

# Programming Assignment IV: CUDA Programming

The purpose of this assignment is to familiarize yourself with CUDA programming.

## 1 Problem Statement

In this problem, you need to use CUDA to parallelize concurrent wave equation ([http://en.wikipedia.org/wiki/Wave\\_equation](http://en.wikipedia.org/wiki/Wave_equation)). Below show a serial implementation of the concurrent wave equation ([http://www.cs.nctu.edu.tw/~ypyou/courses/PP-s19/assignments/HW4/serial\\_wave.c](http://www.cs.nctu.edu.tw/~ypyou/courses/PP-s19/assignments/HW4/serial_wave.c)).

```
/* *****  
 * DESCRIPTION:  
 *   Serial Concurrent Wave Equation - C Version  
 *   This program implements the concurrent wave equation  
 * ***** */  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <time.h>  
  
#define MAXPOINTS 1000000  
#define MAXSTEPS 1000000  
#define MINPOINTS 20  
#define PI 3.14159265  
  
void check_param(void);  
void init_line(void);  
void update (void);  
void printfinal (void);  
  
int nsteps,                /* number of time steps */  
    tpoints,               /* total points along string */  
    rcode;                /* generic return code */  
float values[MAXPOINTS+2], /* values at time t */  
      oldval[MAXPOINTS+2], /* values at time (t-dt) */  
      newval[MAXPOINTS+2]; /* values at time (t+dt) */  
  
/* *****  
 *   Checks input values from parameters  
 * ***** */  
void check_param(void)  
{  
    char tchar[20];  
  
    /* check number of points, number of iterations */  
    while ((tpoints < MINPOINTS) || (tpoints > MAXPOINTS)) {  
        printf("Enter number of points along vibrating string [%d-%d]: ",  
              MINPOINTS, MAXPOINTS);  
        scanf("%s", tchar);  
        tpoints = atoi(tchar);  
        if ((tpoints < MINPOINTS) || (tpoints > MAXPOINTS))  
            printf("Invalid. Please enter value between %d and %d\n",  
                  MINPOINTS, MAXPOINTS);  
    }  
    while ((nsteps < 1) || (nsteps > MAXSTEPS)) {  
        printf("Enter number of time steps [1-%d]: ", MAXSTEPS);  
        scanf("%s", tchar);
```

```

        nsteps = atoi(tchar);
        if ((nsteps < 1) || (nsteps > MAXSTEPS))
            printf("Invalid. Please enter value between 1 and %d\n",
                MAXSTEPS);
    }

    printf("Using points = %d, steps = %d\n", tpoints, nsteps);
}

/*****
 *      Initialize points on line
 *****/
void init_line(void)
{
    int i, j;
    float x, fac, k, tmp;

    /* Calculate initial values based on sine curve */
    fac = 2.0 * PI;
    k = 0.0;
    tmp = tpoints - 1;
    for (j = 1; j <= tpoints; j++) {
        x = k/tmp;
        values[j] = sin (fac * x);
        k = k + 1.0;
    }

    /* Initialize old values array */
    for (i = 1; i <= tpoints; i++)
        oldval[i] = values[i];
}

/*****
 *      Calculate new values using wave equation
 *****/
void do_math(int i)
{
    float dtime, c, dx, tau, sqtau;

    dtime = 0.3;
    c = 1.0;
    dx = 1.0;
    tau = (c * dtime / dx);
    sqtau = tau * tau;
    newval[i] = (2.0 * values[i]) - oldval[i] + (sqtau * (-2.0)*values[
        i]);
}

/*****
 *      Update all values along line a specified number of times
 *****/
void update()
{
    int i, j;

    /* Update values for each time step */
    for (i = 1; i<= nsteps; i++) {

```

```

    /* Update points along line for this time step */
    for (j = 1; j <= tpoints; j++) {
        /* global endpoints */
        if ((j == 1) || (j == tpoints))
            newval[j] = 0.0;
        else
            do_math(j);
    }

    /* Update old values with new values */
    for (j = 1; j <= tpoints; j++) {
        oldval[j] = values[j];
        values[j] = newval[j];
    }
}

/*****
 *      Print final results
 *****/
void printfinal()
{
    int i;

    for (i = 1; i <= tpoints; i++) {
        printf("%6.4f ", values[i]);
        if (i%10 == 0)
            printf("\n");
    }
}

/*****
 *      Main program
 *****/
int main(int argc, char *argv[])
{
    sscanf(argv[1], "%d", &tpoints);
    sscanf(argv[2], "%d", &nsteps);
    check_param();
    printf("Initializing points on the line...\n");
    init_line();
    printf("Updating all points for all time steps...\n");
    update();
    printf("Printing final results...\n");
    printfinal();
    printf("\nDone.\n\n");

    return 0;
}

```

## 2 Requirements

- Your parallelized version of serial\_wave.c should be named `cuda_wave.cu`.
- Your program should take two command-line arguments, which indicate the number of points and the number of iterations, respectively.
- The output format should not be changed.

## 3 Development Environment

### 3.1 Building the CUDA environment on your own computer

If you have an nVIDIA GPU, you can build your own development environment by installing CUDA SDK.

<https://developer.nvidia.com/cuda-downloads>

### 3.2 Using NCTU CS CUDA Server

We have set up four servers for this assignment. You can login to one of the servers to work on your assignment. Each server contains a GeForce GTX 1060 GPU. TAs will grade your implementation on these servers, so please make sure your implementation works on the provided servers.

#### 3.2.1 SSH Login Information

Server IP: 140.113.215.195

Port: 37106, 37107, 37108, 37109

Account: pp<student ID>

Default Password: pp<student ID>

#### 3.2.2 Compilation

You can use `gcc` to compile `serial_wave.c`

```
gcc serial_wave.c -o serial_wave -lm
```

Use `nvcc` to compile `cuda_wave.cu`.

```
nvcc cuda_wave.cu -o cuda_wave
```

## 4 Submission

Be sure to upload your zipped source codes, which includes no folder, to e-Campus system by the due date and name your file as "HW4\_XXXXXXX.zip", where XXXXXXX is your student ID.

**Due Date: 23:59, May 20, Monday, 2019**

## 5 References

- <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>