VizSchema – A Standard Approach for Visualization Of Computational Accelerator Physics Data

Sveta Shasharina#, John Cary, Marc Durant, Scott Kruger, Seth Veitzer

sveta@txcorp.com

Outline

- The importance of being standard
- Motivation
- VizSchema details
- Examples
- Future Directions

Thanks to:

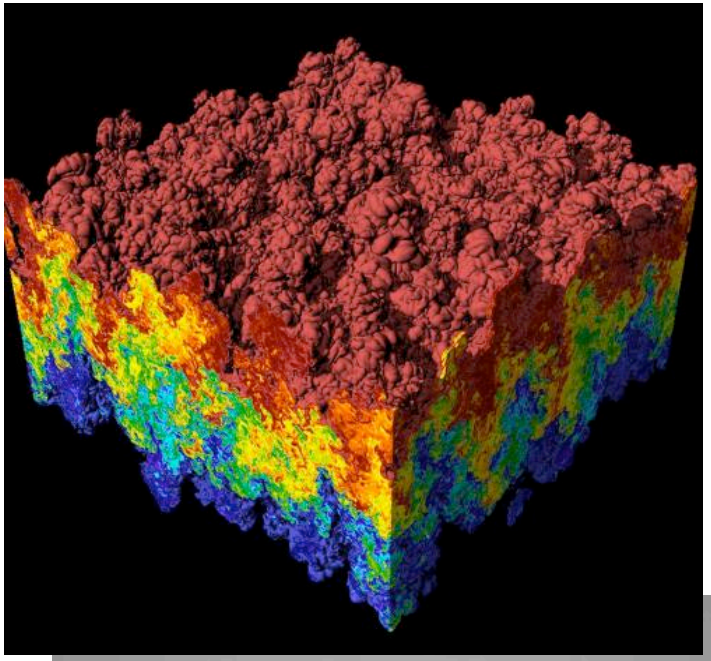
- VisIt team (H. Childs, S. Ahern, J. Meredith, B. Whitlock, A. Sanderson)
- Tech-X FACETS and VORPAL teams
- C. Geddes and G. Weber

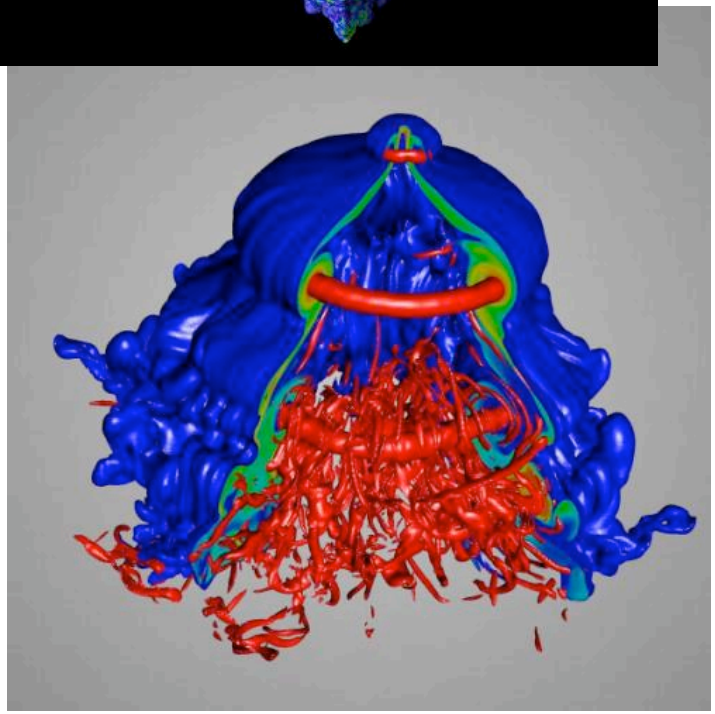
Simulation Data and VizTools are Diverse

- Data
  - ASCII
  - All kinds of home grown binary format
  - Lately moving to self-described data formats
    - NetCDF
    - HDF5 (our current choice)
- VizTools
  - IDL
  - Matlab
  - AVS/Express
  - Lately moving to open standard tools
    - VisIt (our current choice)
    - ParaView

VisIt Rules But…

- VisIt – supported by VACET SciDAC
  - Free
  - Powerful
  - Great team to work with!
- Has internal data structures
- One needs to write a reader for each type of data format to transform into what VisIt expects





One Cannot Write One Reader For All HDF5 data

- HDF5 consists of groups (like directories), datasets (like files – end leaves) and attributes (for small data and metadata)
- One can organize data in many ways using HDF5 constructs
- One could use any kinds of names and no metadata at all
- How one can understand what is what?
  - What is supposed to be visualized?
  - What order is used (row-major etc)?
  - Where is the mesh of the data?
  - What are the components of the mesh?

