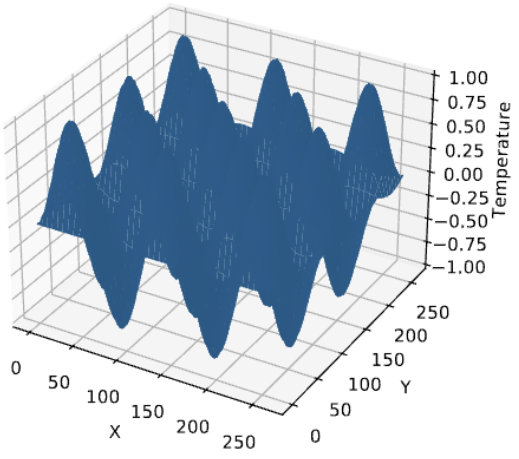
Homework3

Output:

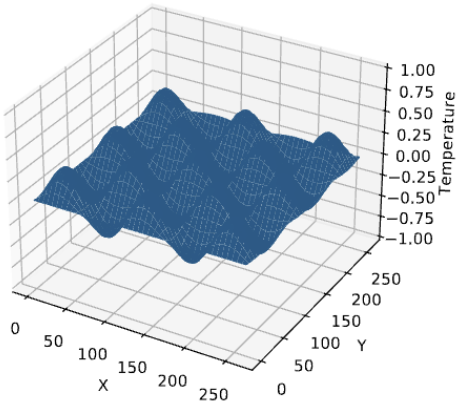
```
Order: 8, 256x256, 2000 iterations
                  time (ms)     GBytes/sec
           CPU        160.889        58.6565
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from DiffusionTest
[ RUN      ] DiffusionTest.GlobalTest
Order: 8, 256x256, 2000 iterations
                  time (ms)     GBytes/sec
        Global        69.8227        135.159

         L2Ref          LInf         L2Err
     0.0545994             0             0

[       OK ] DiffusionTest.GlobalTest (72 ms)
[----------] 1 test from DiffusionTest (72 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (72 ms total)
[  PASSED  ] 1 test.
```

Q1:

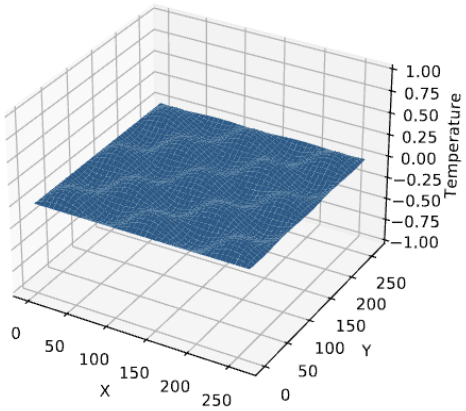
Iteration 0:



Iteration 1000:



Iteration 2000:



Q2:

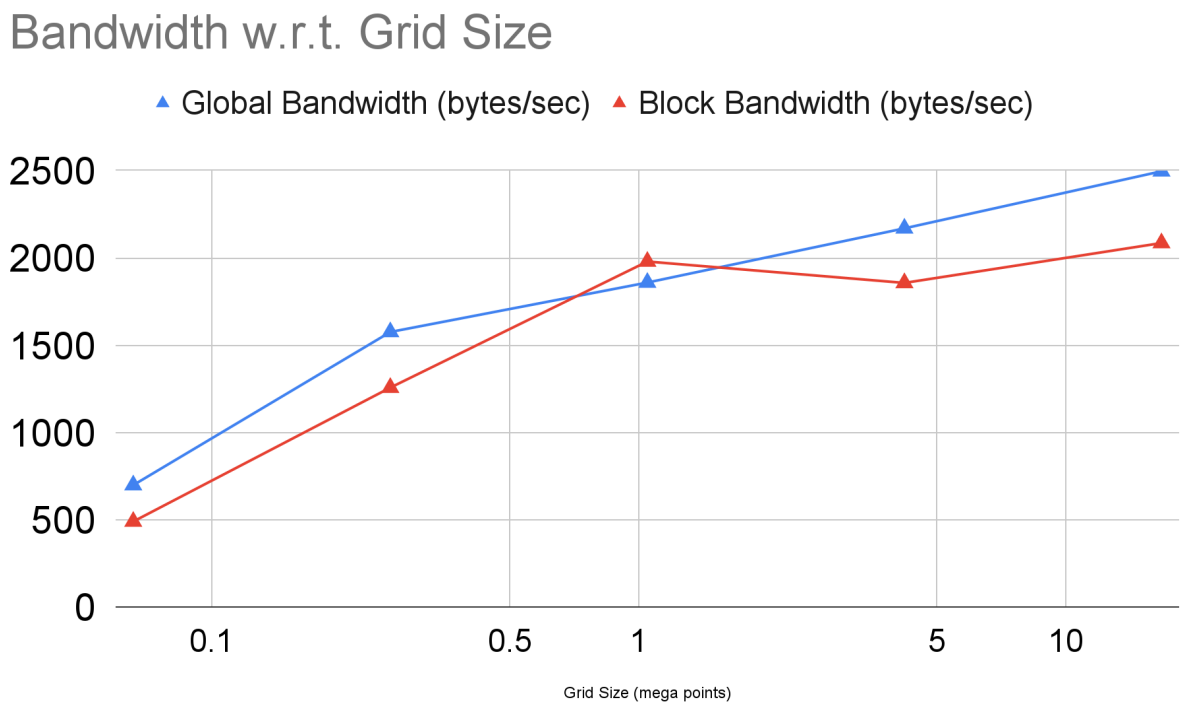
Output:

```
Order: 8, 4096x4096, 100 iterations
                time (ms)     GBytes/sec
         CPU        2148.57        56.2217
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 2 tests from DiffusionTest
[ RUN      ] DiffusionTest.GlobalTest
Order: 8, 4096x4096, 100 iterations
                time (ms)     GBytes/sec
       Global          51.2        2359.3

        L2Ref          LInf          L2Err
     0.447065             0              0
```

```
[       OK ] DiffusionTest.GlobalTest (184 ms)
[ RUN      ] DiffusionTest.BlockTest
Order: 8, 4096x4096, 100 iterations
                time (ms)     GBytes/sec
         Block       58.2812       2072.64

         L2Ref         LInf         L2Err
      0.447065            0             0

[       OK ] DiffusionTest.BlockTest (194 ms)
[----------] 2 tests from DiffusionTest (378 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (378 ms total)
[  PASSED  ] 2 tests.
```

