Project #1: Finite Difference Solution of a Vibrating 2D Membrane on a GPU

Due date: 10/30/2020

Consider a rectangular membrane (e.g., a drumhead) 4ft by 2ft wide in *x* and *y* directions, respectively. The membrane is homogeneous, perfectly flexible and offers no resistance to bending. The displacement of the membrane from rest at a point (*x, y*) is governed by the two dimensional wave equation, which is a second order linear partial differential equation (PDE) that can be written as follows

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right), \tag{1}$$

where ϕ is the wave height, c^2 is is equal to 5, *x* and *y* are the spatial coordinates and *t* is the time. Note that c^2 represents membrane tension *T* divided by the membrane mass per unit area ρ. The two initial conditions are

$$\phi(x, y, t) = 0.1(4x - x^2)(2y - y^2), \tag{2}$$

$$\left.\frac{\partial \phi}{\partial t}\right|_{t=0} = 0, \tag{3}$$

The membrane is clamped at all the edges. Therefore we have the following boundary conditions.

$$\phi(0, y, t) = 0, \tag{4}$$

$$\phi(L_x, y, t) = 0, \tag{5}$$

$$\phi(x, 0, t) = 0, \tag{6}$$

$$\phi(x, L_y, t) = 0, \tag{7}$$

where L_x and L_y are the length of the membrane in *x* and *y* directions, respectively.

The wave equation is discretized on a uniform mesh such that $\Delta x = \Delta y = h$ with a second order accurate central differencing scheme. The discretized form of the PDE can be written as follows

$$\frac{\phi_{i,j}^{t+1} - 2\phi_{i,j}^{t} + \phi_{i,j}^{t-1}}{\Delta t^2} = c^2 \left(\frac{\phi_{i+1,j}^{t} - 2\phi_{i,j}^{t} + \phi_{i-1,j}^{t}}{\Delta x^2} + \frac{\phi_{i,j+1}^{t} - 2\phi_{i,j}^{t} + \phi_{i,j-1}^{t}}{\Delta y^2}\right), \tag{8}$$

where Δt is the time step size. The above equation can be rearranged to compute the future value at time *t+1* as follows

$$\phi_{i,j}^{t+1} = 2\phi_{i,j}^{t} - \phi_{i,j}^{t-1} + \frac{c^2 \Delta t^2}{h^2}\left(\phi_{i+1,j}^{t} + \phi_{i-1,j}^{t} + \phi_{i,j+1}^{t} + \phi_{i,j-1}^{t} - 4\phi_{i,j}^{t}\right). \tag{9}$$

Note that the right hand side depends on values from current (t) and old (t-1) time steps. From the discretization of the initial condition given in Eq. (3), we have the following

$$\frac{\phi_{i,j}^{t+1} - \phi_{i,j}^{t-1}}{2\Delta t} = 0. \tag{10}$$

Substituting the above relation into Eq. (9) we get

$$\phi_{i,j}^{t-1} = \phi_{i,j}^{t} + \frac{c^2 \Delta t^2}{2h^2}\left(\phi_{i+1,j}^{t} + \phi_{i-1,j}^{t} + \phi_{i,j+1}^{t} + \phi_{i,j-1}^{t} - 4\phi_{i,j}^{t}\right) \quad (11)$$

Then the algorithm to calculate the membrane motion can be listed as follows:

- Initialize the solution at time *t* using the initial condition given by Eq. (2) and at time *t-1* using Eq. (11).
- Apply the boundary conditions equations (4-7)
- Compute the membrane height at future time *t+1* using Eq. (9)
- Update your values at time *t* and *t-1*.
- Go to step 3 and repeat until a desired final time is reached.

Because we use an explicit scheme we have to worry about the numerical stability. Make sure your time step and grid size ensure the CFL condition

$$\frac{c\Delta t}{h} < 1.0$$

1. Plot the initial displacement of the membrane (Eq. 2) as a surface. You can use Matlab for plotting purposes.

2. Write a C program to compute the numerical solution. Your program should at the least have separate functions for the following tasks
   - Calculate the numerical mesh
   - Initialize the solution on the mesh
   - Compute the future value
   - Write the solution into a file

   First develop your serial CPU version using dynamic memory allocation with pointer arithmetic such that implementing the GPU version would result in a minimal change of your code. Make sure the serial version produces the same results as the analytical solution.

   You only need to have a GPU kernel for the section that computes the future value of the wave height. You do not need to convert the other components of the code for GPU since they are called once. There is no need to compute the analytical solution on the GPU.

3. Compute the CPU and GPU solutions on a mesh of 41×21 and compare the wave profiles $\phi(x, y, t)$ at $x = 0.25L_x$, $x = 0.5L_x$, $x = 0.75L_x$ and at four different instances against the analytical solution which can be written as

$$\phi(x, y, t) = 0.426050 \sum_{m=1,odd}^{\infty} \sum_{n=1,odd}^{\infty} \left(\frac{1}{m^3 n^3} \cos\left(t \frac{\sqrt{5}\pi}{4} \sqrt{m^2 + 4n^2} \right) \sin\left(\frac{m\pi x}{4} \right) \sin\left(\frac{n\pi y}{2} \right) \right)$$

   The analytical solution is from Chapter 12 of Advanced Engineering Mathematics by Kreyszig.

4. Plot a 3D surface contour plot of the wave near its maximum amplitude.

5. In your report, you need to discuss the parallel performance relative to the serial CPU version on a mesh of 514×258, 1026×514, 2050×1026 points (boundary points are included).

Investigate the impact of execution configuration on performance (i.e. 8×8, 16×16, 32×16 32×32 thread blocks). Note that once your code produces the correct results, which you can check on a modest mesh size against the analytical solution, you can then simulate it for a short period of time to get the timings. Make sure that both the CPU and GPU versions iterate the same number of time steps (i.e. the computational work assigned to both versions should be the same). Present your timing results as a table and a chart.

6. You need to present your work as a technical report. Follow a generic technical publication format for your report with citations, cross referencing of figures and tables etc. Your report should include a brief introduction on GPU computing and CUDA programming.
7. Follow the instructions for project and homework submission.