VizSchema Is An Attempt to Standardize HDF5 for Viz

- Based on experience working with
  - VORPAL (uniform and unstructured meshes and particles)
  - NIMROD (structured meshes)
  - UEDGE (structured, multi-domain meshes)
  - TXFLUID (unstructured meshes)
  - FACETS (combination of the above)
- VizSchema
  - Data Model (agreement about data organization and metadata)
  - C++ reader (independent of Viz tool) of HDF5 data into in memory viz objects
  - VisIt plugin
- Funded by FSML grant (DOE SBIR, FES), FACETS grant (DOE SciDAC) and Tech-X Corporation

Some standards live forever

- Why a Space Shuttle is “this” big?
- Its rocket boosters need to fit in a train tunnel
- US rails are 4 feet 8 1/2 inches apart
- They were built by English expatriates – the same way as road for wagons in England
- Wagon wheels were built to fit old roads ruts
- Old roads were build by Romans
- Roman chariots were build to accommodate two horses back-ends

(Thanks to Wes Bethel for this story)

VizSchema: Minimalistic but Enough for Viz

- Viz entities (based on our experience):
  - Variables (live on external mesh)
  - Variables with meshes (spatial info is mixed in)
  - Meshes
  - Derived variables
- Metadata (internal to files vs XML, for example)
  - Identifying the entities and specifying their kinds (if any)
  - Providing information needed for minimal viz
- Principles
  - Metadata is minimal (and results of many friendly discussions)
  - Metadata is in attributes starting with “vs”
  - Groups and datasets names are not regulated

Variables Need to State Type, Centering (and Ordering)

```
Dataset "phi" {
  Att vsType = "variable"  // Register as something
     // to "look at" and look for vsMesh
  Att vsMesh = "mycartgrid"// This should exist
                          //somewhere in the file
  Att vsCentering = "zonal"// Nodal is default
}
```

Variables With Mesh Show Where to Find Coordinates

```
Dataset "vorpalElectrons" {
  Att vsType = "variableWithMesh" // Find
      //vsNumSpatialDims or VsSpatialIndices
  Att vsNumSpatialDims = 3 // First 3 indices are
   // coordinates
}

Dataset "synergiaElectrons" {
  Att vsType = "variableWithMesh"
  Att vsSpatialIndices = [1, 3, 5] // These indices
      // are coordinates
}
```

Everything Is 1D Array – Which Order?

- Start with fastest varying (3D):

```
compMinorC = (i0, i1, i2, ic)
compMinorF = (ic, i2, i1, i0)
compMajorC = (ic, i0, i1, i2)// Same as compMinorF for 1D
compMajorF = (i2, i1, i0, ic)// Same as compMinorC for 1D
```

- For example, compMajorC in 2D (component after component, row after row):

```
Bx(0,0),Bx(1,0),Bx(2,0),Bx(0,1),Bx(1,1),Bx(2,1),
By(0,0),By(1,0),By(2,0),By(0,1),By(1,1),By(2,1)
```

- One needs to use VsOrder attribute for datasets in variables, variables with meshes and meshes to specify this (compMinorC is default)

Meshes Specify Kinds And Things to Build Itself

- Structured grid has just a list of points
  - Need vsOrder (if not default)
- Rectilinear mesh has list of numbers in each direction
- Uniform mesh has number of points in each direction, max and min in each direction
- Unstructured mesh needs points and elements (polygons, quads etc)
- The list of meshes will grow
- Work with users to provide alternative ways to describes same kinds of meshes

Derived Variables

```
Group anygroupname {
  ATT vsType = "variableDefinition"
  ATT vsDefinition = "elecEnergyDensity =
(E_0*E_0+E_1*E_1+E_2*E_2)"
}
```

- Count on expression language of VisIt
- Use variables as basis
- Make viz richer

Multi-Domain Variables For Integrated Simulations

- Variable can live in many domains but use different names
- MD (Multiple Domain) capability allows combining these in one for integrated viz (skipped other attributes)

```
Dataset "privMesh" {
  Att vsType = “mesh”
  Att vsMD = “edgeMesh”}
Dataset "solMesh" {
  Att vsType = “mesh”
  Att vsMD = "edgeMesh” }
Dataset "psiPriv" {
  Att vsType = “variable”
  Att vsMesh = “privMesh”
  Att vsMD = “psi” }
Dataset "psiSol" {
  Att vsType = "variable"
  Att vsMesh = "solMesh"
  Att vsMD = "psi" }
```

