**CME213, Spring 2022**
**Parallel Computing**
**Julian Cooper**

## Homework 4: CUDA GPU Matrix Operations

Goal of the homework is to implement a finite difference solver for the 2-dim heat equation using CUDA GPU programming.

**Problem 1: Implement gpuStencil Global** Idea: parallelize spatial dimension updates for each time step iteration. One key challenge was to handle the different border sizes for order inputs of 2, 4 and 8. Kernel logic included below.

```
/**
 * Kernel to propagate finite difference grid from the current
 * time point to the next.
 *
 * This kernel should be very simple and only use global memory
 * and 1d threads and blocks.
 *
 * @param next[out] Next grid state.
 * @param curr Current grid state.
 * @param gx Number of grid points in the x dimension.
 * @param nx Number of grid points in the x dimension to which the full
 *           stencil can be applied (ie the number of points that are at least
 *           order/2 grid points away from the boundary).
 * @param ny Number of grid points in the y dimension to which th full
 *           stencil can be applied.
 * @param xcfl Courant number for x dimension.
 * @param ycfl Courant number for y dimension.
 */
template<int order>
__global__
void gpuStencilGlobal(float* next, const float* __restrict__ curr,
                      int gx, int nx, int ny, float xcfl, float ycfl) {

    int borderSize = (int) (order / 2);
    int i = blockIdx.x * blockDim.x + threadIdx.x;

    if (i < nx*ny) {
  int x = borderSize + (int) (i / nx);
  int y = borderSize + (i % nx);
        int idx = gx * y + x;
        next[idx] = Stencil<order>(&curr[idx], gx, xcfl, ycfl);

    }
}
```

3D surface plots of temperature on 256 x 256 grid at iterations 0, 1000 and 2000 respectively, with 8th order. To do this, I used parameter settings of: order = 8 and nx = ny = 248.

```
// Allocate num_bytes of memory to the device arrays
```

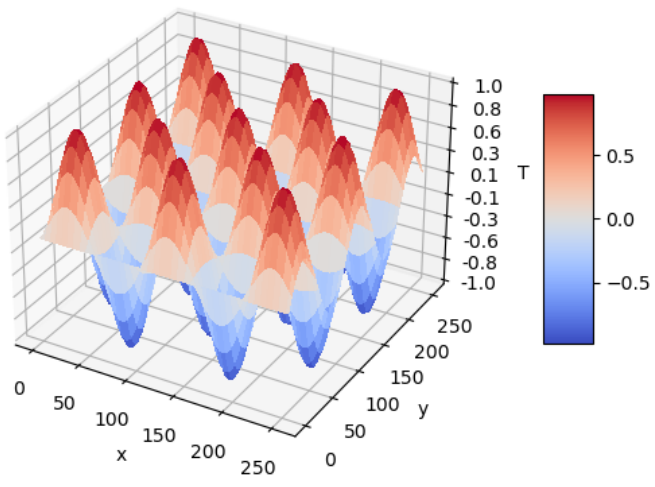


Figure 1: 3D surafce plot of temperature at iteration 0

```
cudaMalloc(&device_input_array, num_bytes);
cudaMalloc(&device_output_array, num_bytes);
...
...

// Deallocate memory from both device arrays
cudaFree(device_input_array);
cudaFree(device_output_array);
```

Submission information logs.

```
jelc@cardinal2:~$ /afs/ir.stanford.edu/class/cme213/script/submit.py hw3 private/cme213-
Submission for assignment 'hw3' as user 'jelc'
Attempt 2/10
Time stamp: 2022-05-01 21:36
List of files being copied:
    private/cme213-jelc53/hw3/main_q1.cu  [13253 bytes]
    private/cme213-jelc53/hw3/recurrence.cuh  [1589 bytes]
    private/cme213-jelc53/hw3/pagerank.cuh  [5894 bytes]
    private/cme213-jelc53/hw3/benchmark.cuh  [795 bytes]

Your files were copied successfully.
Directory where files were copied: /afs/ir.stanford.edu/class/cme213/submissions/hw3/jel
List of files in this directory:
    main_q1.cu  [13253 bytes]
    recurrence.cuh  [1589 bytes]
```

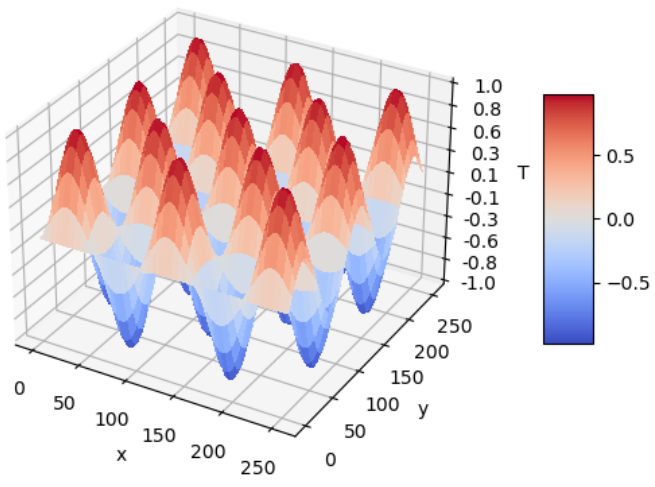


Figure 2: 3D surafce plot of temperature at iteration 1000

```
    pagerank.cuh  [5894 bytes]
    benchmark.cuh  [795 bytes]
    metadata  [137 bytes]

This completes the submission process. Thank you!

jelc@cardinal2:~$ ls /afs/ir.stanford.edu/class/cme213/submissions/hw3/jelc/2
benchmark.cuh  main_q1.cu  metadata  pagerank.cuh  recurrence.cuh
```

,,