MOVING AVERAGE LAB REPORT

LAB #7

SECTION M

SUBMITTED BY:

SHOUNAK LAHIRI

SUBMISSION DATE:

11/2/2018

Problem

The Dual Shock 4 controller sends back actual real time data from different sensors in the controller, but the data that is received is rough, and not very "pretty" to look at. To make the data smoother and "pretty" to look at, implementing a moving average for each of the axis's gyroscopic values. The program should include a function that calculates the moving average of the gyroscopic values for each axis and another function which finds the running maximum and minimum values. Additionally, the program should stop running when the square button on the controller is pressed.

Analysis

When the program is run, the number of values that should be used to calculate the average is entered into the command line. The skeleton code for the program included the function prototypes, which included the average, the move function, and the min/max function. The average function finds the average of the elements inside the array, while using the number of items for calculating the average which was entered by the user in the command line. The move function moves the elements in an array, while adding the new gyroscopic values to the beginning of the array. The min and max function finds the running min and max from the arrays. The output should be comma separated and on one line so that it will be easy to create a graph in Excel or a similar program. Additionally, the program should stop when the square button is pressed.

Design

The problem presented in this lab can easily be broken into smaller pieces, which would easily fit into the provided functions

1. Calculating the Average
2. Updating the buffer (array)
3. Finding the Min/Max
4. Update

To find the average of the elements in the array at the current time, the function avg needs the array and the length of the number of elements that should be used, which was entered into the command prompt by the users. To calculate the array, a simple for loop that runs from the beginning of the array going length which was sent into the function. While iterating through the loop, the loop should add the value of the element to a running total. Once the loop finishes the running total should be divided by the length that the user entered, the result is returned to the function call.

For the updatebuffer function, the elements in the array need to be moved to the right by one, and the new gyroscopic value which is sent to the function needs to be added to the beginning of the array. First, to move everything to the right by one, another simple loop should be implemented which runs from zero to the length of the average values, sent to the function. While iterating through the loop, each element is moved to the right by one by adding one to the

index and setting the value of the old index equal to the new index. After finishing the loop, assigning the new gyroscopic value to the zeroth element in the array.

To find the min/max of the array using pointers, the array, length of array average, and the min and max pointers are sent to the function. By using an if statement, we were able to find the min and max of the array. Then, we were able to send the value back without using a return statement because the function uses pointers.

Outputting the information in the main was a little tricky because we needed to wait to fill the buffer. The best way to make the program run while the square button was not pressed was to put all the code in the main function inside in a while loop. Inside the while loop we placed the scan statement which assigned the new values to the variables. Then, calling the updatebuffer with the arrays for each axis, the length entered by the user, and the newly found gyroscopic value. Then to make sure the array has been filled with the proper length of elements, a counter was created which counted how many times the while loop iterated. Then an if statement made sure that the counter was greater than the length of the average, entered by the user. Once the if statement was entered, the average for each x, y, and z arrays were found by calling the avg function. Then, the min and max for each of the x, y, and z was found by calling the maxmin function. Then, the printf statement prints out the values for each axis's gyroscopic value, average, and min/max values all separated by only a comma.

Testing

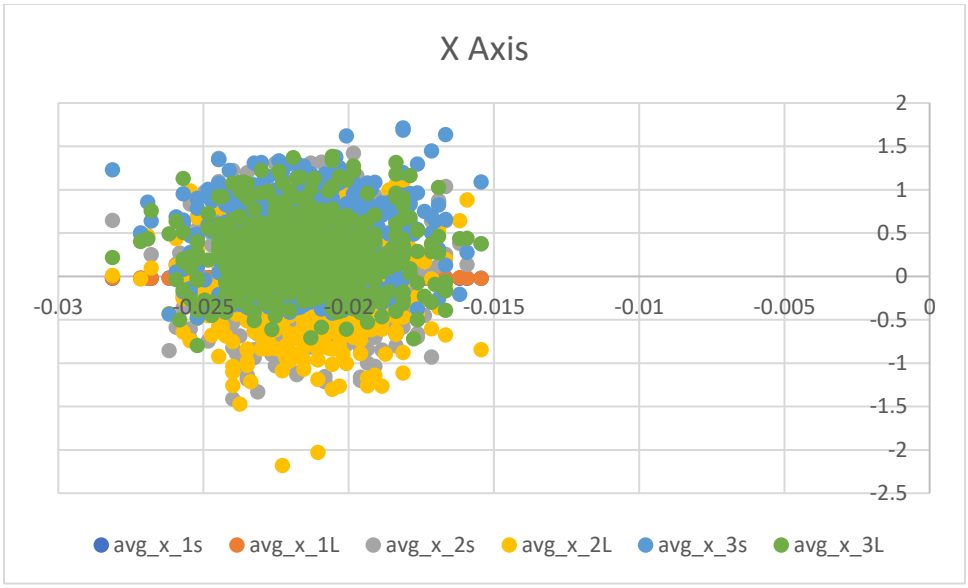
To test the program, we just looked at the output and checked to see whether or not the values made sense. After checking if the values made sense, we demoed the program for the TA's, who confirmed that the values that the program was outputting made sense.

