High Performance Computer
Rovira e Virgili-UOC

Student: Felix

1. **Describe how the code works, i.e., how does it get to write "Hello World" from the input vectors and the provided kernel.**

1. Program header <stdio.h> for c
1.1 Declare variables N and blocksize
1.2 Declare the global specifier and function hello run on device(GPU)
2. Main
2.1 Declare variables as array a[N],b[N]
2.2 Declare pointer *ad and *bd as pointers and csize, isize as constant
2.3 Print the string Hello from input vector a[N] "Hello\0\0\0\0\0\0" then pass the string to CUDA with an array of offsets. Knowledge of pointer and memory addresses must be observed
2.4 Dynamically allocate memory space for device copies of a of size csize
2.5 Dynamically allocate memory space for device copies of b of size isize
2.6 Copy a memory of size csize from a to ad direction host to device
2.7 Copy a memory of size isize from b to bd direction host to device
2.8 Declare integer vector type dim3 and number of parallel treads per block =16
2.9 Declare integer vector type dim3 and number of threads block dimgrid(1, 1) =1
2.10 Execute kernel on device hello<<<dimgrid, dimblock>>>ad, bd here the program pass the array elements of vector a and vector b, and add them using the built in threadIdx variable as index . creating offsets which are added in parallel to produce the string World!. Important to notice the use of **ASCII character code** to create the string **World!**
2.11 Copy a memory of size csize from ad to a direction device to host back to CPU
2.12 Free dynamically allocated device memory
2.13 Print Hello World!

2. What memory transfer is produced between the host and the device (GPU)?

Answer: Transferring data between host and device memory can be done through **cudaMencpy.**

• Function copy a memory of csize from a to ad (*d standard notation for device*)
`cudaMemcpyHostToDevice` indicate the direction for typical usage
• Function copy a memory of isize from b to bd (*d standard notation for device*)
`cudaMemcpyHostToDevice` indicate the direction for typical usage
• Function copy a memory of csize from ad to a (*d standard notation for device*)
cudaMemcpyDeviceToHost indicate the direction for typical usage.

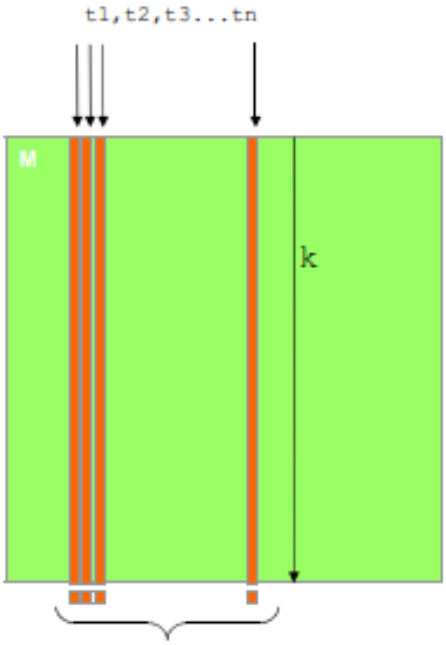
3. Provide your implementation using CUDA.
Attached moving_average.cu

Provide you implementation.

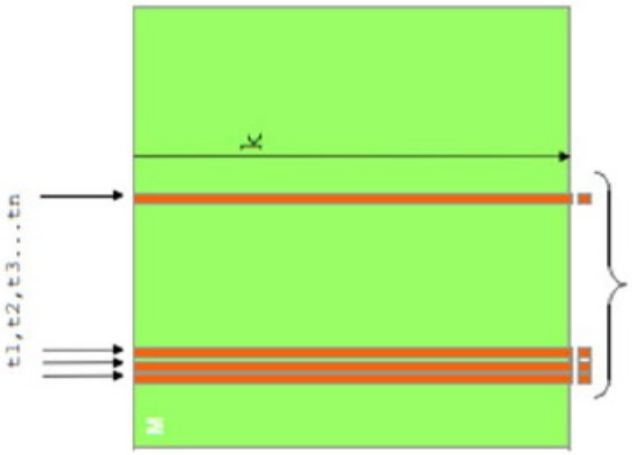
• I run the program with Nvidia tool kit plug in 10.2
Steps to make the program code. Here basically I have the next blocks :
1.1 Create two global functions one for each vector vectoravg and vectormov
1.2 Generate a random matrix to avoid be inputting data.
1.3 Call global function
1.4 Send the results to a excel file to plot

Critical Thinking
• Each thread calculate the mov of one rows matrix and we are going to get a column vector.
• Each thread calculate the avg in descent way of one column matrix and we are going to get a row vector.

