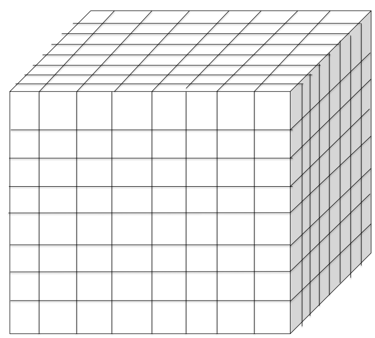
CMPE 478: Parallel Processing
Homework 1 (due Nov 18th)
(This project can be done in groups of at most 2 students)

The aim of this homework is to solve systems of equations that arise from the discretization of Poisson's equation on a cube domain. Poisson's equation is given as:

$$u_{xx} + u_{yy} + u_{zz} = f(x, y, z)$$

with (x, y, z) belonging to a square domain $[0, 1]^3$ and the values of the solution at the boundary points (x^*, y^*, z^*) specified. Your program should input n and discretize the domain into a $(n+1) \times (n+1) \times (n+1)$ cube grid :



In order to solve the partial differential equation, we will use a technique called *finite difference method*. Here, we will give present it briefly. Let $h = 1/n$ stand for the grid spacing. To approximate the partial derivatives, we will use Taylor's expansion. Using Taylor's expansion in the variable x results in:

$$u(x+h, y, z) = u(x, y, z) + hu_x(x, y, z) + \frac{1}{2}h^2 u_{xx}(x, y, z) + \frac{1}{6}h^3 u_{xxx}(x, y, z) + O(h^4)$$

$$u(x-h, y, z) = u(x, y, z) - hu_x(x, y, z) + \frac{1}{2}h^2 u_{xx}(x, y, z) - \frac{1}{6}h^3 u_{xxx}(x, y, z) + O(h^4)$$

Adding these last two equations together and solving for u_{xx} results in central difference approximation:

$$u_{xx}(x, y, z) = \frac{1}{h^2}[u(x+h, y, z) - 2u(x, y, z) + u(x-h, y, z)] + O(h^2)$$

If we use Taylor's expansion in the variable y, we get:

$$u_{yy}(x,y,z) = \frac{1}{h^2}[u(x,y+h,z) - 2u(x,y,z) + u(x,y-h,z)] + O(h^2)$$

Taylor's expansion in the variable z gives us:

$$u_{zz}(x,y,z) = \frac{1}{h^2}[u(x,y,z+h) - 2u(x,y,z) + u(x,y,z-h)] + O(h^2)$$

Introducing some shorthand notation for the coordinates:

$$(x_i, y_j, z_k) = (ih, jh, kh) \qquad (0 \leq i,j,k \leq n)$$

and the solution u and the function f as :

$$u_{i,j,k} = u(x_i, y_j, z_k) \qquad f_{i,j,k} = f(x_i, y_j, z_k)$$

we can write the approximation at each grid point as:

$$u_{i,j,k} = \frac{1}{6}[u_{i+1,j,k} + u_{i-1,j,k} + u_{i,j+1,k} + u_{i,j-1,k} + u_{i,j,k+1} + u_{i,j,k-1}] - \frac{1}{6n^2} f_{i,j,k}$$

You will need the boundary conditions (i.e. the solution at the domain boundaries). You can assume existence of a function called `exact(x,y,z)` that returns the solution value at the boundary points.

To-Do List

1. Write an OpenMP program that solves the resulting linear system by using Jacobi iteration. Use maximum norm to test for convergence.

2. Test your routines by solving the problem with $u(x,y,z) = xyz$ as the exact solution. Report timings and speedups obtained for various grid resolutions (n) and numbers of threads/cores used.