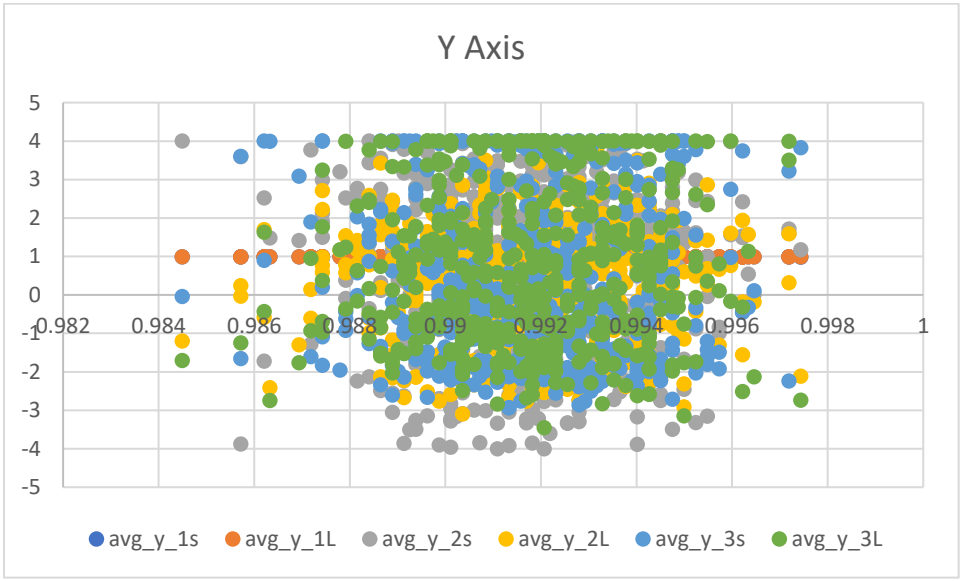
Comments

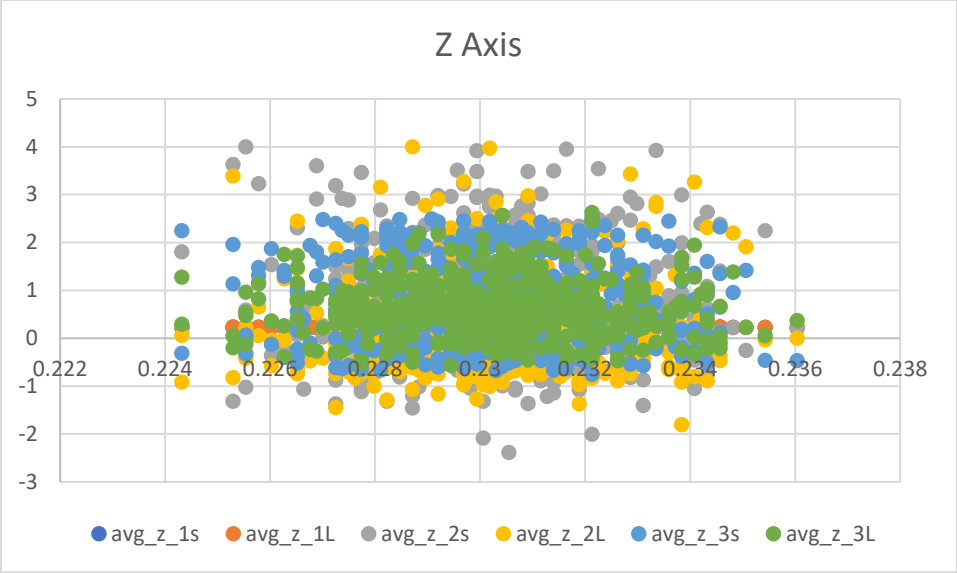
I think the design that was implemented was good because the program worked correctly and in a timely manner. I learned that things that you think will work do not always work. When writing the code for finding the average I thought that putting the value of the zeroth element in the array as the initial min and max then checking the other values in the array, would work, but it was outputting values that did not make sense.

