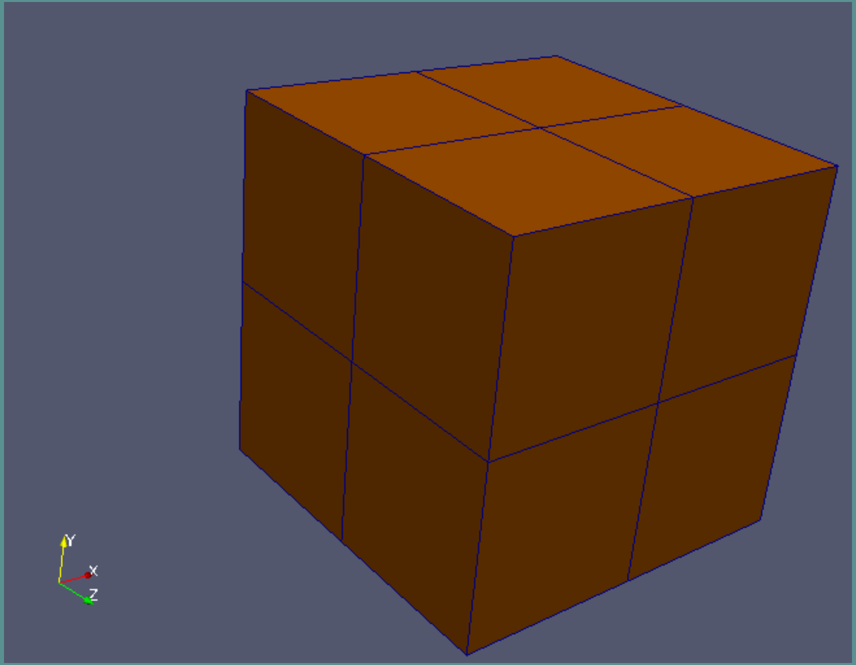
Computational Mechanics: HDF Parallel I/O Implementation in Warp3D

Daniel Pledger Carlos Estrada Rocco Febbo
Mentors: Dr. Tim Truster Dr. Kwai Wong

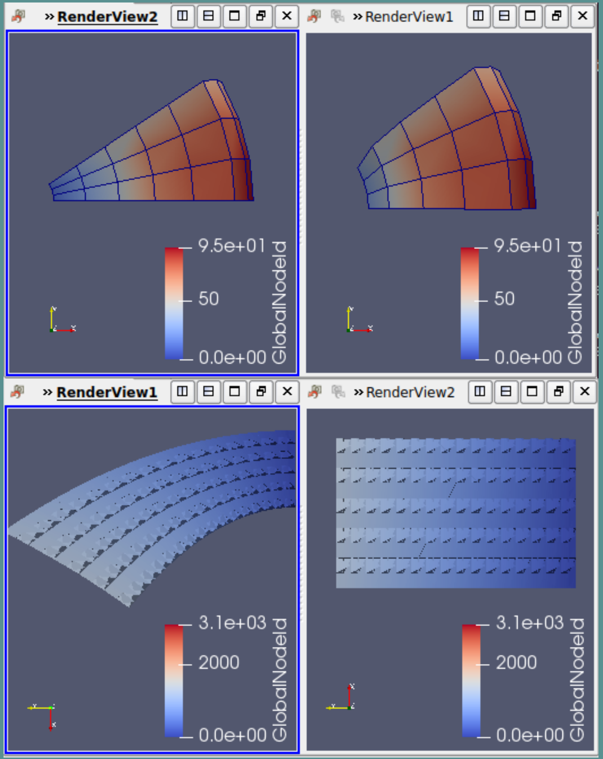
Project Overview

Create a workflow using multiple open source programs to create more complex geometries for Warp3D input

Use HDF5 to store output in parallel from Warp3D to be rendered by visualization software like Paraview



Warp3D

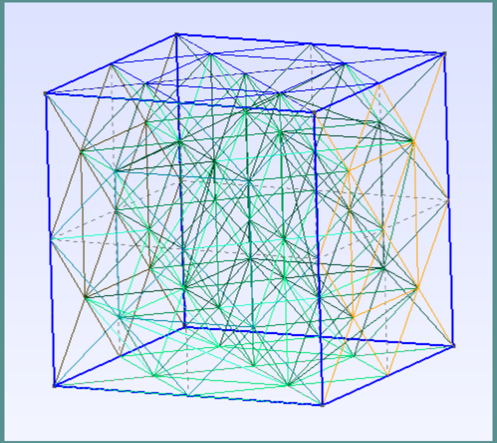


Description: Open source code for 3D nonlinear analysis of solids primarily for fatigue and fracture simulations for materials under static, dynamic ,and thermal loadings

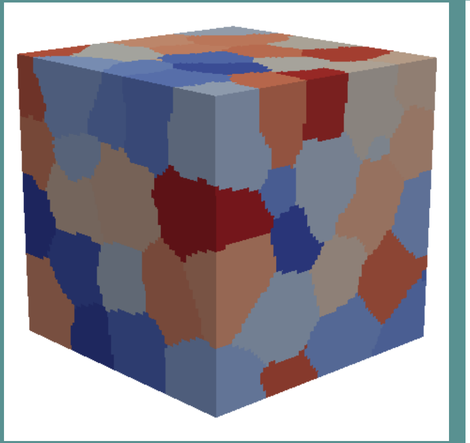
Purpose: Analyze material mechanics under stresses to improve designs.