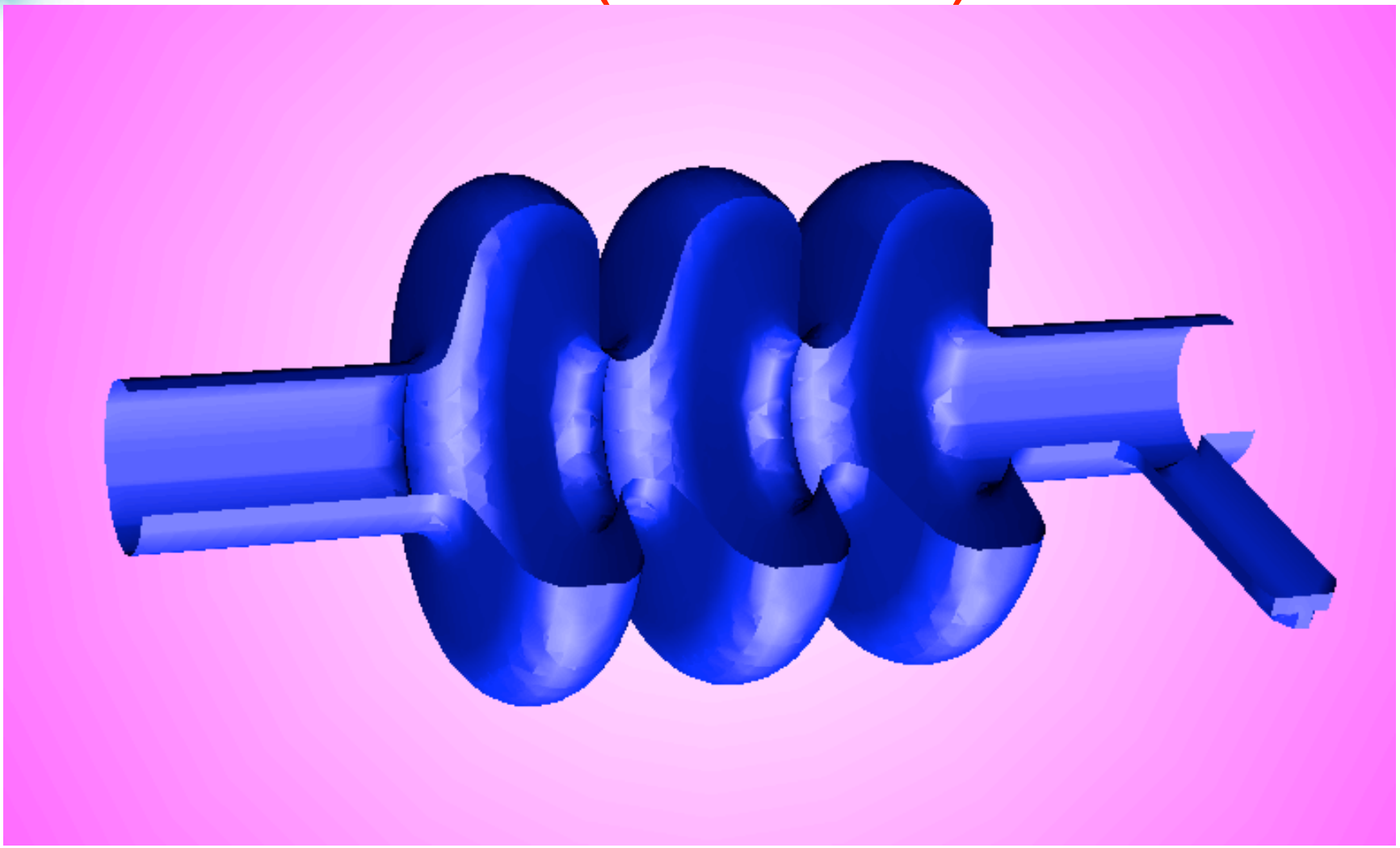
How To Adopt VizSchema?

- Change your I/O
  - VORPAL
  - FACETS (Fusion SciDAC)
  - NIMROD (MHD)
  - PolySwift++ (nanotech)
- Or change your files using PyTables (very easy interface to modify and add attributes)

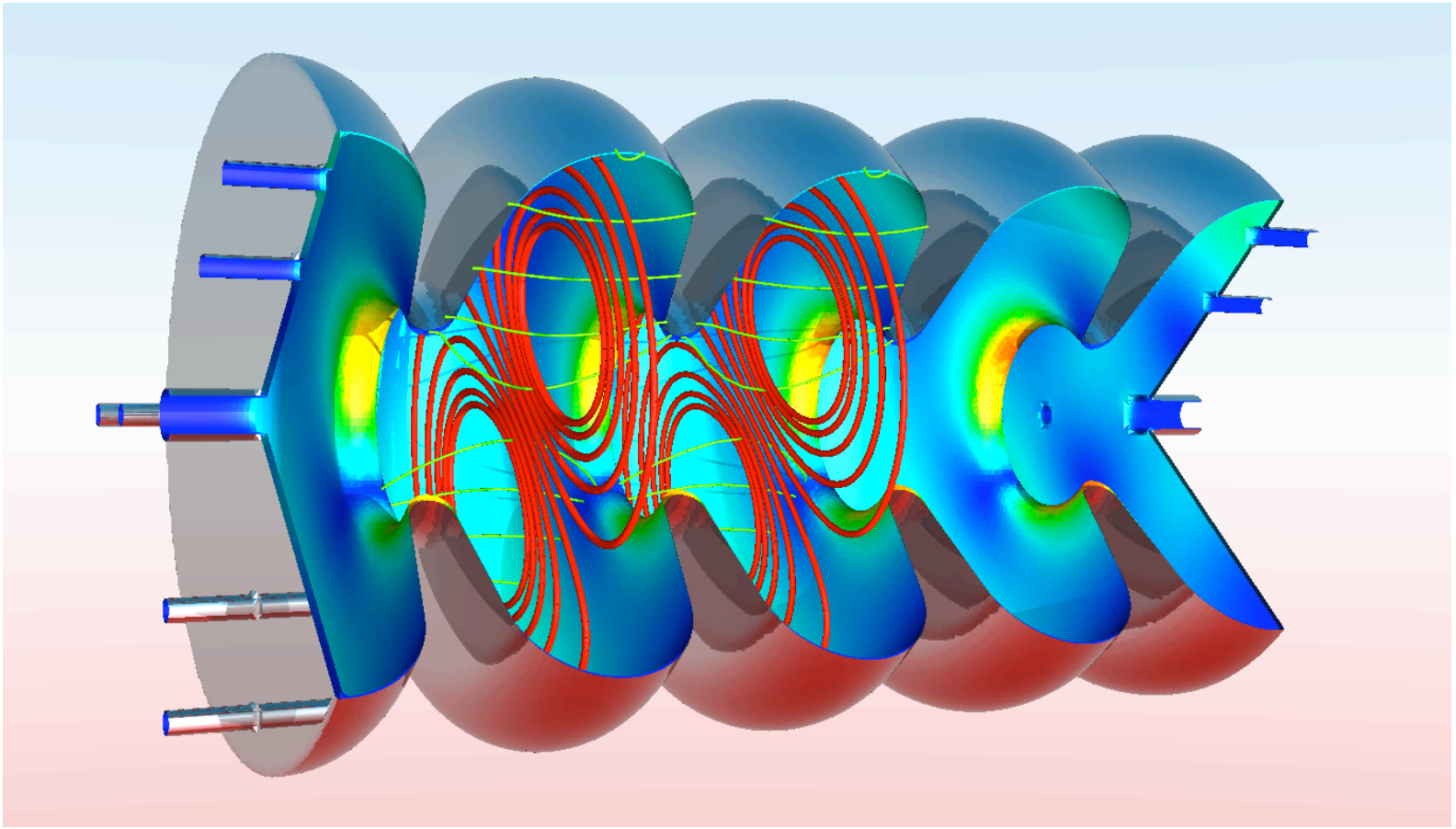
```
h5file = tables.openFile(fileName, mode='a')
dataSet = h5file.getNode("/" + dataSetName)
dataSet.attrs.vsType = "variable"
h5file.close()
```