Here we get row vector



Here get a column vector



```
//Felix feliu exercise 3 cuda using visual studio
/*C/Cuda program to accept a matrix of order M x N and find the movil average and average
of each row and each column of a matrix*/

#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <cuda.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <memory>

#define SIZE 1024

//function 1 vectoravg_c column average
__global__ void vectoravg_c(int m, int n, int* data_d, float* avg_d)
{
	const int avg_index = (blockIdx.y * gridDim.x + blockIdx.x) * blockDim.x +
threadIdx.x;

	if (avg_index >= n) {
		return;
	}

	float sum = 0;
	for (int i = 0; i < m; ++i)
	{
		sum = sum + data_d[i * n + avg_index];
	}
	avg_d[avg_index] = (sum / m);
}


//function 2 vectormov decrement loop
__global__ void vectormov(int m, int n, int* data_d, float* mov_d)
{
	const int mov_index = (blockIdx.y * gridDim.x + blockIdx.x) * blockDim.x +
threadIdx.x;

	if (mov_index >= m * n) {
		return;
	}
	int index = mov_index;
	float sum = data_d[index];
	int count = 1;
	while (index % n != 0 && count < 9)
	{
		count++;
		index--;
		sum = sum + data_d[index];
	}
	mov_d[mov_index] = (sum / count);

}
```

Global functions

```
int main()
{

	//Secuential part

	// initialize variables

	int i, j, m, n;
	int* data = (int*)malloc(1000 * 1000 * sizeof(int));
	float* mov = (float*)malloc(1000 * 1000 * sizeof(float));
	float* avg_c = (float*)malloc(1000 * sizeof(float));
	memset(data, 0, 1000 * 1000 * sizeof(int));
	memset(mov, 0, 1000 * 1000 * sizeof(float));
	memset(avg_c, 0, 1000 * sizeof(float));


	printf("Enter the order of the matrix\n");
	scanf_s("%d %d", &m, &n);



	srand(time(0)); //clock to generate random numbers
	for (i = 0; i < m; ++i)
	{
		for (j = 0; j < n; ++j)
		{
			int index = i * n + j;
			data[index] = rand() % 100;
			//printf("%d ", data[index]);
		}

		//printf("\n");
	}
```

Generate Random Numbers

```
	// CUDA part

	//initialized variables
	int* array_d;
	float* mov_d, * avg_d;

	int size = m * n * sizeof(int);
	int size1 = m * n * sizeof(float);
	int size2 = n * sizeof(float);

	// Allocate memory on device
	cudaMalloc(&array_d, size);
	cudaMalloc(&mov_d, size1);
	cudaMalloc(&avg_d, size2);

	// initilized matrix on device
	cudaMemcpy(array_d, data, size, cudaMemcpyHostToDevice);
	cudaMemset(mov_d, 0, size1);
	cudaMemset(avg_d, 0, size2);
```

