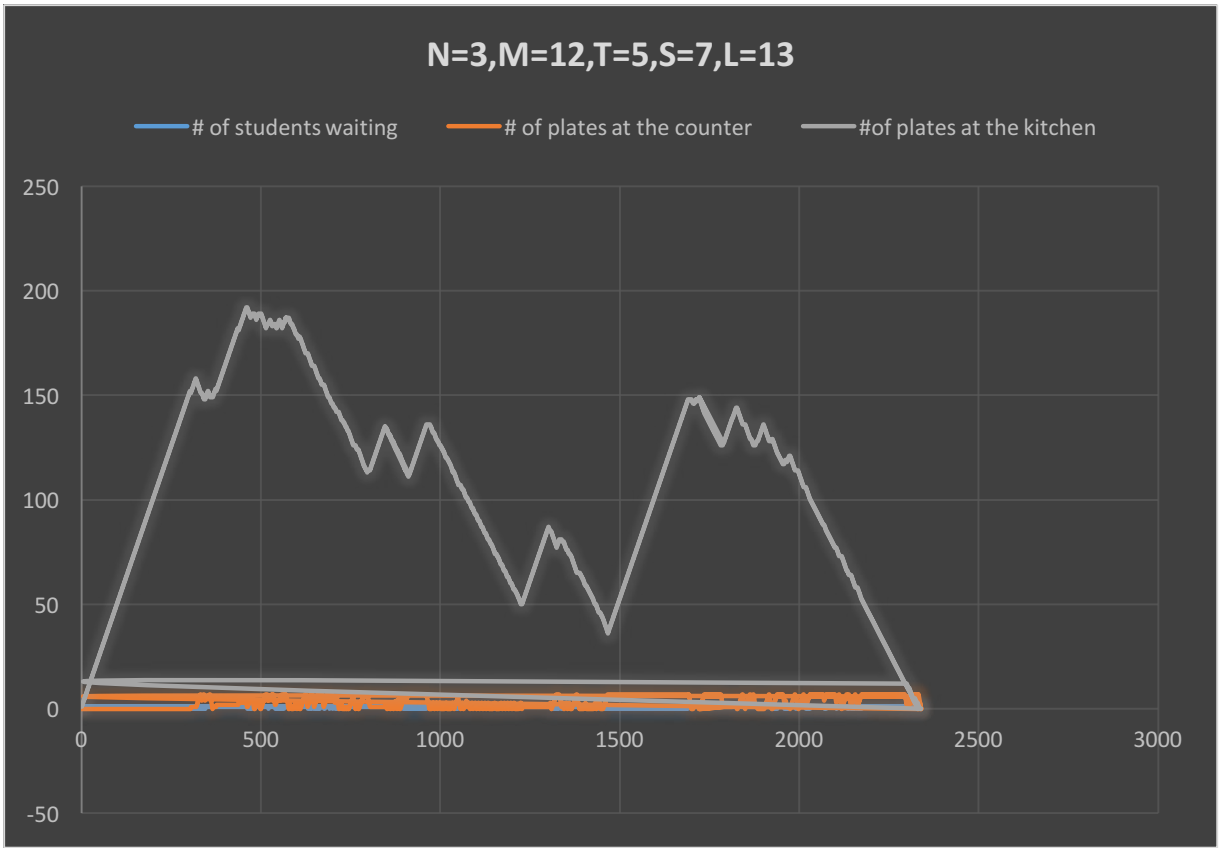

N=3,M=12,T=8,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500

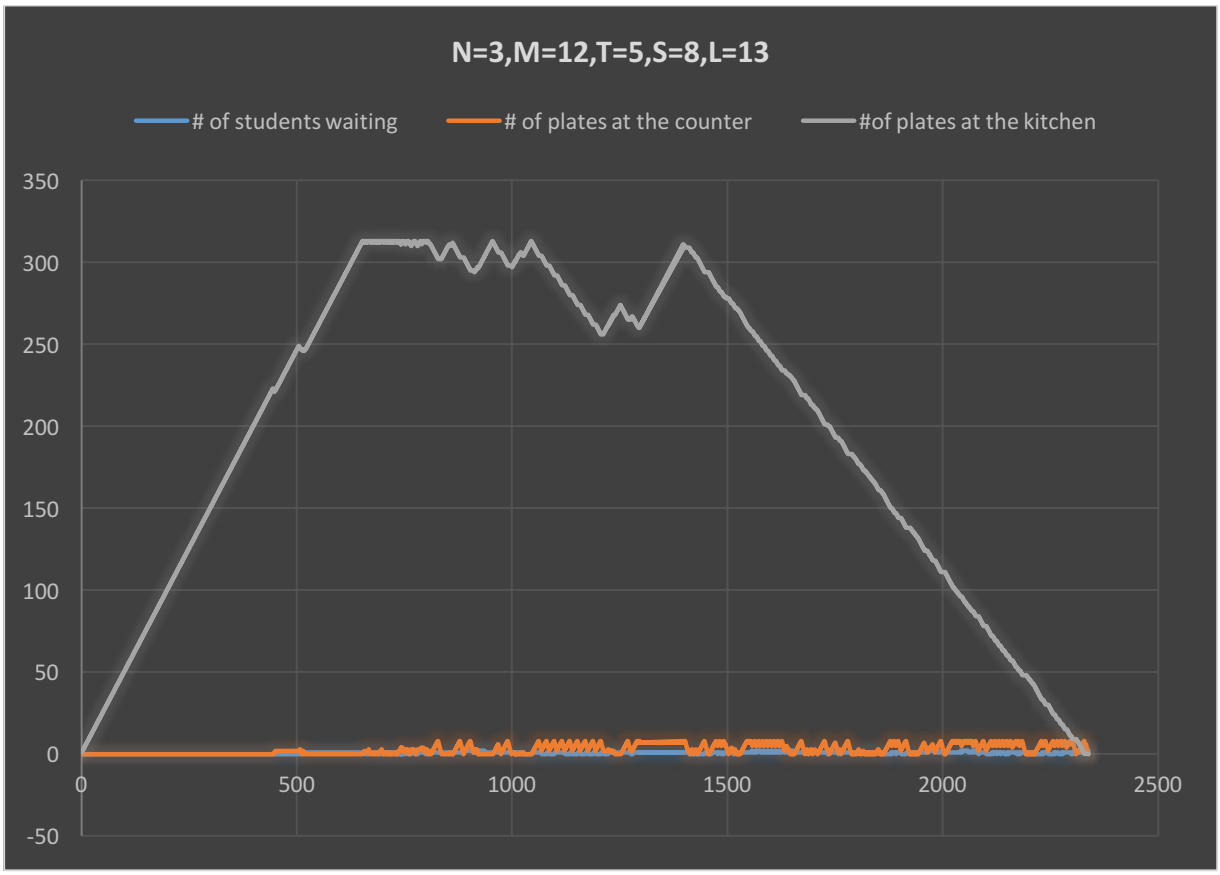

N=3,M=12,T=9,S=4,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500