  - Changing old VORPAL outputs to fit evolving schema
  - Changing SYNERGIA output
- First step is look at “h5dump –A” command and send to us…

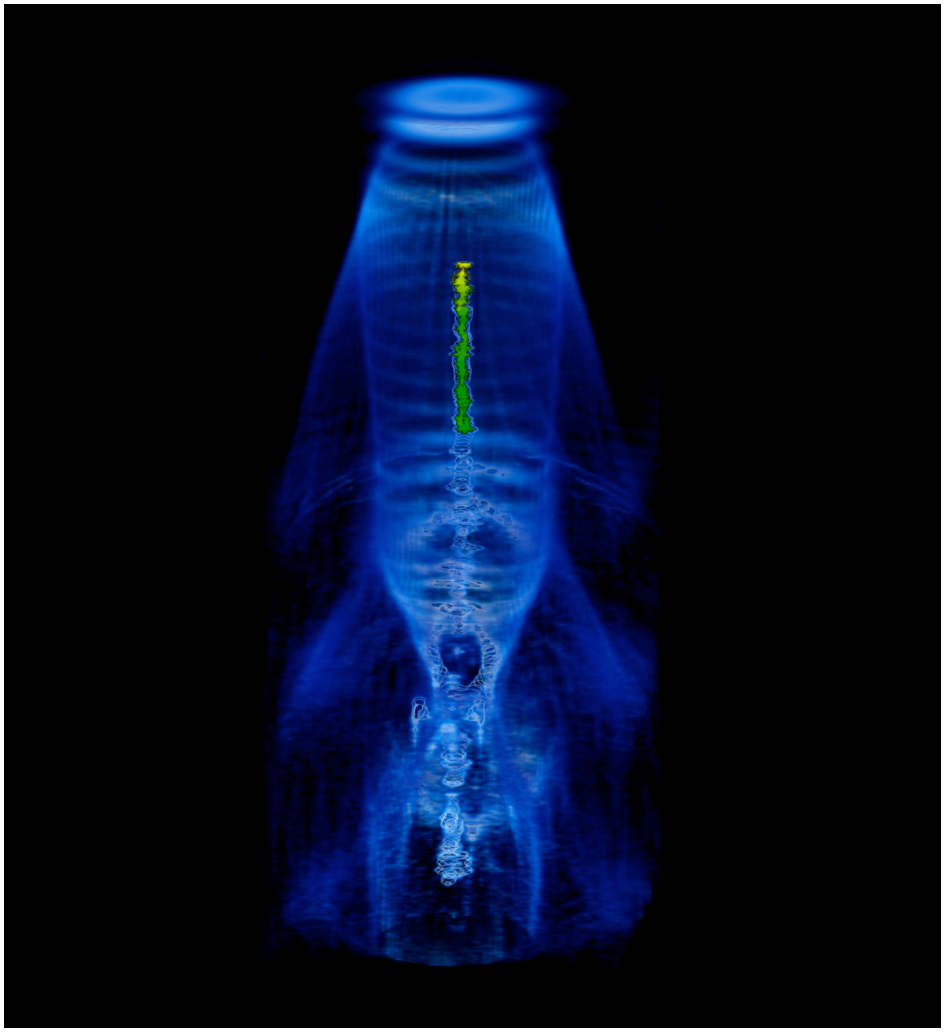
RF Cavity Represented As Unstructured Mesh of Polygons (VORPAL)