Cuda Code

Cuda Code

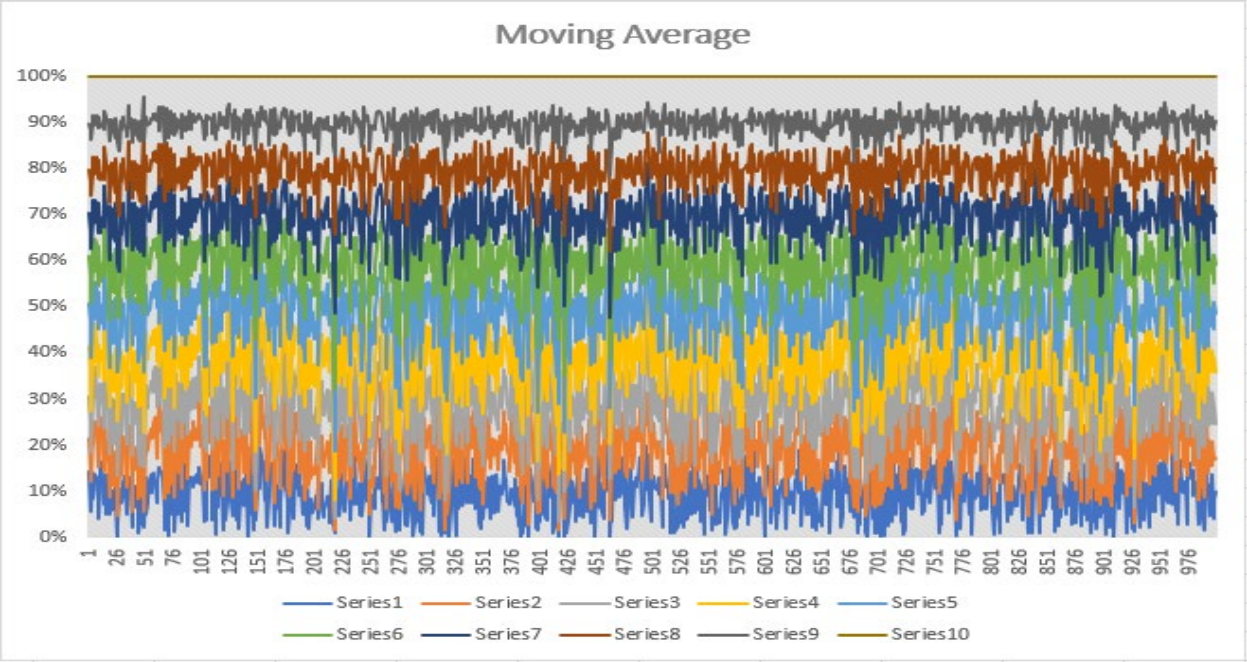
```
// call kernel function 1 vectoravg increment loop
	int bloques = (n + 128 - 1) / 128;
	int y_bloques;
	int x_bloques;

	if (bloques > 128) {
		y_bloques = (bloques + 128 - 1) / 128;
		x_bloques = 128;
	}
	else {
		y_bloques = 1;
		x_bloques = 128;

	}
	dim3 tamGrid1(x_bloques, y_bloques); //grid dimensión
	dim3 tamBlock1(128, 1, 1); //glock dimensión

                                                    // launch the device computation
threads!
	vectoravg_c << <tamGrid1, tamBlock1 >> > (m, n, array_d, avg_d);
	cudaDeviceSynchronize();

	// call kernel function 2 vectormov decrement loop
	bloques = (m * n + 128 - 1) / 128;
	if (bloques > 128) {
		y_bloques = (bloques + 128 - 1) / 128;
		x_bloques = 128;
	}
	else {
		y_bloques = 1;
		x_bloques = 128;

	}
	dim3 tamGrid2(x_bloques, y_bloques); //grid dimensión
	dim3 tamBlock2(128, 1, 1); //block dimensión

	// Launch the device computation threads!
	vectormov << <tamGrid2, tamBlock2 >> > (m, n, array_d, mov_d);