Code: Written in Fortran (Late 1980’s - Current)

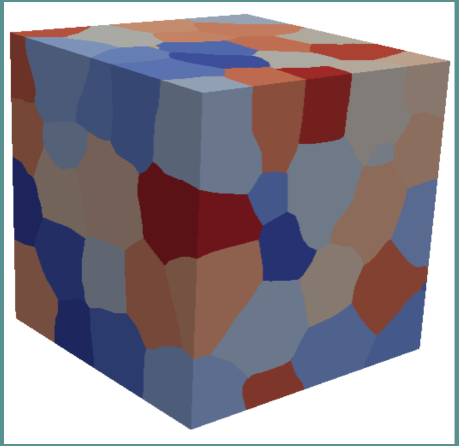
Front End



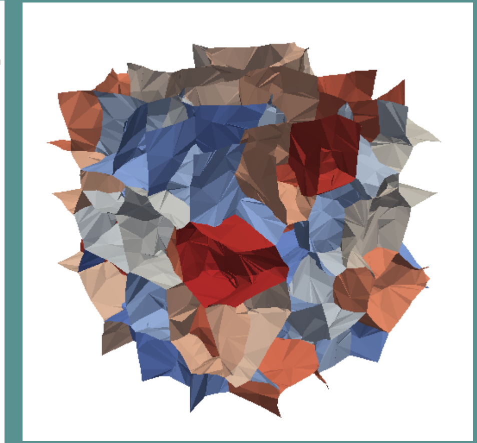
Gmsh



Dream3D



Voxel2Tet

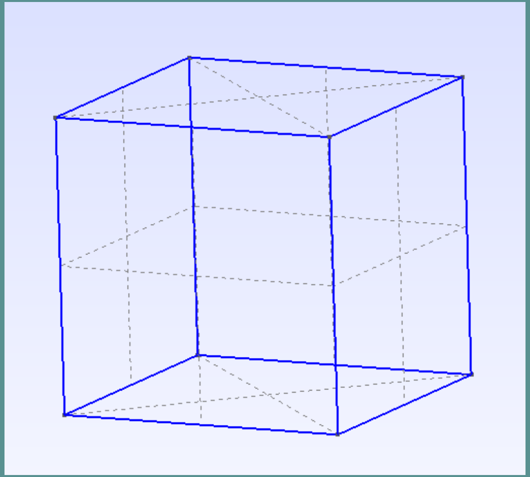


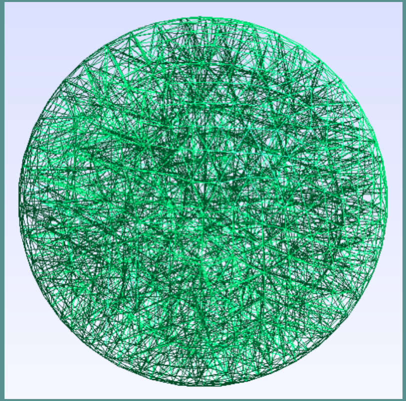
DEIP

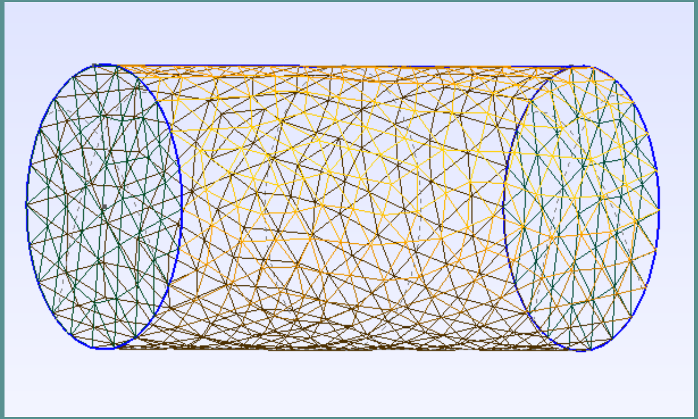
Gmesh

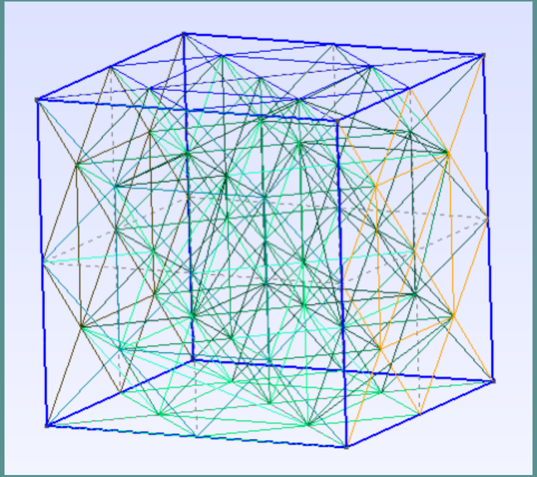
Description: Gmsh is a free 3D finite element mesh generator with a built-in CAD engine

Purpose: Gmsh was used to create model geometries and meshes. Gmsh comes ready to export model and mesh data to an STL file.