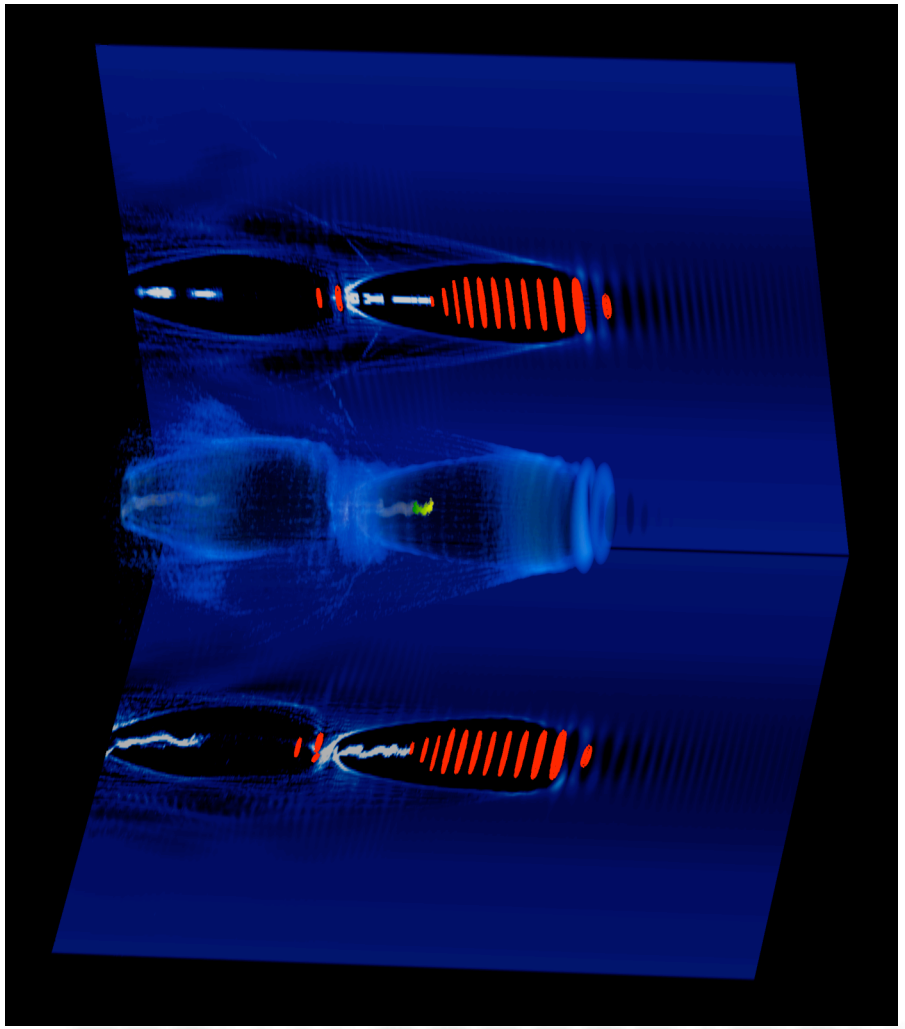


RF Cavity Represented as Unstructured Mesh of Polygons Colored By Magnetic Field Magnitude of the Surface. Electric and Magnetic Field Lines (VORPAL)