	cudaDeviceSynchronize();

	// bring results;
	cudaMemcpy(avg_c, avg_d, size2, cudaMemcpyDeviceToHost);
	cudaMemcpy(mov, mov_d, size1, cudaMemcpyDeviceToHost);

	// free memory in  device
	cudaFree(avg_d);
	cudaFree(mov_d);
	cudaFree(array_d);
```

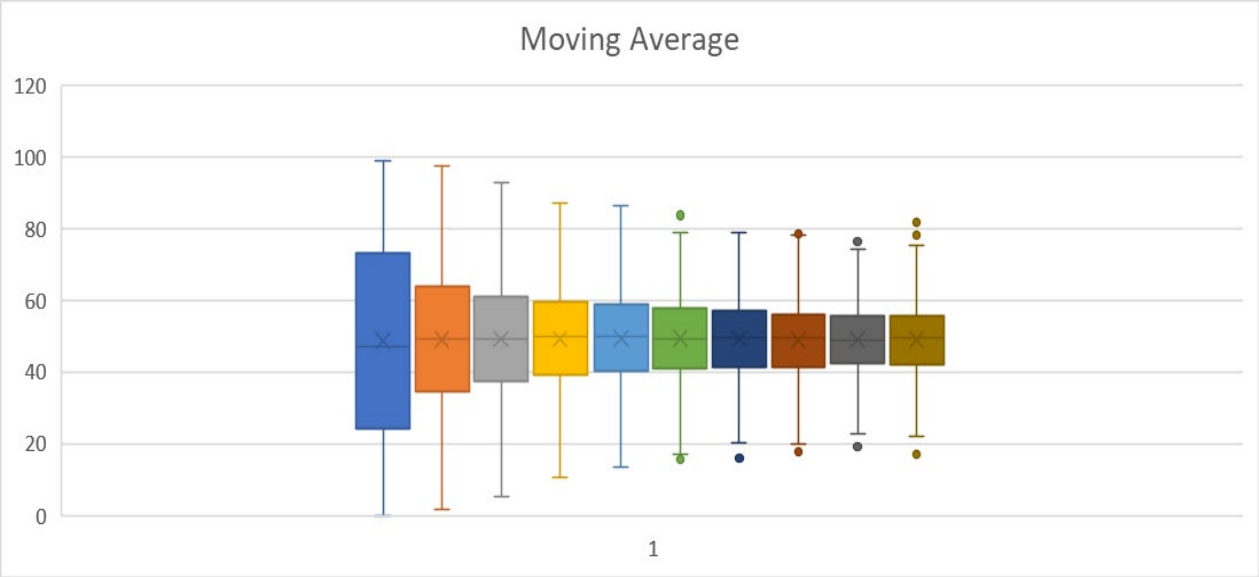
Send Matrix To csv file And plot

```
	//	print mov matrix
	FILE* file;
	fopen_s(&file, "moving average.csv", "w"); //if does not  work, try
file=fopen("vectors.csv", "w");

	fprintf(file, "MOV[M,N] : \n");
	for (i = 0; i < m; ++i)
	{
		for (j = 0; j < n; ++j)
		{
			int index = i * n + j;
			if (j != n - 1)
				fprintf(file, "%.2f,", mov[index]);
			else
				fprintf(file, "%.2f", mov[index]);
		}

		fprintf(file, "\n");
	}

	//	print mov matrix
	fprintf(file, "AVG[N] : \n");
	for (j = 0; j < n; ++j)
	{
		if (j != n - 1)
			fprintf(file, "%.2f,", avg_c[j]);
		else
			fprintf(file, "%.2f", avg_c[j]);
	}

	fclose(file);
	char t[10];
	printf("enter an string");
	scanf_s("%s", t);


	return 0;
}
```

