Design Credit MEN2020

Name: Satyam Kumar Roll No.: B19ME076

Generation of binary files for flow visualization in Tecplot using MPI

Supervisor Name: Dr. Nipun Arora

Project Overview

Tecplot's tecio library was used to generate binary files, with MPI for parallel processing.

- Instanlled the libraries and utilised the compilers that the software supported.
- For flow visualisation, I converted the ASCII files to binary files.
- The processors generated the data and created the data files simultaneously.

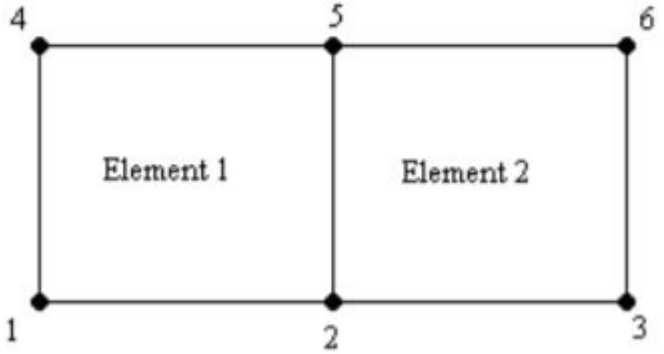
Data Structure

There are two forms of data.

I) Ordered Data: points in a one-dimensional, two-dimensional, or three-dimensional array.

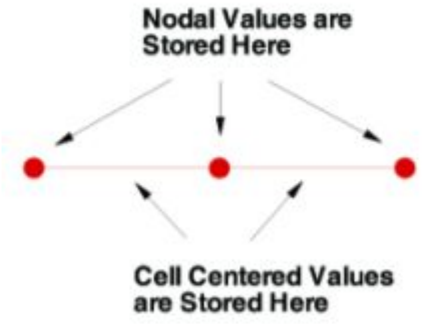
II) Finite Element Data : consist of two arrays, a variable array and a connectivity matrix.

http://home.ustc.edu.cn/~cbq/360_c

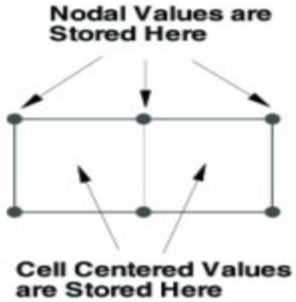


Ordered Data

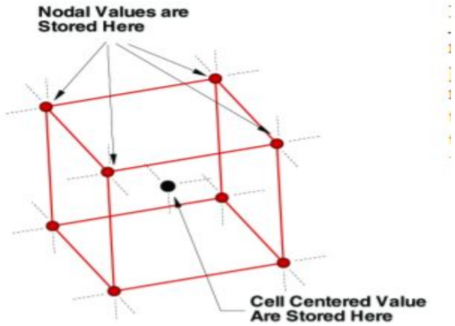
- One-dimensional Ordered Data (I-ordered, J-ordered, or K-ordered)
- Two-dimensional Ordered Data (IJ-ordered, JK-ordered, IK-ordered)
- Three-dimensional Ordered Data (IJK-ordered)



(i)



(ii)

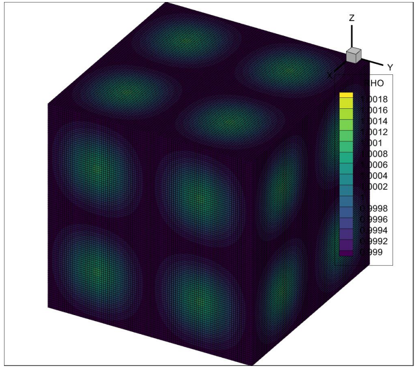


(iii)

Work Done

01 Converting the ASCII format to binary files

```
VARIABLES= X, Y, Z, RHO, UX, UY, UZ
ZONE I =100, J=100, K=100
1    1    1    0.99884222    0.00001645    0.00001645     0.00001645
2    1    1    0.99884332    0.00002987    0.00002698     0.00001016
3    1    1    0.99884515    0.00003756    0.00004384    -0.00000021
4    1    1    0.99884771    0.00003745    0.00006606    -0.00001441
5    1    1    0.99885102    0.00002836    0.00009241    -0.00003215
6    1    1    0.99885505    0.00001005    0.00012146    -0.00005296
```



- Converted ASCII files to binary files, allowing for faster Tecplot visualisation.
- We used a set of functions for this, which are as follows:
  - TECINI142 (initialize the data file)
  - TECZNE142 (information for the next zone to be added)
  - TECDAT142 (to add the data in form of arrays to the zone)
  - TECEND142 (to close the current data file)
- It takes a long time to read large files with millions of points.