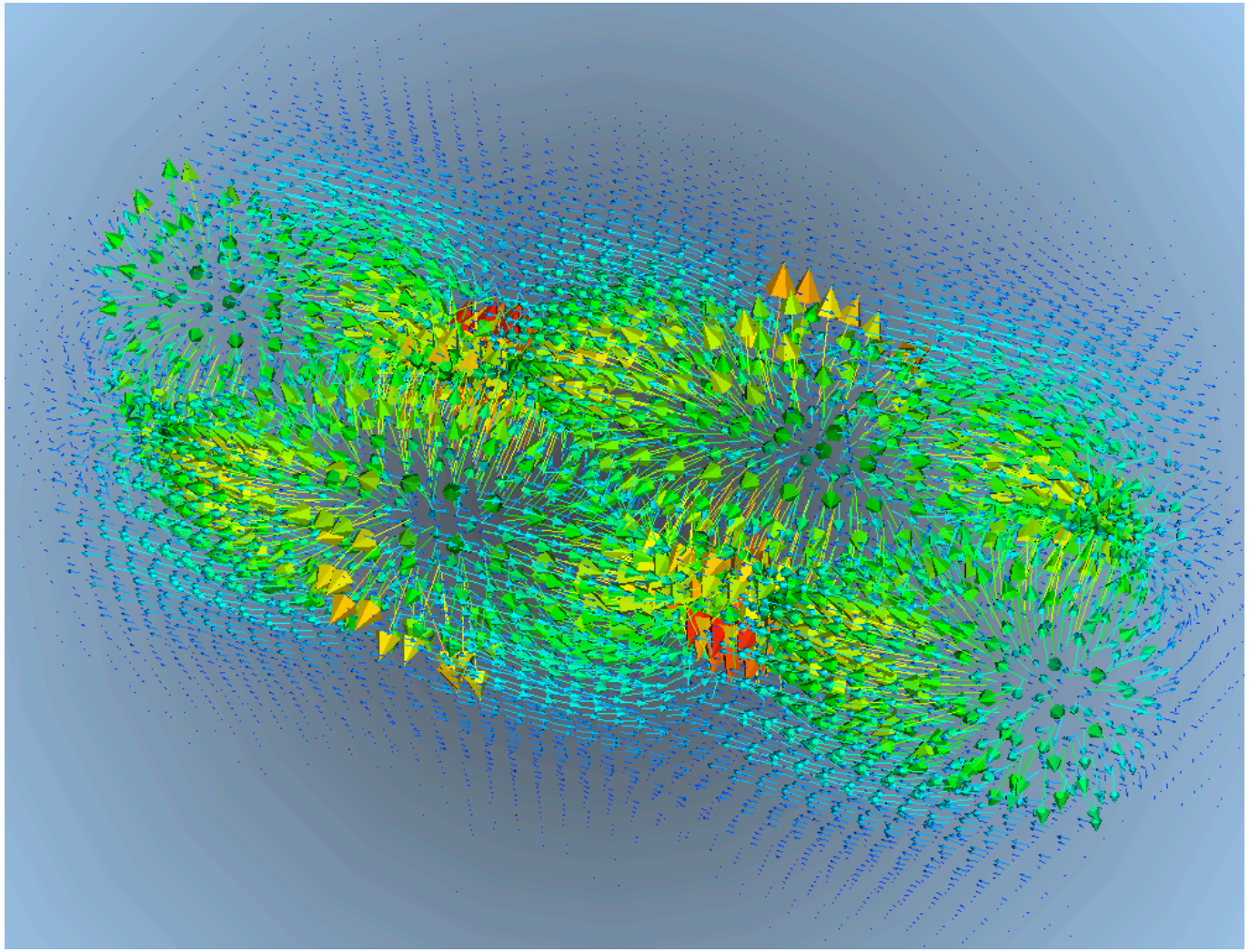


Laser-Plasma Accelerator Simulation (VORPAL). Left: plasma bubble and generated beam. Right: Laser field. Images - courtesy of C. Geddes and G. Webber.

