**Moving Average**

**LAB #8**

**SECTION AA**

**SUBMITTED BY:**

**ANDREW HANCOCK**

**SUBMISSION DATE:**

**12/4/2016**

**Lab 8 code**

#include <stdio.h>

#define MAXPOINTS 10000

// compute the average of the first num\_items of buffer

double avg(double buffer[], int num\_items){

double average = 0.0;

for(int i = 0;i < num\_items;i++){

average = (average + buffer[i]);

}

average = average/num\_items;

return average;

}

//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){

for(int i = 1;i <= length;i++){

buffer[i - 1] = buffer[i];

}

buffer[length] = new\_item;

}

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

double x[MAXPOINTS], y[MAXPOINTS], z[MAXPOINTS];

double ax, ay, az;

int BUTTON\_LEFT, BUTTON\_RIGHT, BUTTON\_UP, BUTTON\_DOWN, EXTRA\_1, EXTRA\_2;

double avgx = 0.0, avgy = 0.0, avgz = 0.0;

int lengthofavg = 0;

int i = 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;

}

do{

scanf("%lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &ax, &ay, &az, &BUTTON\_DOWN, &BUTTON\_UP, &BUTTON\_LEFT, &BUTTON\_RIGHT, &EXTRA\_1, &EXTRA\_2);

//printf("%lf, %lf, %lf, %d, %d, %d, %d, %d, %d\n", ax, ay, az, BUTTON\_DOWN, BUTTON\_UP, BUTTON\_LEFT, BUTTON\_RIGHT, EXTRA\_1, EXTRA\_2);

fflush(stdout);

x[i] = ax;

y[i] = ay;

z[i] = az;

if(i > lengthofavg - 1){

updatebuffer(x, lengthofavg, x[i]);

}

if(i > lengthofavg - 1){

updatebuffer(y, lengthofavg, y[i]);

}

if(i > lengthofavg - 1){

updatebuffer(z, lengthofavg, z[i]);

}

if(i > (lengthofavg - 2)){

avgx = avg(x, lengthofavg);

}

if(i > (lengthofavg - 2)){

avgy = avg(y, lengthofavg);

}

if(i > (lengthofavg - 2)){

avgz = avg(z, lengthofavg);

}

printf("%lf, %lf, %lf, %lf, %lf, %lf\n", ax, ay, az, avgx, avgy, avgz);

i++;

}while(1);

/\* Put your code here \*/

}

**Questions**

2. Since motion 3 moved less than motion two you could probably use a shorter window. Motion 2 was more rapid so in order to smooth it out a longer window would be better.

**Problem**

The program needs to be able to input the three accelerometer values from the Arduino Esplora and needs to calculate the average of the last n values for each time it scans in a new set of values. The program needs to be able to input a number from the user that will tell it how many values it needs to calculate the moving average for. The program will calculate a moving average for each of the three values.

**Analysis**

The skeleton code is provided and the function that are allowed to be used are already created. The inputs are the uses, because the command line will take the number of values it needs to calculate each moving average, and the Arduino Esplora, because the three accelerometer values will come from it. The output is in the console. The output will be the original values for each time data is scanned in and then the three moving average values for that loop of scanned in data. In order to calculate the moving average, the program will take the last n values and add them together and divide them by n. n is the value provided by the user.

**Design**

My program has two functions besides main. The first is avg. This takes in the buffer which is the last n values that it needs to calculate the average of, and it adds them all together then divides by n and returns the value. The next function is updatebuffer. This takes function moves every item in the buffer, which are the values that will have the average calculated, and moves them all back one in order to make space for the new data point. In main, the first thing that happens are the variable declarations. Next, the buffer length is read in from the command line. Next, the program enters into a do while loop. The first line scans in a new line of data from the Arduino Esplora accelerometer and buttons. Next, three if statements, one for each direction in the accelerometer, see if there have been enough data points to calculate the average. If there have, then the update buffer function is called. Next, three more if statements are called to calculate the average for each accelerometer value by calling the avg function. Lastly, everything is printed on one line, separated by commas and i, the variable keeping track of how many lines of Arduino data have been scanned in increments.

**Testing**

Since it was hard to tell if the moving average was being calculated with a small buffer value, I usually always used a big one, like 1000 for example. With a value of that size the values changed more evenly and the moving average is more obviously calculated. I think the easiest way to see it, though, was with the excel graphs. Mainly for testing, I just ran it a bunch with a high buffer value and made sure it was calculating the moving average.

**Comments**

I think my program had a solid design and I had fun learning how to calculate the moving average. Looking back on my code as I write this lab report, I see some things that I could write better than they are currently written. For example, I didn’t need to write three separate if statements for the updatebuffer function calling and the avg function calling.