02 Writing the data by the processors directly to binary

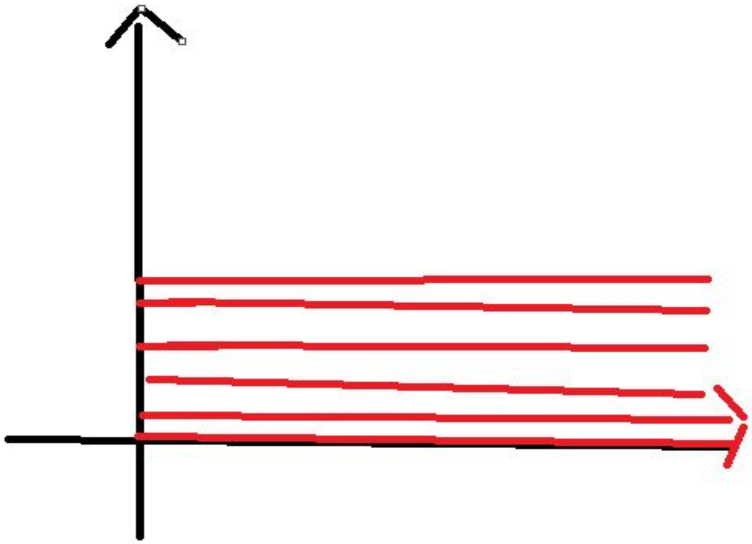
- Instead of reading, the data from the code was appended, resulting in a considerable reduction in the function's run time.

- Using the index, I created 1D arrays for the data points and directly inserted the arrays to the binary file.

- Processor limitations in handling such a high amount of data points

http://home.ustc.edu.cn/~cbq/360_data_format_guide.pdf

```
double U0 = 0.05;

for (int k = 0; k < ZDIM; k++)
{

    for (int j = 0; j < YDIM; j++)
    {

        for (int i = 0; i < XDIM; i++)
        {
            int index = (k * YDIM + j) * XDIM + i;
            x[index] = (double)i;
            y[index] = (double)j;
            z[index] = (double)k;
            rho[index] = 1;
            ux[index] = U0 * sin(i * (2 * pi / XDIM)) * (cos(3 * j * (2 * pi /
            uy[index] = U0 * sin(j * (2 * pi / YDIM)) * (cos(3 * k * (2 * pi /
            uz[index] = U0 * sin(k * (2 * pi / ZDIM)) * (cos(3 * i * (2 * pi /
        }
    }
}
```



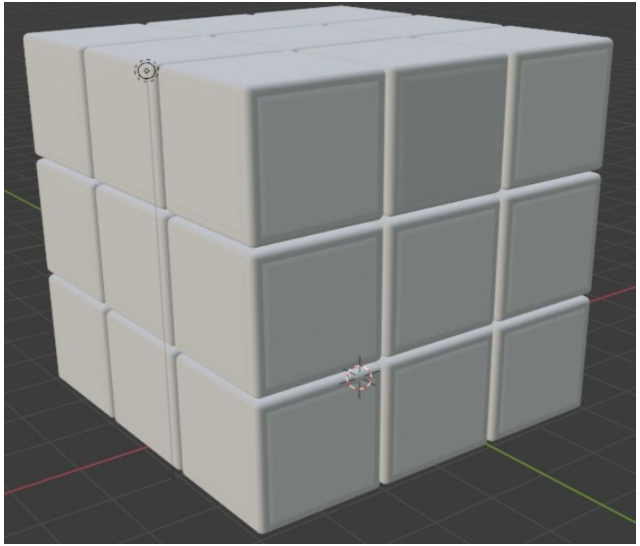
03 Integrating parallel processing and TecioMPI

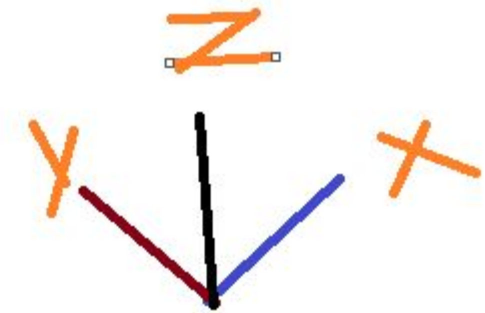
- Partitioned the data sets, and each CPU adds its own partition.

```
INTEGER4 numPartitions = 4;
vector<INTEGER4> partitionOwners;
for (INTEGER4 ptn = 0; ptn < numPartitions; ++ptn)
    partitionOwners.push_back(ptn % commSize);
TECZNEMAP142(&numPartitions, &partitionOwners[0]);


for (INTEGER4 partition = 1; partition <= 4; ++partition)
{

    if (commRank == mainRank || partitionOwners[partition - 1] == commRank)
    {

        INTEGER4 partitionIMin = partitionIndices[partition - 1][0];
        INTEGER4 partitionJMin = partitionIndices[partition - 1][1];
        INTEGER4 partitionKMin = partitionIndices[partition - 1][2];
        INTEGER4 partitionIMax = partitionIndices[partition - 1][3];
        INTEGER4 partitionJMax = partitionIndices[partition - 1][4];
        INTEGER4 partitionKMax = partitionIndices[partition - 1][5];
        I = TECIJKPTN142(&partition, &partitionIMin, &partitionJMin, &partitionKMin, &partitionIMax, &partitionJMax, &pa
        I = outputVarData(x, XDIM, YDIM, ZDIM, partitionIMin, partitionJMin, partitionKMin, partitionIMax, partitionJMax
```

- Additional functions used here were as follows:
  - TECMPIINIT142 (Initializes MPI and joins a specified MPI communicator)
  - TECZNEMAP142 (it maps the processes to the partition to be printed in the zone)
  - TECIJKPTN142 (manages information about partitions, later reassembled into a single zone)
- For TECIJKPTN142 to function properly, the ending and starting indices of the next partition must be the same.

How Did we partition?





https://free3d.com/3d-model/rubiks-cube-5499.html

Thank You