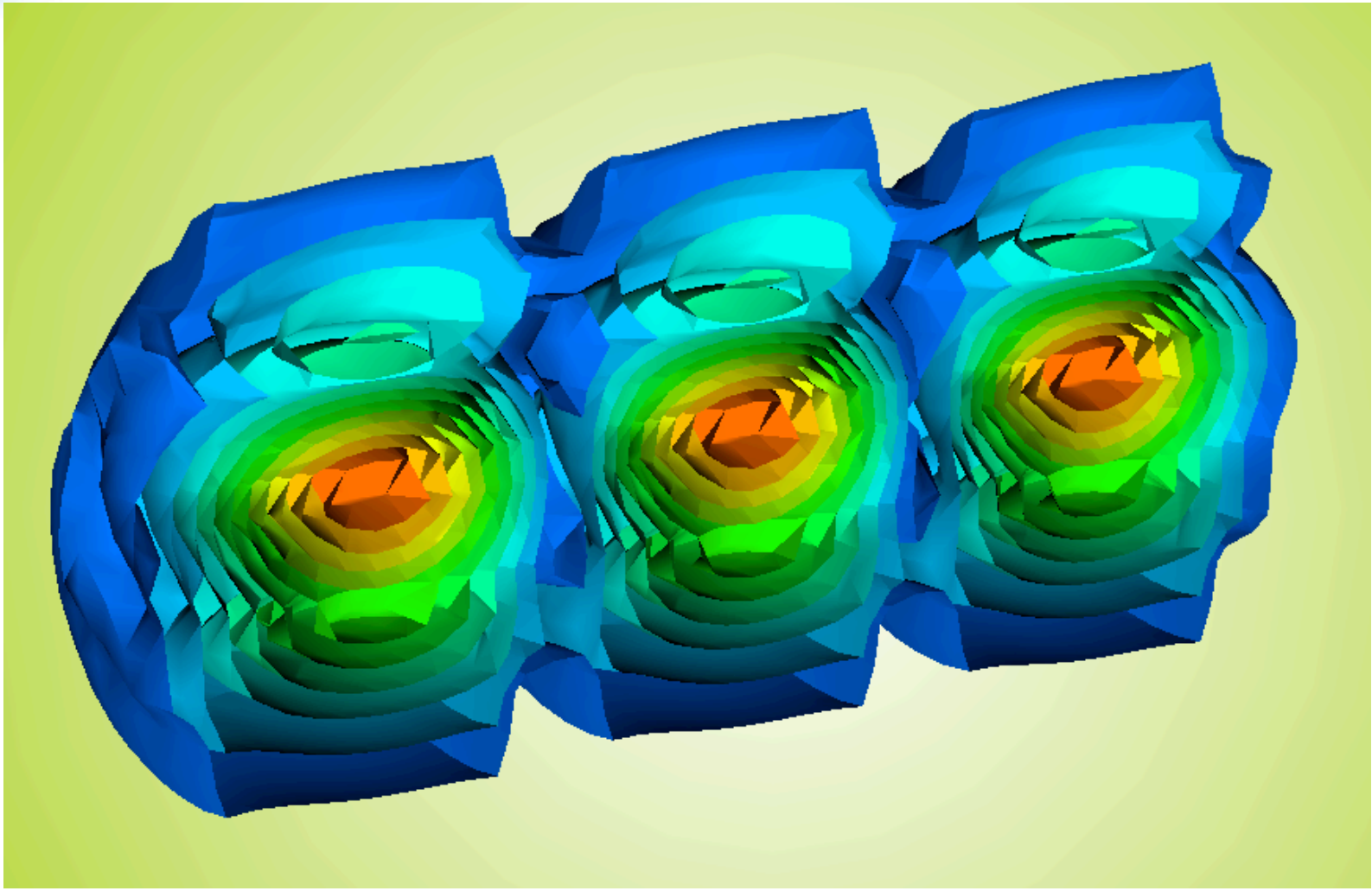




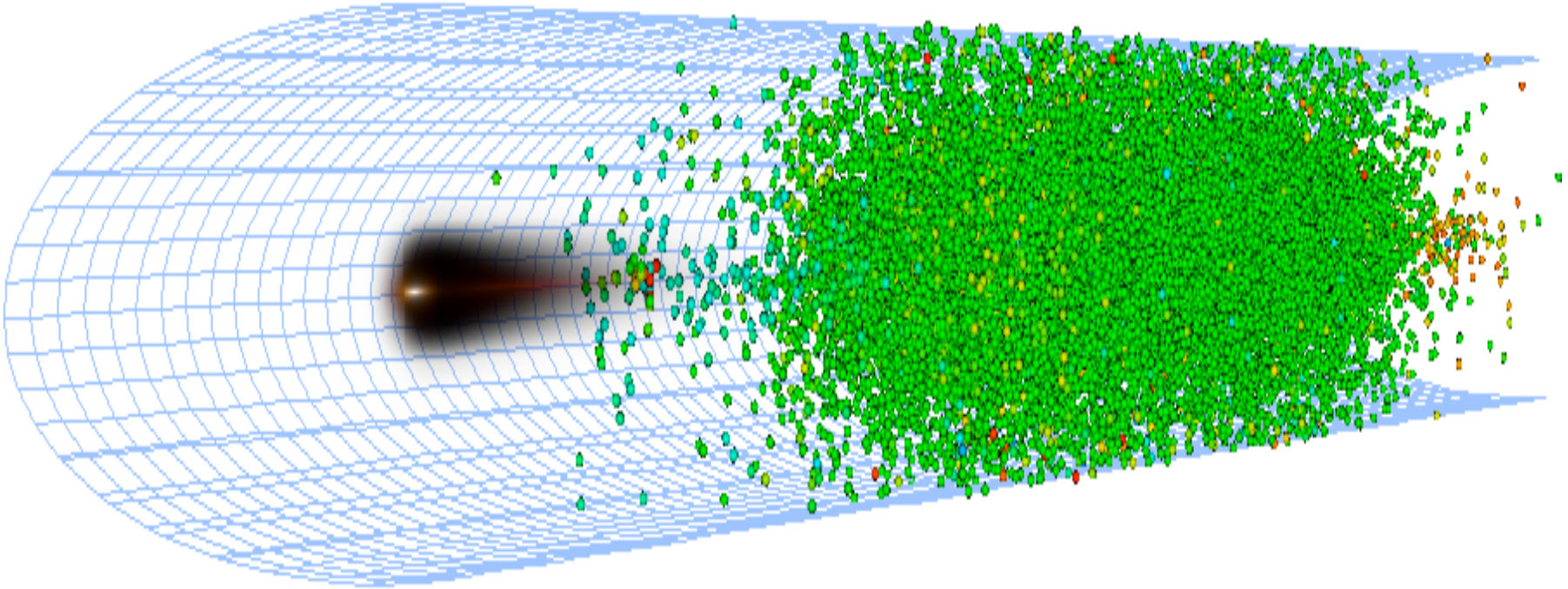
Electric Field (Vector) in A 3 Cell Crab Cavity (VORPAL)



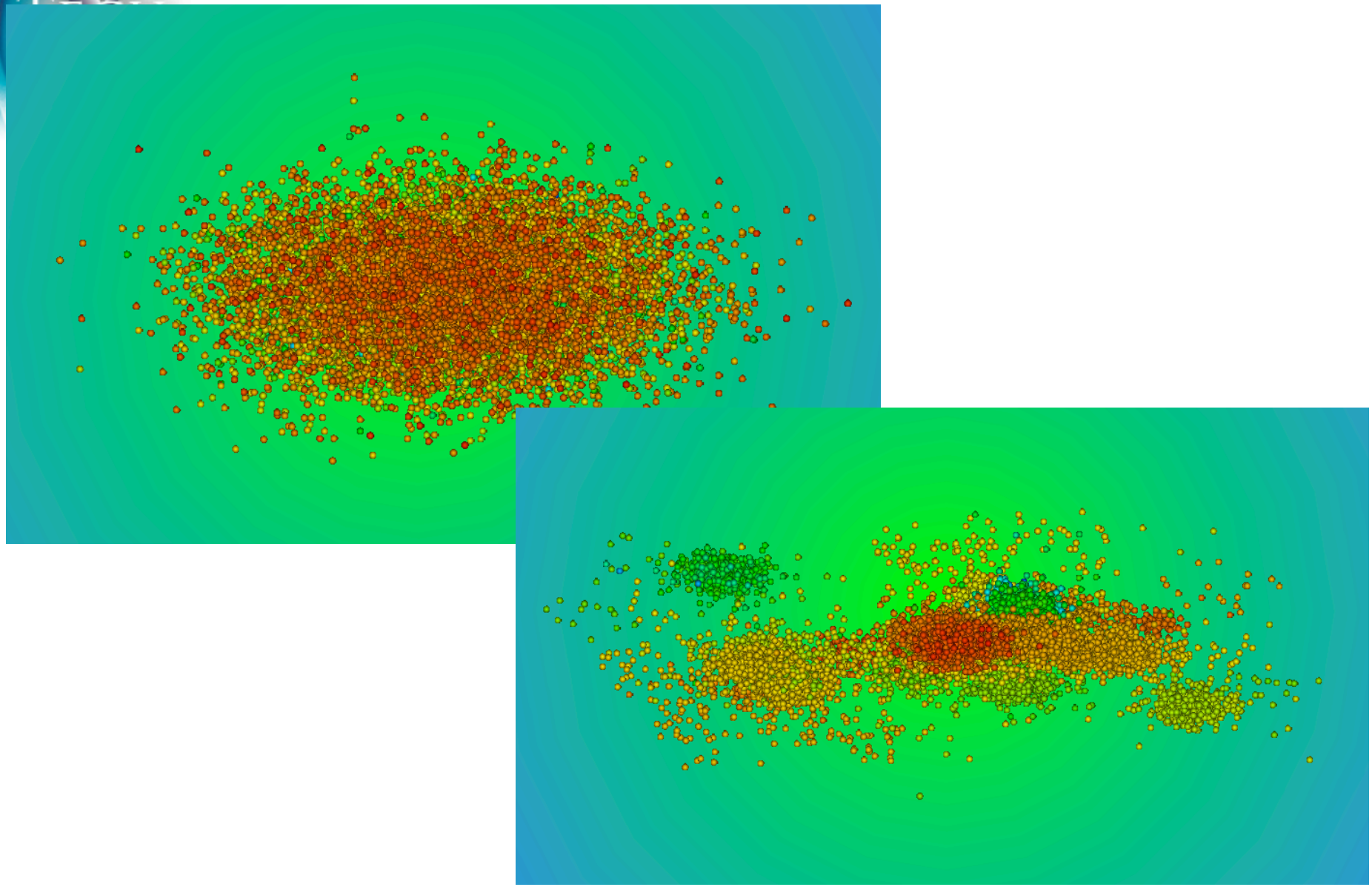
Magnetic Field (isosurfaces) in A Cavity (VORPAL)