4. Provide an example use of your code. For example, you can generate an input matrix of small size (1000x10) with random values (or following a given distribution) in such a way that you can generate a plot from the input data (plot with 10 series of 1000 points), and the matrix and output vector.

Attached: **Attached moving_average.xlsx**





Using CUDA with Data from Cyberinfrastructure

5. Use your CUDA implementation with one of the provided datasets (from the previous list such as "water pressure") to illustrate the effect of the "moving average" on the data when they are visualized. You can use different window sizes (e.g., 10, 15 and 20 data points for implementing the "moving average").

Attached: mov_avg_ciberinfrastructure.cu
Excel files: 194m_pres_mov_avg_cyber_10.xlsx
581m_temp_mov_avg_cyber_15.xlsx
1542m_pres_mov_avg_cyber_20.xlsx
2903m_temp_mov_avg_cyber_25.xlsx

Fig 1

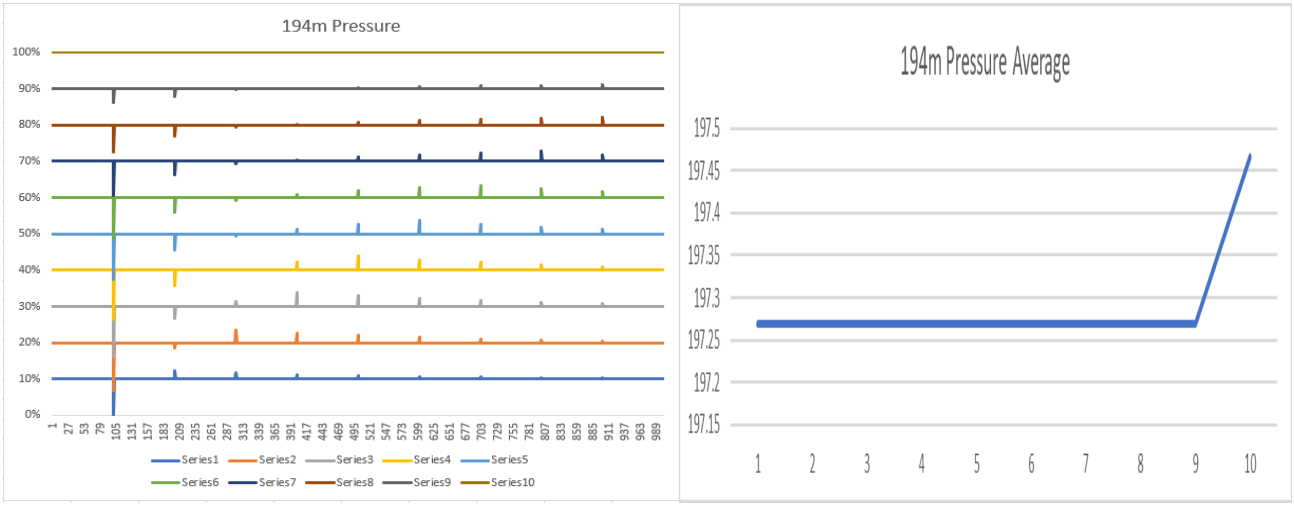


Fig 2

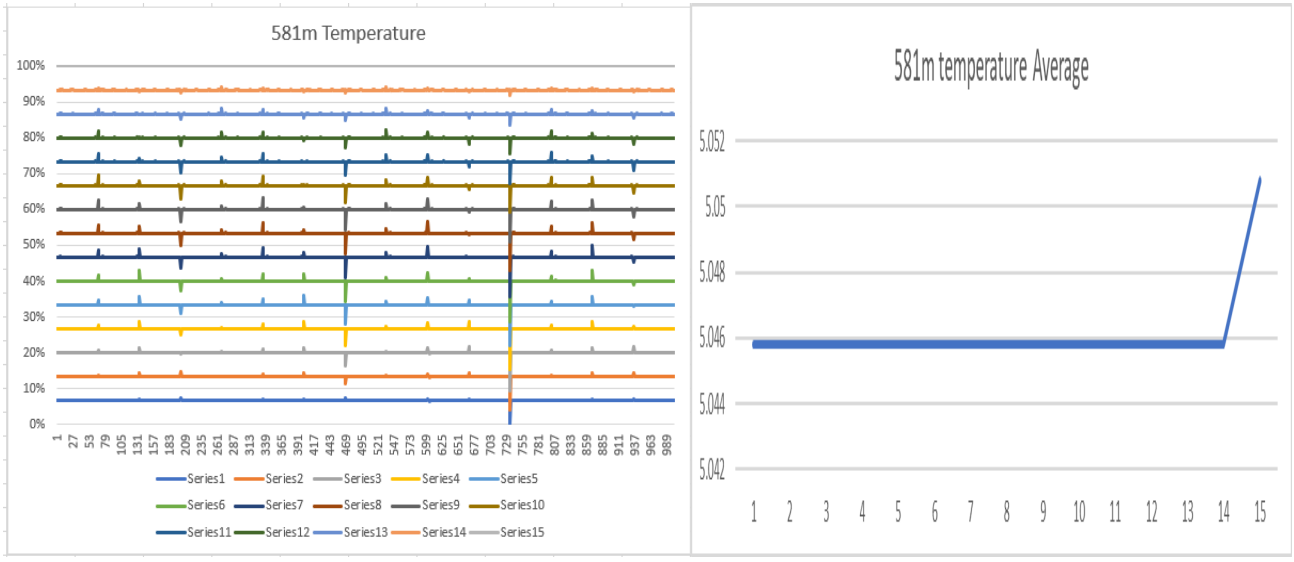


Fig 3

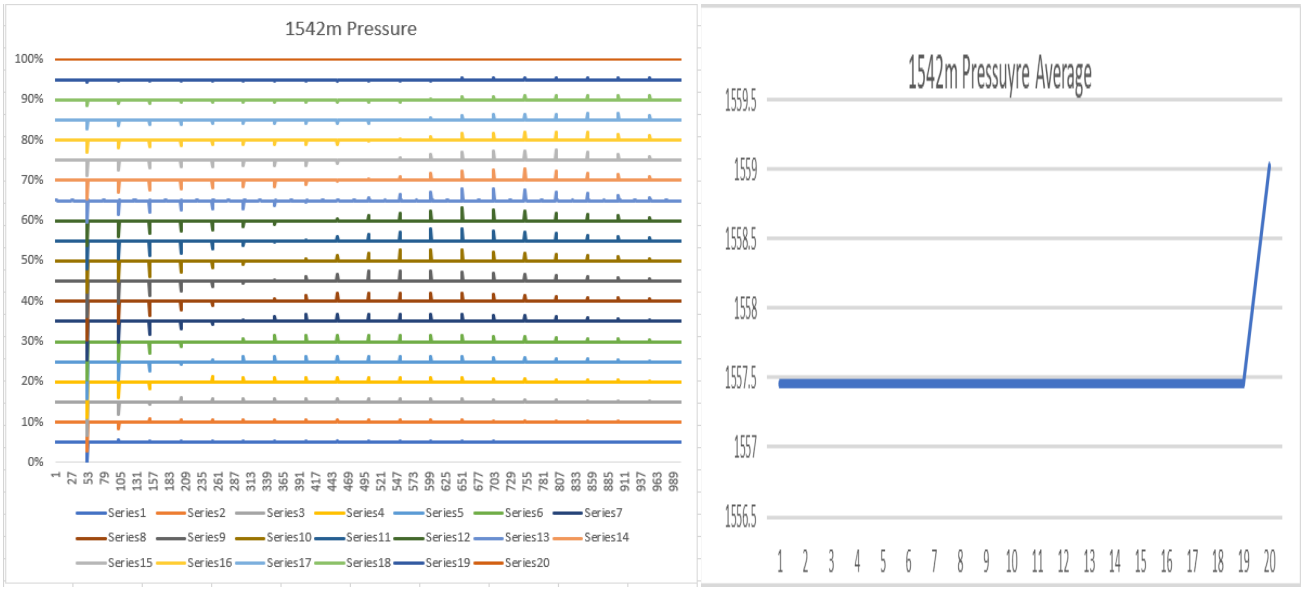
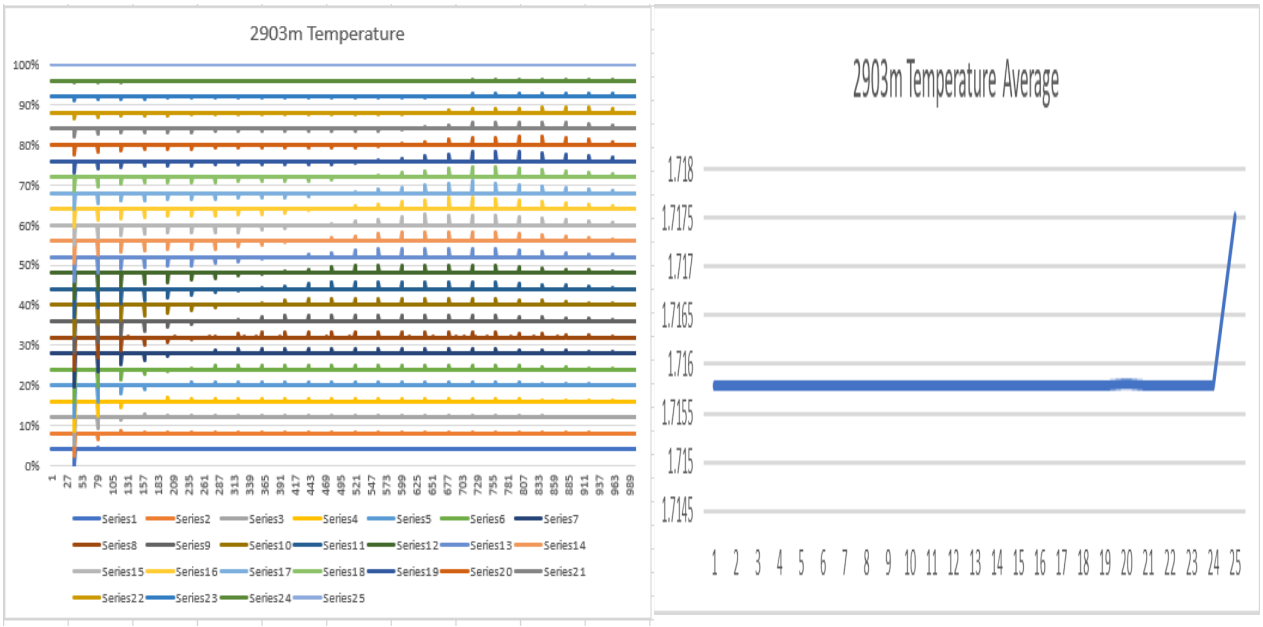

1542m Pressure
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Series1 Series2 Series3 Series4 Series5 Series6 Series7
Series8 Series9 Series10 Series11 Series12 Series13 Series14
Series15 Series16 Series17 Series18 Series19 Series20
1542m Pressuyre Average
1559.5
1559
1558.5
1558
1557.5
1557
1556.5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