Questions/Experiments

1)







2) The motion for motion 2 and motion 3 are very similar, which maybe due to the fact that the motions are very similar, motion 3 was preformed by moving the controller up and down vertically. Based on the results of the graph, using a long window provided a more condensed version of the data. Therefore, I would most likely use the larger window so the data would be more consistent and overall more accurate.

Source Code

```
// 185 Lab 7
#include <stdio.h>


#define MAXPOINTS 10000

// compute the average of the first num_items of buffer
double avg(double buffer[], int num_items);

//update the max and min of the first num_items of array
void maxmin(double array[], int num_items, double* max, double* min);

//shift length-1 elements of the buffer to the left and put the
//new_item on the right.
void updatebuffer(double buffer[], int length, double new_item);



int main(int argc, char* argv[]) {

	/* DO NOT CHANGE THIS PART OF THE CODE */

	double x[MAXPOINTS], y[MAXPOINTS], z[MAXPOINTS];
	int lengthofavg = 0;
	if (argc>1) {
		sscanf(argv[1], "%d", &lengthofavg );
		printf("You entered a buffer length of %d\n", lengthofavg);
	}
	else {
		printf("Enter a length on the command line\n");
		return -1;
	}
	if (lengthofavg <1 || lengthofavg >MAXPOINTS) {
		printf("Invalid length\n");
		return -1;
	}

	//My Variables
	int t, b1, b2, b3, b4, counter = 0;
    double ax, ay, az, gx, gy, gz;
	double max_x = 0, min_x = 0, max_y = 0, min_y = 0, max_z = 0, min_z =
0, avg_x = 0, avg_y = 0, avg_z = 0;

	while(b4 != 1)
	{
		//Get new values
```

```
            scanf("%d, %lf, %lf, %lf, %lf, %lf, %lf, %d, %d, %d, %d", &t,
&ax, &ay, &az, &gx, &gy, &gz, &b1, &b2, &b3, &b4 );

            //Fill the respective arrays with new values
            updatebuffer(x, lengthofavg , gx);
            updatebuffer(y, lengthofavg , gy);
            updatebuffer(z, lengthofavg , gz);

            if(counter > lengthofavg)
            {

                  //Calculate the average for each
                  avg_x = avg(x, lengthofavg);
                  avg_y = avg(y, lengthofavg);
                  avg_z = avg(z, lengthofavg);

                  //Find the Max/Min for each
                  maxmin(x, lengthofavg, &max_x, &min_x);
                  maxmin(y, lengthofavg, &max_y, &min_y);
                  maxmin(z, lengthofavg, &max_z, &min_z);

                  //Print the result for CSV file

      printf("%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf\n", gx, gy,
gz, avg_x, avg_y, avg_z, max_x, min_x, max_y, min_y, max_z, min_z);

            }

            counter++;
            fflush(stdout);
      }//End of while



}//End of main


void maxmin(double array[], int num_items, double* max, double* min)
{

      int i;
      //*max = array[0];
      //*min = array[0];
      for(i = 0; i < num_items; i++)
      {
            if(*max < array[i])
            {
                  *max = array[i];
```

```
            }
            if(*min > array[i])
            {
                  *min = array[i];
            }
      }

}//End of maxmin


void updatebuffer(double buffer[], int length, double new_item)
{

      int i;
      for(i = 0; i < length; i++)
      {
            buffer[i+1] = buffer[i];
      }
      buffer[0] = new_item;

}//End of updatebuffer


double avg(double buffer [], int num_items)
{

      int i;
      double sum = 0;
      for( i = 0; i < num_items; i++)
      {
            sum += buffer[i];
      }
      return sum/ num_items;

}//End of avg
```