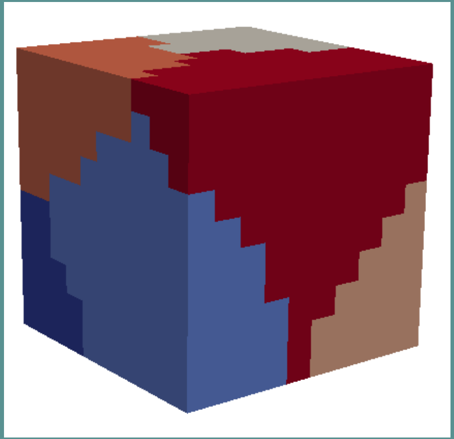
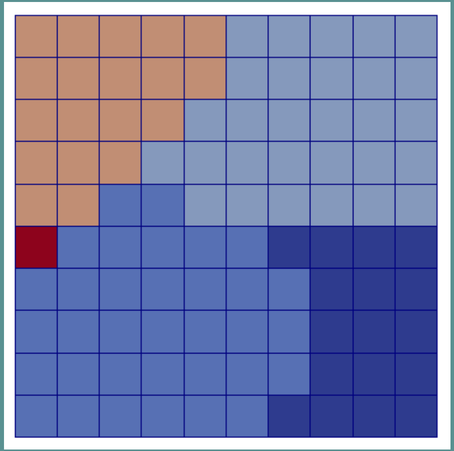
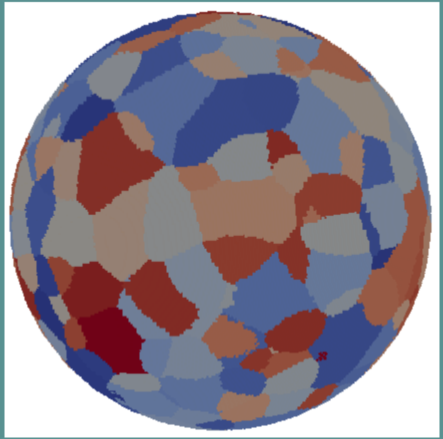




Dream3d

Description: Dream3D allows users to fill solid CAD models with microstructural grains created from input statistics and properties.

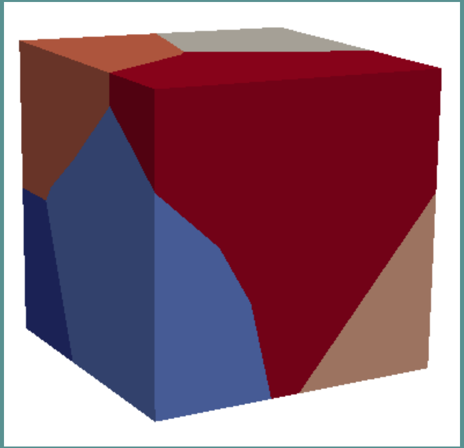
Purpose: Dream3D was used to create initial grain structures in Gmsh models. Dream 3D can input STL files from Gmsh and output a .dream3D file containing grain and model data.

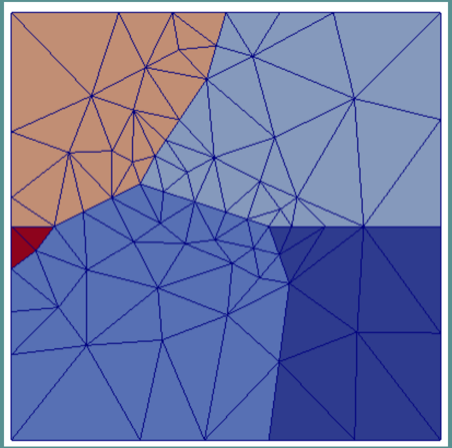




Voxel2Tet

Description: Voxel2Tet is a code written in C++ that can take voxel (cubic) grain structures and convert them to tetrahedral mesh with smooth interfaces.

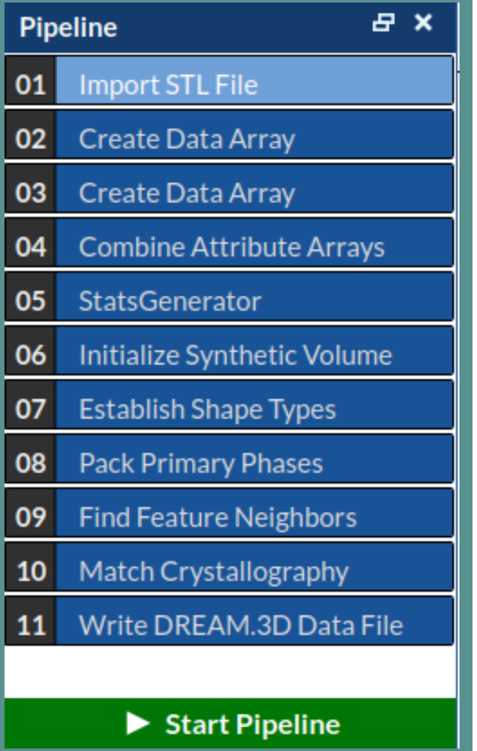
Purpose: Voxel2Tet was used to make Dream3D grain structures smoother and more realistic. Users can input a .dream3D file, and Voxel2Tet will output an Abaqus .inp file containing grain and model data.