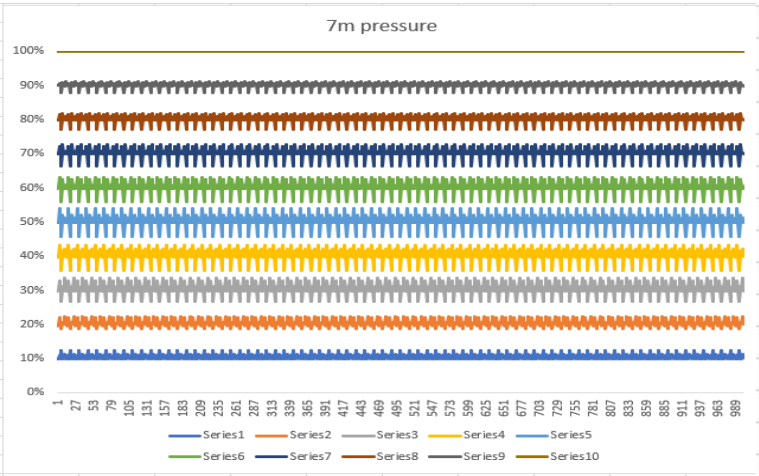
Fig 4

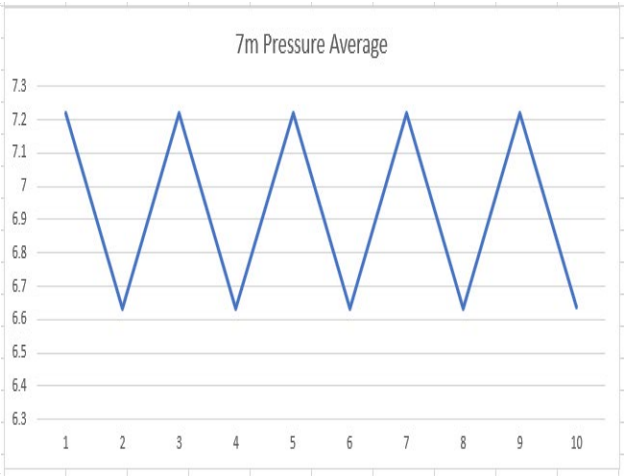

2903m Temperature
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Series1 Series2 Series3 Series4 Series5 Series6 Series7
Series8 Series9 Series10 Series11 Series12 Series13 Series14
Series15 Series16 Series17 Series18 Series19 Series20 Series21
Series22 Series23 Series24 Series25
2903m Temperature Average
1.718
1.7175
1.717
1.7165
1.716
1.7155
1.715
1.7145
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Optional 1. Explore the OOI data portal and OOI website (oceanobservatories.org) to obtain other data sets and analyze them with your CUDA code.
See **mov_avg_ciberinfrastructure.cu**
Attached: 7m_pres_mov_avg_cyber_10.xlsx
7m_temp_mov_avg_cyber_10.xlsx

