Fragment Identification Script

Kumar Mithraratne, University of Auckland. September 2014

Introduction

The purpose of this script is to identify fragments generated from a solid undergoing fracture. Once the fragments are identified and if they are not subject to any external forces, they can be treated as rigid bodies and hence reduce the computational costs involved in simulating solid fracture. The solid, which is being subject to external forces, typically impact forces, is represented by a collection of discrete particles. If there are fragments during fracture, the particles within each fragment will represent the fragment.

The cells in a background regular 3d grid (mesh) are used to register relative location of each particle. Once the particle locations with respect to grid cells are established, one can search for the cells in the neighbourhood to check what adjacent cells contain particles and form the fragments. The script consists of three parts: 1. Determine adjacent cells for each grid cell 2. Calculate particle location with respect to grid cells and 3. Identify fragments in terms of grid cells and hence particles in each fragments.

Determine adjacent cells for each grid cell

It is assumed that grid boundaries are parallel to spatial coordinate axes X,Y and Z. One can then easily generate a regular mesh consisting of several cells. The size of the cells determines the accuracy of fragment (boundary) identification. Fig. 1 depicts a typical 3d regular grid consisting of 1000 cells. Each cell has a unique identifier (number). These cells are numbered sequentially starting on Z = 0 plane marching from left to right (X direction) and progressing in the Y direction in the X-Y plane. Once a layer (X-Y plane) is finished then numbering moved to the next layer in the Z direction (see fig 1 for cell numbering).

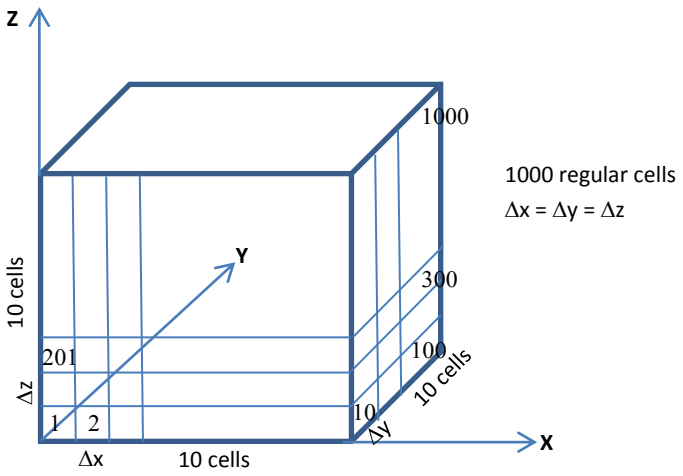

Figure 1

There are four types of cells found in the above grid (mesh) from neighbours' point of view.

(a) Corner cells – For any regular grid (fig. 1) in 3d, there are 8 corner cells. Each corner cell has 7 neighbours (adjacent cells)
(b) Edge cells – Cells along the edges excluding corner cells. There are 12 edges, 4 parallel to X axis, 4 parallel to Y axis etc. Each cell on the edge has 11 neighbouring cells.
(c) Surface cells – Cells on external surfaces. There are 6 such surfaces. 2 normal to X axis, 2 normal to Y axis etc. These surface cells have 17 adjacent cells each.
(d) Interior cells – These include all other cells which are not defined under (a) – (c). Each interior cell has 26 neighbouring cells.

There is an example file (grid.dat) of the grid in 'tecplot' format for visualisation.

Calculate particle location with respect to grid cells

This can be achieved relatively easily with this regular mesh and the numbering scheme that was described earlier. The details of this calculation can be easily inferred from the script and therefore not explained here. However, it is worth mentioning here that once the cell that hosts a particle is determined, this information is stored for both ways. i.e. particle-to-cell and cell-to-particle.

Particle (consecutively numbered) spatial coordinates are stored in a simple text file with comma-separated values (e.g. particle.dat in the directory). It is assumed that for each time step a file containing particle spatial coordinates will be output from the fracture simulation code.

Identify fragments in terms of grid cells and hence particles in each fragments.

The following flow chart explains the fragment search algorithm implemented in the script.