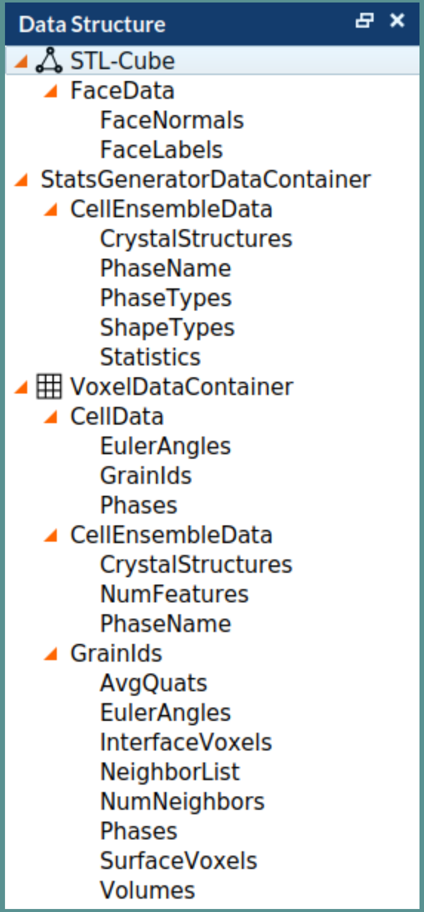
Voxel2Tet cont.

- Voxel2Tet last updated in 2016.
- Source code modification to read most recent Dream3D data outputs
- Reading of source code to find required data and structure for Voxel2Tet to run successfully
- Created a specific dream3d pipeline to create suitable files for Voxel2Tet input



Pipeline

01 Import STL File
02 Create Data Array
03 Create Data Array
04 Combine Attribute Arrays
05 StatsGenerator
06 Initialize Synthetic Volume
07 Establish Shape Types
08 Pack Primary Phases
09 Find Feature Neighbors
10 Match Crystallography
11 Write DREAM.3D Data File



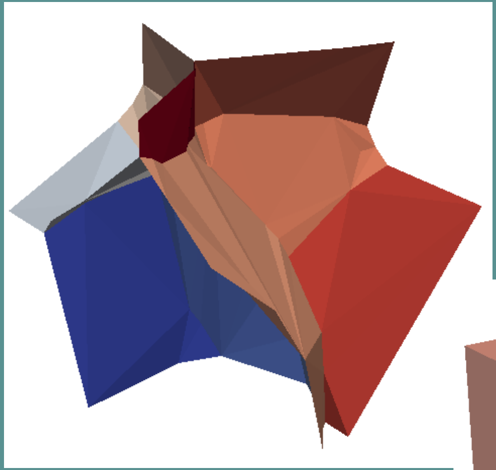
Data Structure

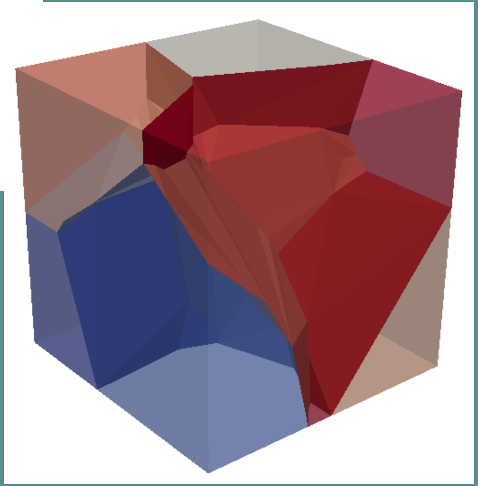
- STL-Cube
  - FaceData
    - FaceNormals
    - FaceLabels
- StatsGeneratorDataContainer
  - CellEnsembleData
    - CrystalStructures
    - PhaseName
    - PhaseTypes
    - ShapeTypes
    - Statistics
- VoxelDataContainer
  - CellData
    - EulerAngles
    - GrainIds
    - Phases
  - CellEnsembleData
    - CrystalStructures
    - NumFeatures
    - PhaseName
  - GrainIds
    - AvgQuats
    - EulerAngles
    - InterfaceVoxels
    - NeighborList
    - NumNeighbors
    - Phases
    - SurfaceVoxels
    - Volumes

Discontinuous Element Insertion Program (DEIP)

Description: Discontinuous Element Insertion Program is a program written in MATLAB that inserts zero-thickness elements in between grain surfaces in a finite element mesh in two and three dimensions

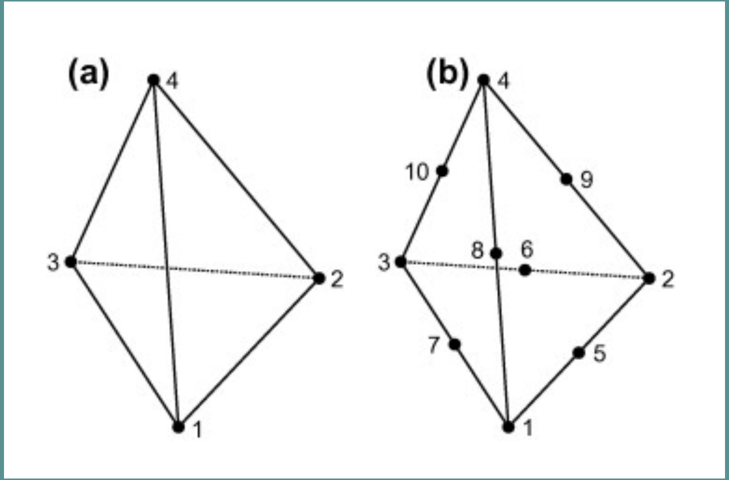
Purpose: DEIP was used to place interface elements between the Voxel2Tet grain structures. DEIP comes with a Warp3D input file writer to output model, grain, and grain interface data into Warp3D for simulation.