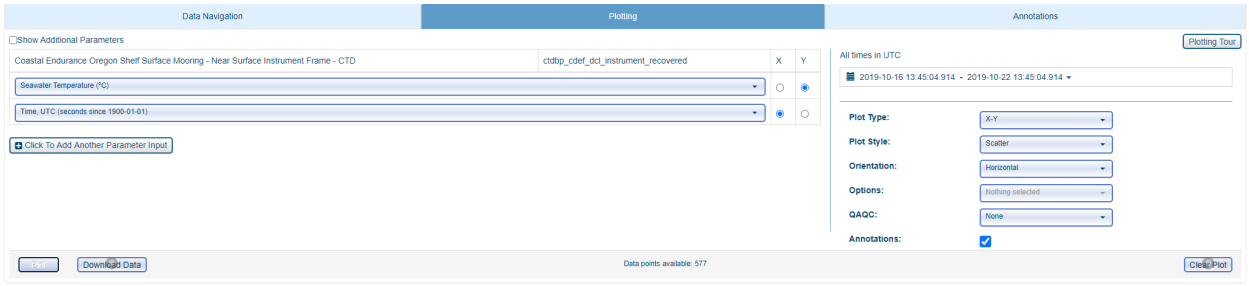
Pressure

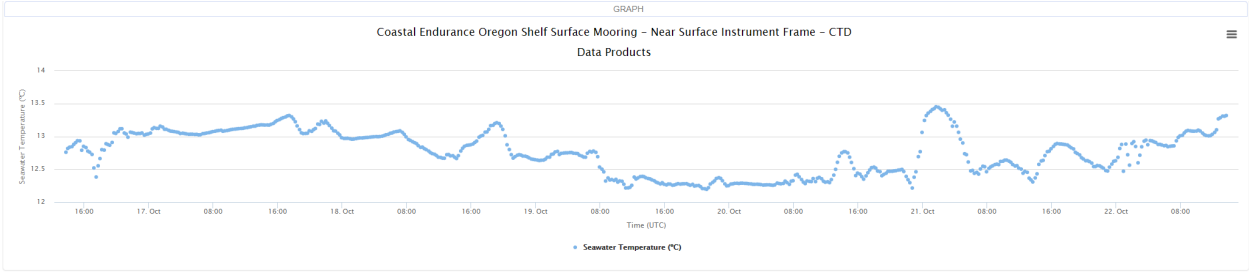
Cuda code





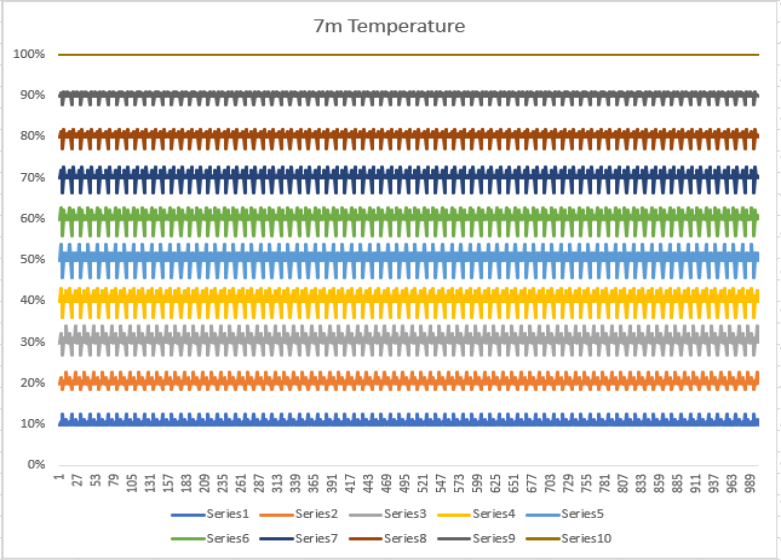
OOI Portal

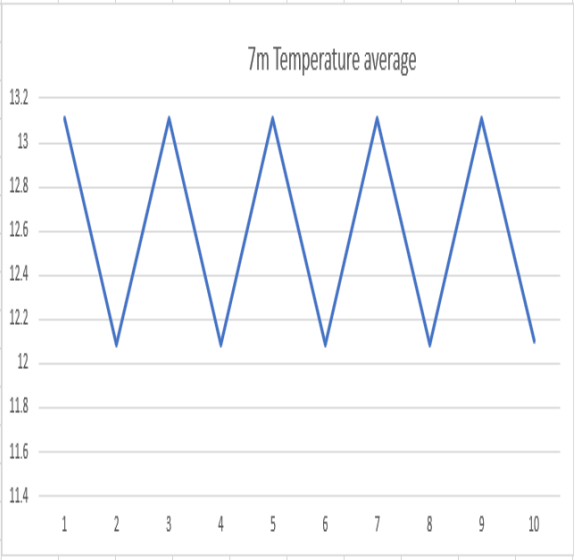




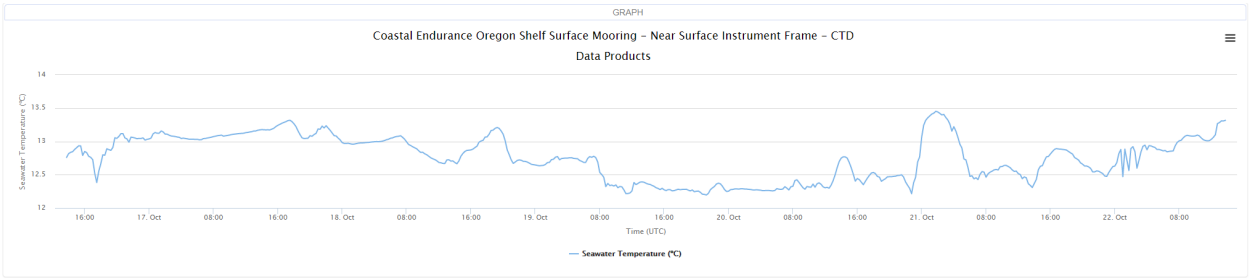
Temperature

Cuda code





OOI Portal



Observation: temperature and pressure look constant at this interval time with not a considerable amount of noise incidence. I think is due to the measures being done at the surface.

Optional 2. Respond the survey shown below

Please rate (1-5) the following questions in a table as shown below.

1. How much did you know about CI before this course?
2. How much do you know about CI after introducing this topic in this course?
3. Would you like to know more about CI?
4. Do you find useful to use CI in the context of this course?
5. Do you think adding more CI-related topics and activities would improve this course?
6. Would you like to have an assignment/assignment using HPC/CI-related problems (e.g., MPI using python)?
7. Do you have experience using python, data and scientific libraries (e.g., pandas, matplotlib, etc.), GitHub, Jupyter Notebooks, etc.?
8. How positively (5) or negatively (1) do you think including more CI-related topics and Related tools (items listed in the previous question) would improve your skills and/or Professional competences?
9. How useful do you think including CI-related topics in the textbook materials would be?
10. How useful do you think including formal or informal videos including demos/tutorials And use case scenarios would be in this course

Question #	1	2	3	4	5	6	7	8	9	10
Rating (1-5)	2	4	5	5	4	4	3	4	4	5