S is variable and M,N,T,L are constant

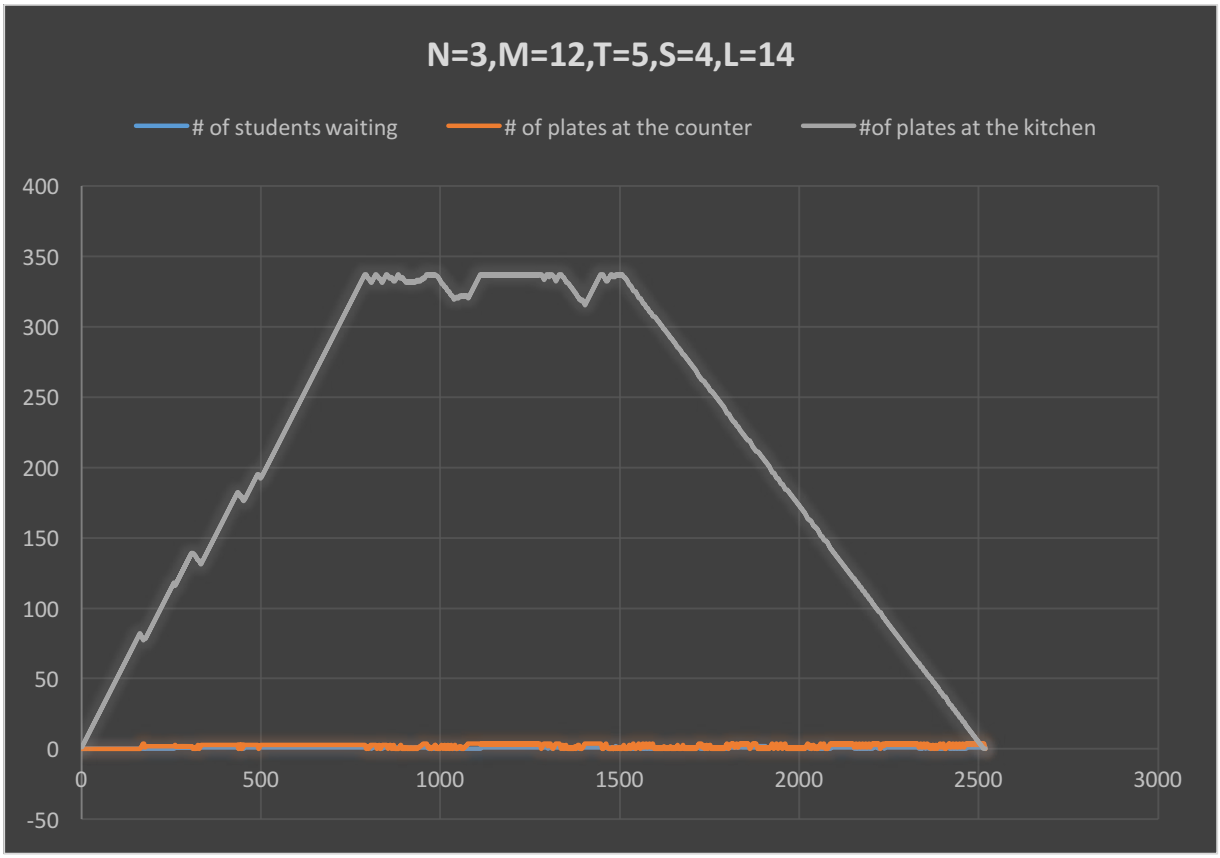


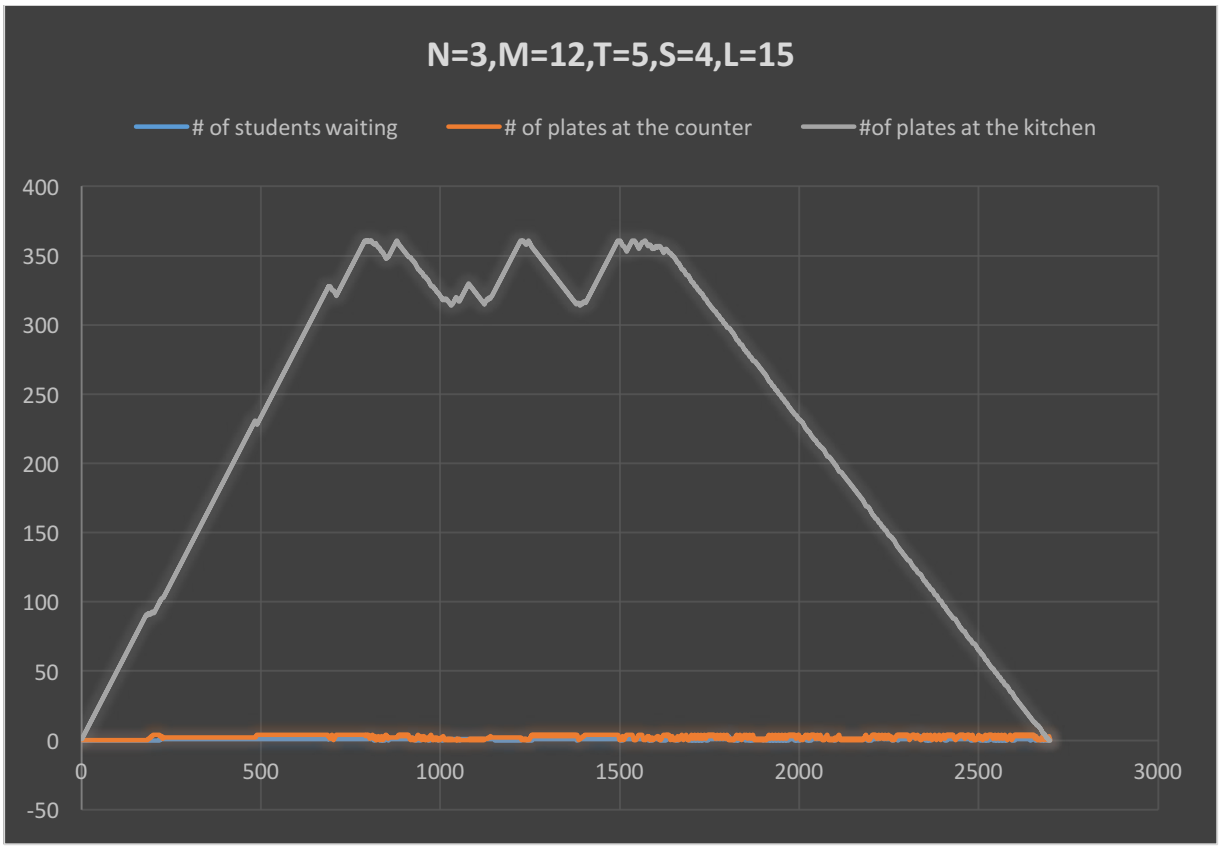


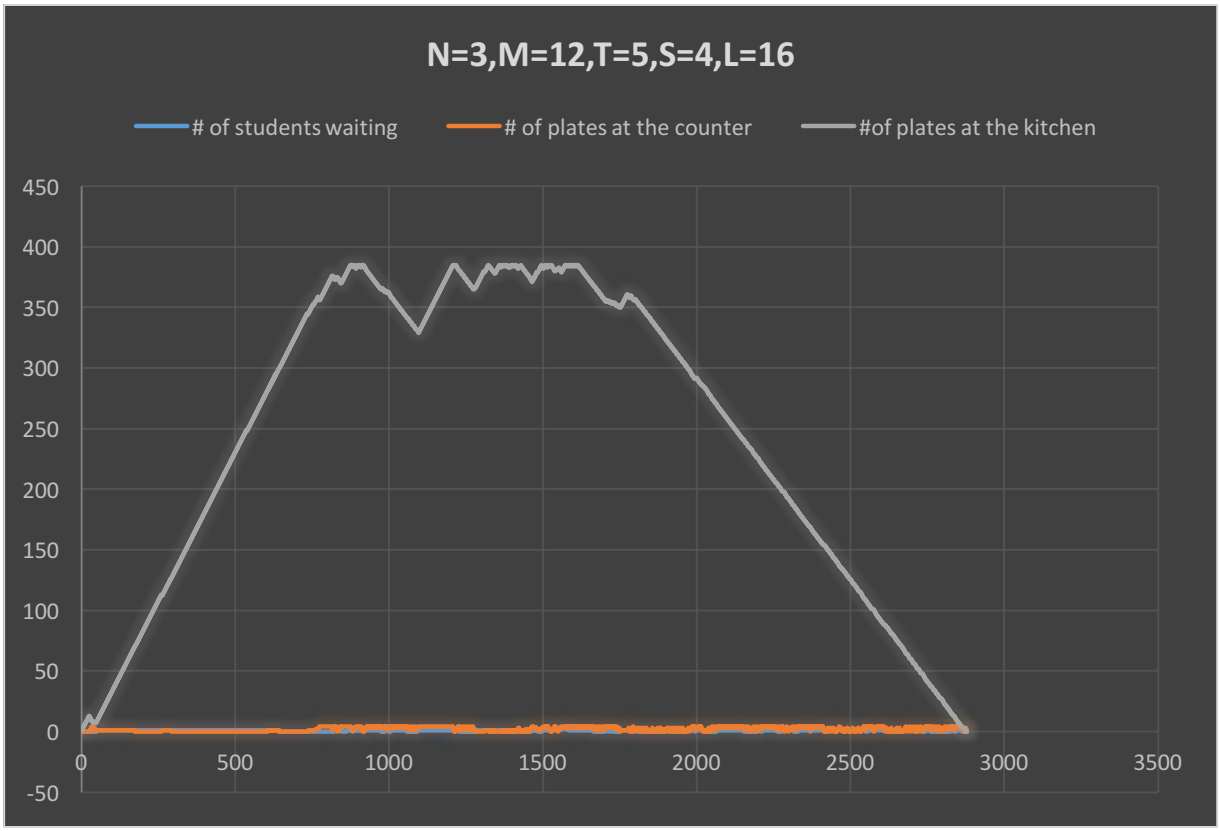