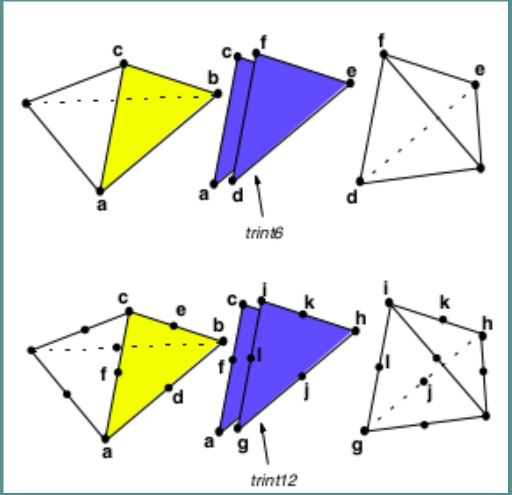




DEIP cont.

- MATLAB program was written to read the Voxel2Tet abaqus output file.
- Linear tetrahedral elements were converted to quadratic using code written by John Burkardt, this step was added to the DEIP program.
- Warp3D file writer needed modification to account for quadratic elements and surface elements





Overview of Front End Workflow

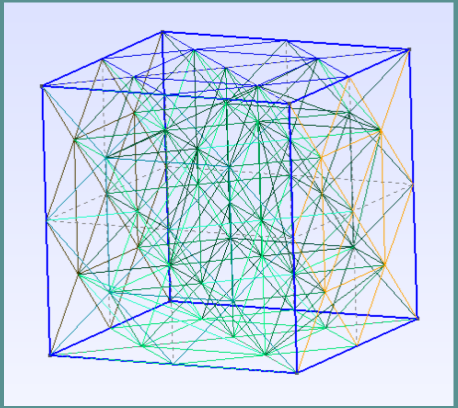
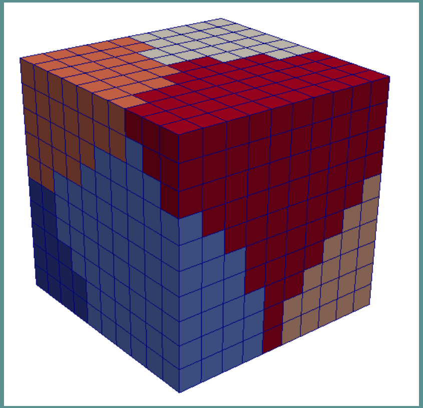
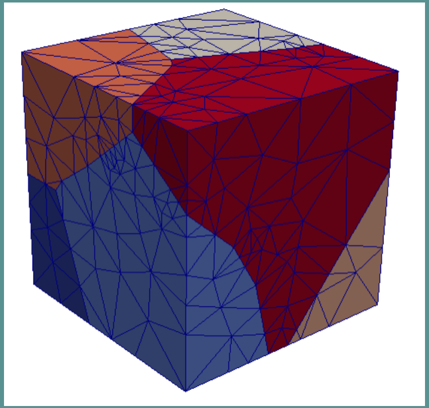
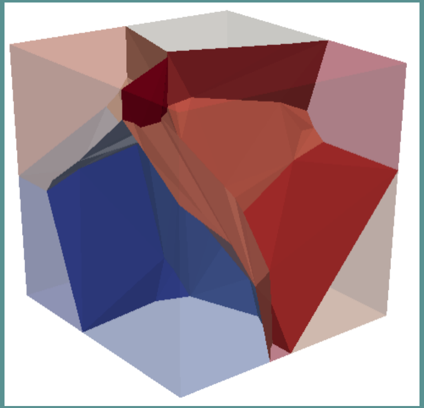
Gmsh → Dream3D → Voxel2Tet → DEIP

Gmsh is used to create initial geometry and mesh

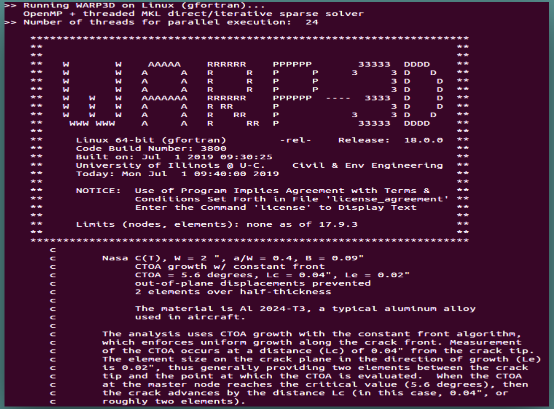
Dream3D creates initial grain structures