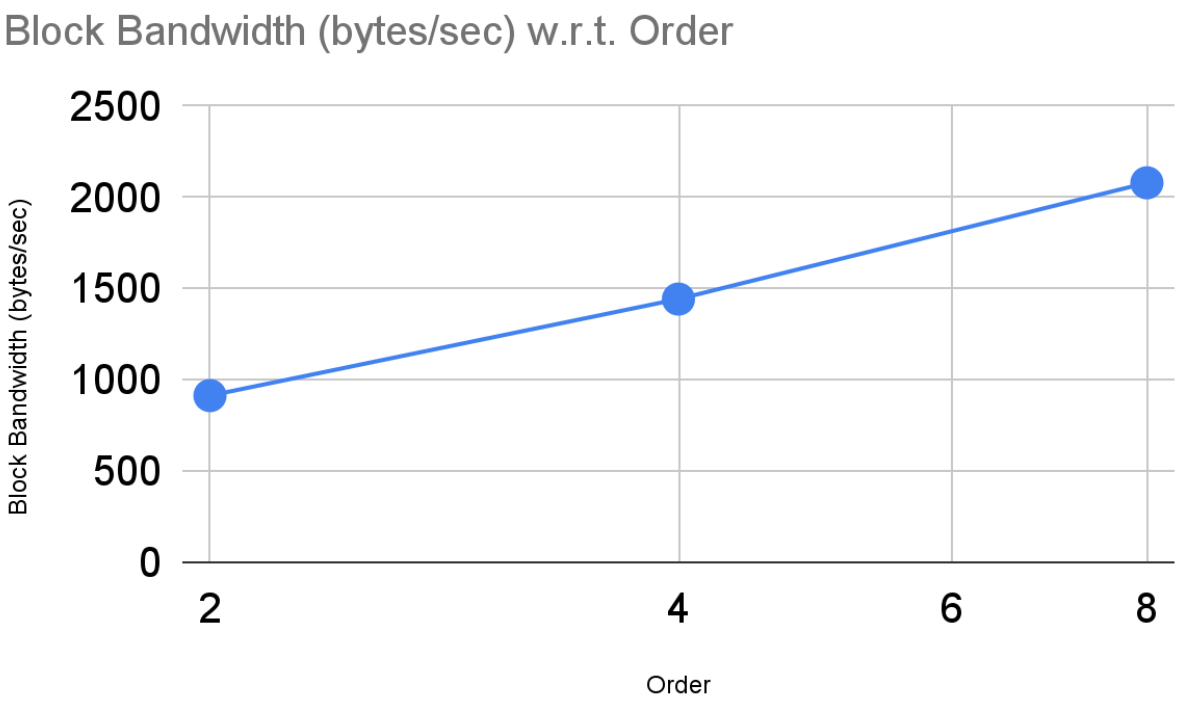
Q3:

1) For order = 8, plot the bandwidth for the 2 different algorithms.



2) For block algorithm, plot the bandwidth for the 3 different orders:

We fix the grid size to be 4096x4096, and use 1000 iterations.



Q4:

1) Performance among kernels:

   The global kernel largely outperforms the block kernel except for the 1024x1024 case. The reason is that there are so many choices of thread block decomposition in launching the kernel, and each one may have a different occupancy. So I am not surprised that on different problem sizes, the preferred kernel is different. In my code, I use a fixed number for choosing threads per block, but these numbers are not tuned. Theoretically, an autotuner should be written to tune these parameters for different input sizes so that the kernel can be optimized for a specific input size.

2) Performance among varying order:

   For the block kernel, the higher order has a higher bandwidth. This is because higher order stencils have higher arithmetic intensity. Higher-order stencils often lead to more data reuse – a single data point fetched from global memory is used for multiple computations.

3) Performance from varying grid size.

For global kernel, the performance goes up by increasing the grid size. But for block kernel, it peaks at the 1024x1024 case. So there is no direct conclusion about the performance concerning the grid size. To get the best performance, we need to tune the thread block decomposition factors for each input size.