N=3,M=12,T=5,S=7,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500
3000

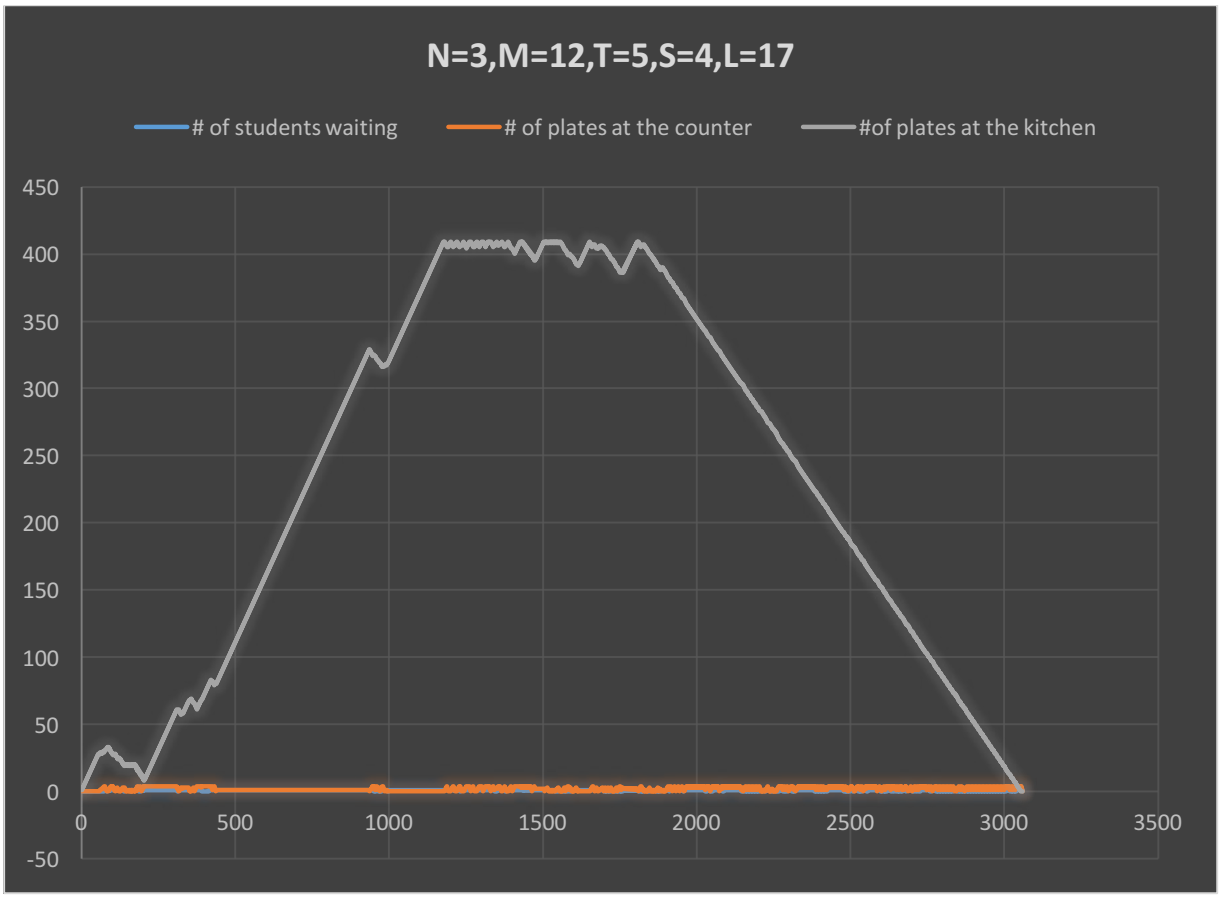

N=3,M=12,T=5,S=8,L=13
# of students waiting
# of plates at the counter
#of plates at the kitchen
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500

L is variable and M,N,T,S are constant






N=3,M=12,T=5,S=4,L=16
# of students waiting
# of plates at the counter
#of plates at the kitchen
450
400
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500
3000
3500


N=3,M=12,T=5,S=4,L=17
# of students waiting
# of plates at the counter
#of plates at the kitchen
450
400
350
300
250
200
150
100
50
0
-50
0
500
1000
1500
2000
2500
3000
3500