Voxel2Tet smoothes and converts elements to tetrahedral

DEIP inserts interface elements and writes Warp3D input file

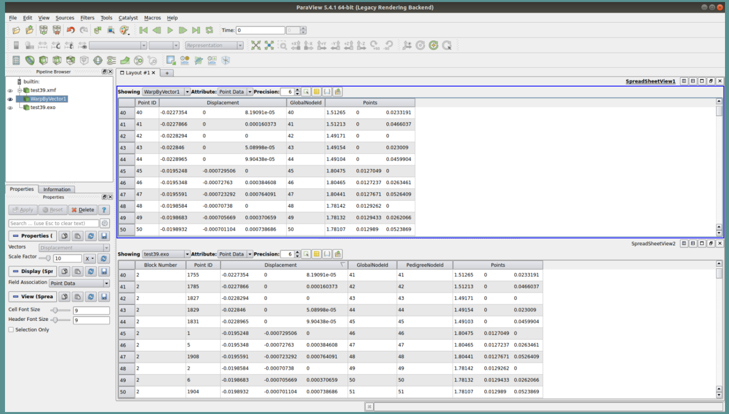
Back End



```
>> Running WARP3D on Linux (gfortran)...
   OpenMP + threaded MKL direct/iterative sparse solver
>> Number of threads for parallel execution:  24

    *******************************************************************
    **                                                               **
    **                                                               **
    **   W       W    AAAAA    RRRRRR    PPPPPP       33333  DDDD    **
    **   W       W   A     A   R     R   P     P     3     3 D   D   **
    **   W       W   A     A   R     R   P     P           3 D    D  **
    **   W       W   A     A   R     R   P     P           3 D    D  **
    **   W   W   W   AAAAAAA   RRRRRR    PPPPPP  ----  3333  D    D  **
    **   W   W   W   A     A   R RR      P                 3 D    D  **
    **   W   W   W   A     A   R   RR    P           3     3 D   D   **
    **    WWW WWW    A     A   R     RR  P            33333  DDDD    **
    **                                                               **
    **     Linux 64-bit (gfortran)        -rel-    Release:  18.0.0  **
    **     Code Build Number: 3800                                   **
    **     Built on: Jul  1 2019 09:30:25                            **
    **     University of Illinois @ U-C.    Civil & Env Engineering  **
    **     Today: Mon Jul  1 09:40:00 2019                           **
    **                                                               **
    **     NOTICE:  Use of Program Implies Agreement with Terms &    **
    **              Conditions Set Forth in File 'license_agreement' **
    **              Enter the Command 'license' to Display Text      **
    **                                                               **
    **     Limits (nodes, elements): none as of 17.9.3               **
    **                                                               **
    *******************************************************************
       c
       c       Nasa C(T), W = 2 ", a/W = 0.4, B = 0.09"
       c            CTOA growth w/ constant front
       c            CTOA = 5.6 degrees, Lc = 0.04", Le = 0.02"
       c            out-of-plane displacements prevented
       c            2 elements over half-thickness
       c
       c            The material is Al 2024-T3, a typical aluminum alloy
       c            used in aircraft.
       c
       c       The analysis uses CTOA growth with the constant front algorithm,
       c       which enforces uniform growth along the crack front. Measurement
       c       of the CTOA occurs at a distance (Lc) of 0.04" from the crack tip.
       c       The element size on the crack plane in the direction of growth (Le)
       c       is 0.02", thus generally providing two elements between the crack
       c       tip and the point at which the CTOA is evaluated.  When the CTOA
       c       at the master node reaches the critical value (5.6 degrees), then
       c       the crack advances by the distance Lc (in this case, 0.04", or
       c       roughly two elements).
```
]

Paraview



Time: 0

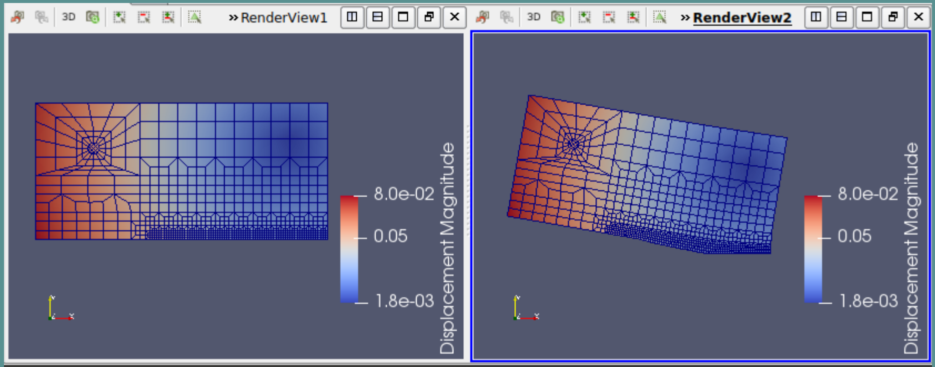
Vectors: Displacement
Scale Factor: 10
Field Association: Point Data
Cell Font Size: 9
Header Font Size: 9
Selection Only

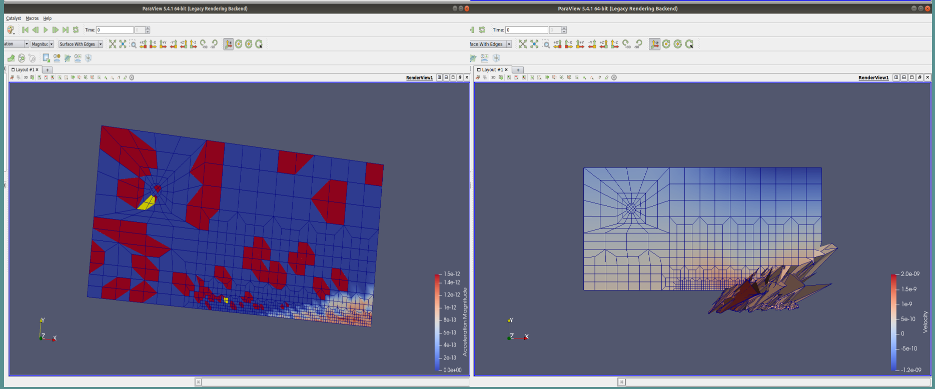
Showing: WarpByVector1 Attribute: Point Data Precision: 6

