MOVING AVERAGES

LAB 7 SECTION 1

SUBMITTED BY:

ETHAN GRIESMAN

SUBMISSION DATE:

10/26/2022

Problem:

The objective for this lab was to better familiarize ourselves with arrays in the C programming language and how they interact with real-time data from the DualShock 4 controller. We developed a program that reads from this data, and computes moving averages on the data in real time.

Analysis:

The problem states that we use actual characters to represent amounts of data and use those to create a repeating graphical representation of the DualShock data. So my program needs to take in the DualShock data, convert it to a useable form by using the a buffer to smooth out averages, analyze how many characters should be displayed on screen from the converted data, and then display them in real time.

Design

The problem the lab provides was to determine a moving average of large amounts of data which was completed in 4 main steps:

1) Creating an average calculator
2) Creating a maxmin calculator
3) Creating an update buffer function
4) Calling and printing these functions

The first step was accomplished by making a for loop that takes the buffer statement based on an [i] value then added onto the total sum and divided by the number of items being summed together. The second step was accomplished by creating a for loop to continue checking numbers as the data comes in and uses if statements to check if the value is larger or smaller than the prev_max or prev_min. The third step was accomplished by a for loop that decrements the value until such value is smaller than 0. The value of the buffer[0] is what allows the averages to actually be calculated. The fourth step was accomplished by writing the two while loops. One to find the used magnitudes and the next to find and print the moving averages. This was done by using simple scan statements used in previous labs and calling on the functions written.

Testing

To test this program, I collected data from all three motions into a spreadsheet so that we can use it later for graphical analysis. For each trial, we will calculate the average using a calculator and compare our answer with the result displayed in Cygwin. We also try our program with a running Dualshock controller and test to see if pressing the square button ends the program.

Demo'd to TA 10/19/2022

Comments

In this lab, I learned to use the pass by pointer and inserting commands from the command line to change a variable on the go. The only real struggle I personally had with this lab was in graphing the large values for the controller motions after writing the program. I wasn't as consistent in my time measurements for each motion, and tested for way too much time. If I could repeat this lab, I would have more accurately timed exactly each motion and repeated that same time for the subsequent jerking and slow motions after leaving the controller stationary.

Question #1:

The window length would affect the averages by having a larger set of numbers. The larger the set of numbers the average is taken from the less the average is going to change as viewed on the graph

Question #2:

The graphs smooth out when the buffer array is 100 instead of 20. The curve seems to start a little late into the graph in 100 window buffer compared to 20. This is possibly due to the averages which smooth out the graph to rationalize small changes in accelerometer values and to smooth out sudden spikes in values. Since the small bumps are smoother out in the 100 window buffer, I'd prefer to use that as the data is smoother and looks more neatly. However, I would still check the small window buffer for any large spikes in the data.

```
#include <stdio.h>


#define MAXPOINTS 10000

//9 graphs

int read_acc(double* a_x, double* a_y, double* a_z,  int* Button_UP,
int* Button_DOWN, int* Button_LEFT, int* Button_RIGHT,int* time);
// 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[]) {
        double ax, ay, az, avg_x, avg_y, avg_z, xmin, xmax, ymin, ymax,
zmin, zmax;

        //do not change
        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;
        }

        int time, Button_T, Button_X, Button_S, Button_C;
        int i;
        for(i=0; i < lengthofavg; i++)
        {
                        scanf("%d,%lf,%lf,%lf,%d,%d,%d,%d", &time, &ax,
&ay, &az, &Button_T, &Button_X, &Button_C, &Button_S);
                        x[i] = ax;
                        y[i] = ay;
                        z[i] = az;
        }


        while(Button_S!= 1) {
```

```
        for(i=0; i < lengthofavg; i++)
        {
                        printf("%lf, %lf, %lf\n", x[i], y[i], z[i]);
        }
        scanf("%d,%lf,%lf,%lf,%d,%d,%d,%d", &time, &ax, &ay, &az,
&Button_T, &Button_X, &Button_C, &Button_S);

        updatebuffer(x, lengthofavg, ax);
        maxmin(x, lengthofavg, &xmin, &xmax);
        avg_x = avg(x, lengthofavg);

        updatebuffer(y, lengthofavg, ay);
        maxmin(y, lengthofavg, &ymin, &ymax);
        avg_y = avg(y, lengthofavg);

        updatebuffer(z, lengthofavg, az);
        maxmin(z, lengthofavg, &zmin, &zmax);
        avg_z = avg(z, lengthofavg);

        printf("%lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf,
%lf\n", ax, ay, az, xmax, ymax, zmax, xmin, ymin, zmin, avg_x, avg_y,
avg_z);
        fflush(stdout);

        }
}


// compute the average of the first num_items of buffer
double avg(double buffer[], int num_items) {
        int i;
        double tot = 0;

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

//update the max and min of the first num_items of array
void maxmin(double array[], int num_items, double* min, double* max) {
        int i = 0;
        double prev_max=array[0];
        double prev_min=array[0];


        for (i = 0; i < num_items; i++) {

                if (array[i] > prev_max) {
                        prev_max=array[i];
                }
                if(array[i] < prev_min) {
                        prev_min=array[i];
                }
        }
        *max = prev_max;
```

```
        *min = prev_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 z;
        for(int z = 1; z < length; z++) {
                buffer[z] = buffer[z-1];
        }
        buffer[0] = new_item;

}
```