	Point ID	Displacement			GlobalNodeId	Points		
40	40	-0.0227354	0	8.19091e-05	40	1.51265	0	0.0233191
41	41	-0.0227866	0	0.000160373	41	1.51213	0	0.0466037
42	42	-0.0228294	0	0	42	1.49171	0	0
43	43	-0.022846	0	5.08998e-05	43	1.49154	0	0.023009
44	44	-0.0228965	0	9.90438e-05	44	1.49104	0	0.0459904
45	45	-0.0195248	-0.000729506	0	45	1.80475	0.0127049	0
46	46	-0.0195348	-0.00072763	0.000384608	46	1.80465	0.0127237	0.0263461
47	47	-0.0195591	-0.000723292	0.000764091	47	1.80441	0.0127671	0.0526409
48	48	-0.0198584	-0.00070738	0	48	1.78142	0.0129262	0
49	49	-0.0198683	-0.000705669	0.000370659	49	1.78132	0.0129433	0.0262066
50	50	-0.0198932	-0.000701104	0.000738686	50	1.78107	0.012989	0.0523869

Showing: test39.exo Attribute: Point Data Precision: 6

	Block Number	Point ID	Displacement			GlobalNodeId	PedigreeNodeId	Points		
40	2	1755	-0.0227354	0	8.19091e-05	41	41	1.51265	0	0.0233191
41	2	1785	-0.0227866	0	0.000160373	42	42	1.51213	0	0.0466037
42	2	1827	-0.0228294	0	0	43	43	1.49171	0	0
43	2	1829	-0.022846	0	5.08998e-05	44	44	1.49154	0	0.023009
44	2	1831	-0.0228965	0	9.90438e-05	45	45	1.49103	0	0.0459904
45	2	1	-0.0195248	-0.000729506	0	46	46	1.80475	0.0127049	0
46	2	5	-0.0195348	-0.00072763	0.000384608	47	47	1.80465	0.0127237	0.0263461
47	2	1908	-0.0195591	-0.000723292	0.000764091	48	48	1.80441	0.0127671	0.0526409
48	2	2	-0.0198584	-0.00070738	0	49	49	1.78142	0.0129262	0
49	2	6	-0.0198683	-0.000705669	0.000370659	50	50	1.78132	0.0129433	0.0262066
50	2	1904	-0.0198932	-0.000701104	0.000738686	51	51	1.78107	0.012989	0.0523869





Description: 3D Object Rendering Software

Purpose: Visualize a 3D object and represent displacements, temperatures, etc. calculated in Warp3D

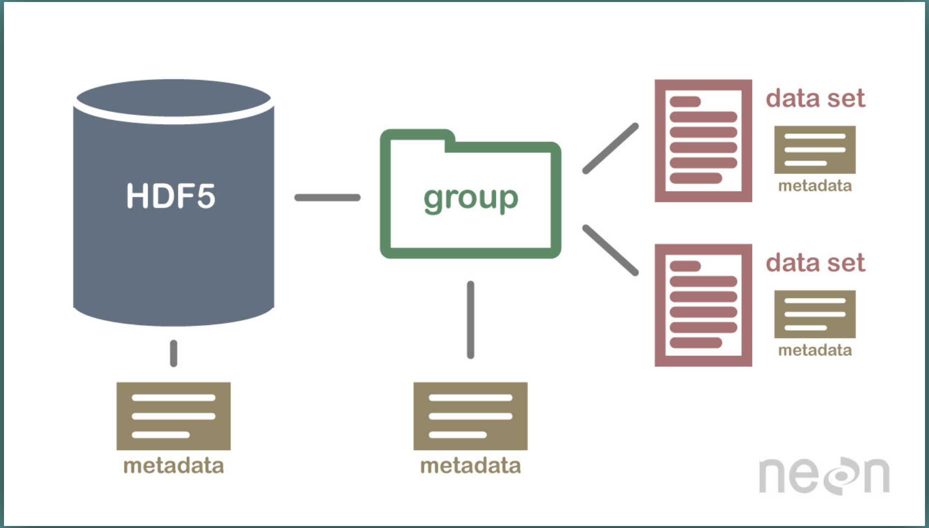
Program Use: Opens .exo file given by Warp3D or .xmf file used with .h5 file

HDF5

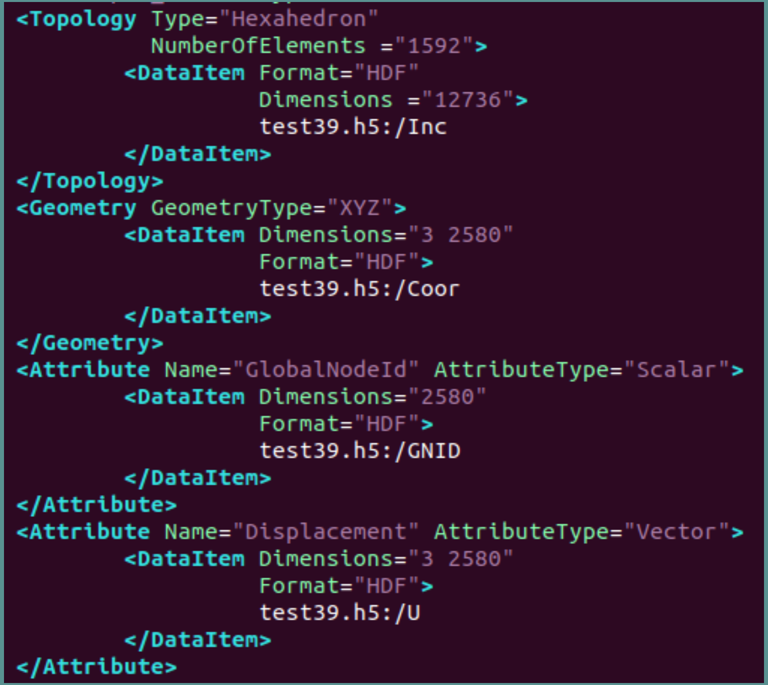
Description: File Type .h5

Purpose: Store files/data by efficient and compact means using a hierarchical format (similar to Unix file storage and directories)

Code: Written in C, C++, Fortran



XDMF



```
<Topology Type="Hexahedron"
          NumberOfElements ="1592">
        <DataItem Format="HDF"
                  Dimensions ="12736">
                  test39.h5:/Inc
        </DataItem>
</Topology>
<Geometry GeometryType="XYZ">
        <DataItem Dimensions="3 2580"
                  Format="HDF">
                  test39.h5:/Coor
        </DataItem>
</Geometry>
<Attribute Name="GlobalNodeId" AttributeType="Scalar">
        <DataItem Dimensions="2580"
                  Format="HDF">
                  test39.h5:/GNID
        </DataItem>
</Attribute>
<Attribute Name="Displacement" AttributeType="Vector">
        <DataItem Dimensions="3 2580"
                  Format="HDF">
                  test39.h5:/U
        </DataItem>
</Attribute>
```

Description: File Type .xmf

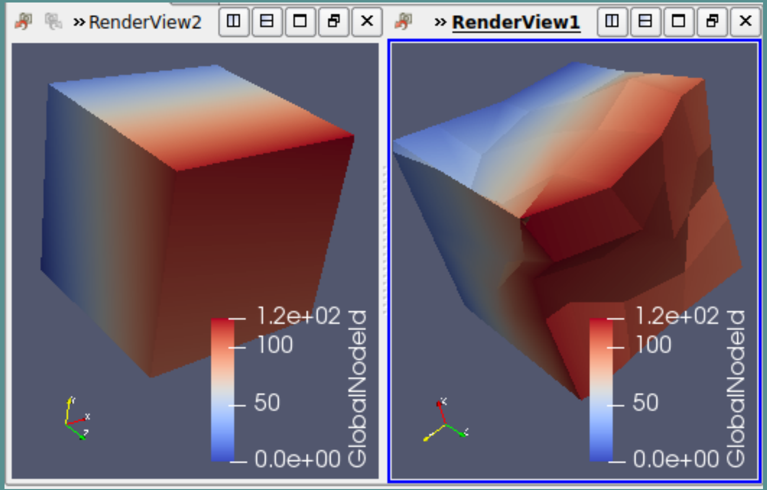
Purpose: XDMF uses XML to store Light data and describe the data Model. Either HDF5 or binary files can be used to store Heavy data.

Light data or "metadata" is used by software such as Paraview to read and render data from Heavy data files like .h5

Code: Written in C, C++, Fortran

Measures Taken

- Installed, compiled and ran all required programs separately
- Wrote program to convert .geo or Patran Format .text to .h5 and opened with Paraview by giving a .xmf to represent formatting and attributes
- Learned the structure of Warp3D source and subroutine calls
- Manipulated and developed Warp3D source code to write to .h5 from C function called in Fortran
- Developed workflow through various front end programs to better represent 3D objects resulting in more accurate simulations



What’s next?

- Compare Warp3D results for models with and without grain structures and grain boundary interfaces.
- Implement parallelism in Voxel2Tet and DEIP to lower run times.
- Test input workflow on large scale, complex geometries to find real world solutions.
- Persist in achieving more accurately simulated models for optimal data most precise to the physical world
- Continue developing parallelism in input and output processing

References

1. C. Geuzaine and J.-F. Remacle. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. International Journal for Numerical Methods in Engineering, 2009
2. Truster TJ. *Discontinuous element insertion algorithm. Advances in Engineering Software*, 2015
3. Dodds, R. (2019, July 01), et al. *WARP3D-Release 18.1.5 3-D Dynamic Nonlinear Fracture Analyses of Solids Using Parallel Computers*. University of Illinois at Urbana-Champaign
4. Sandstrom, Carl. *A Novel Tool for Converting the Voxel Representation of Microstructures ToSmooth Tetrahedral Meshes*. 2016.
5. *XDMF: Main Page*. (2017, March 13). Retrieved July 31, 2019, from http://www.xdmf.org/index.php/Main_Page

Questions?