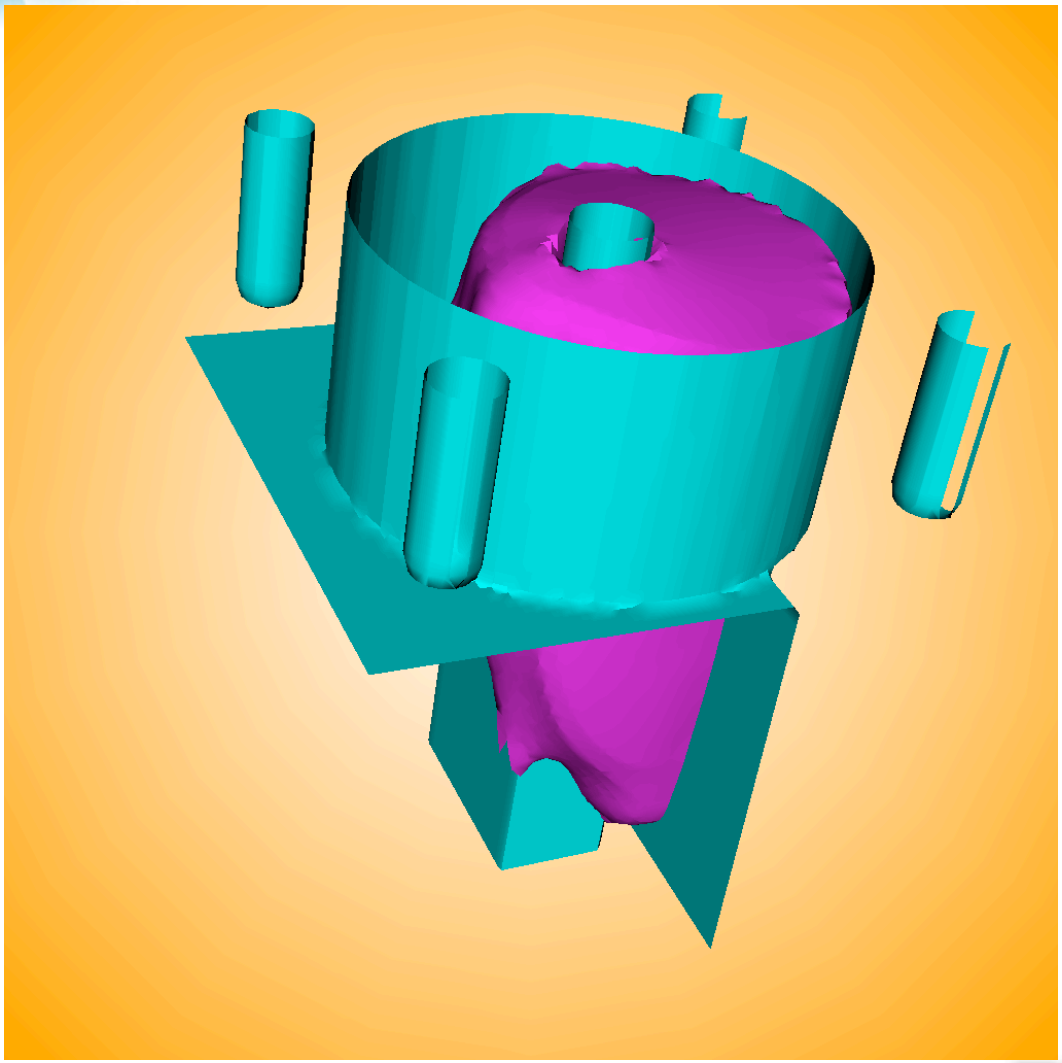
Electron Cloud and Beam Current (VORPAL)



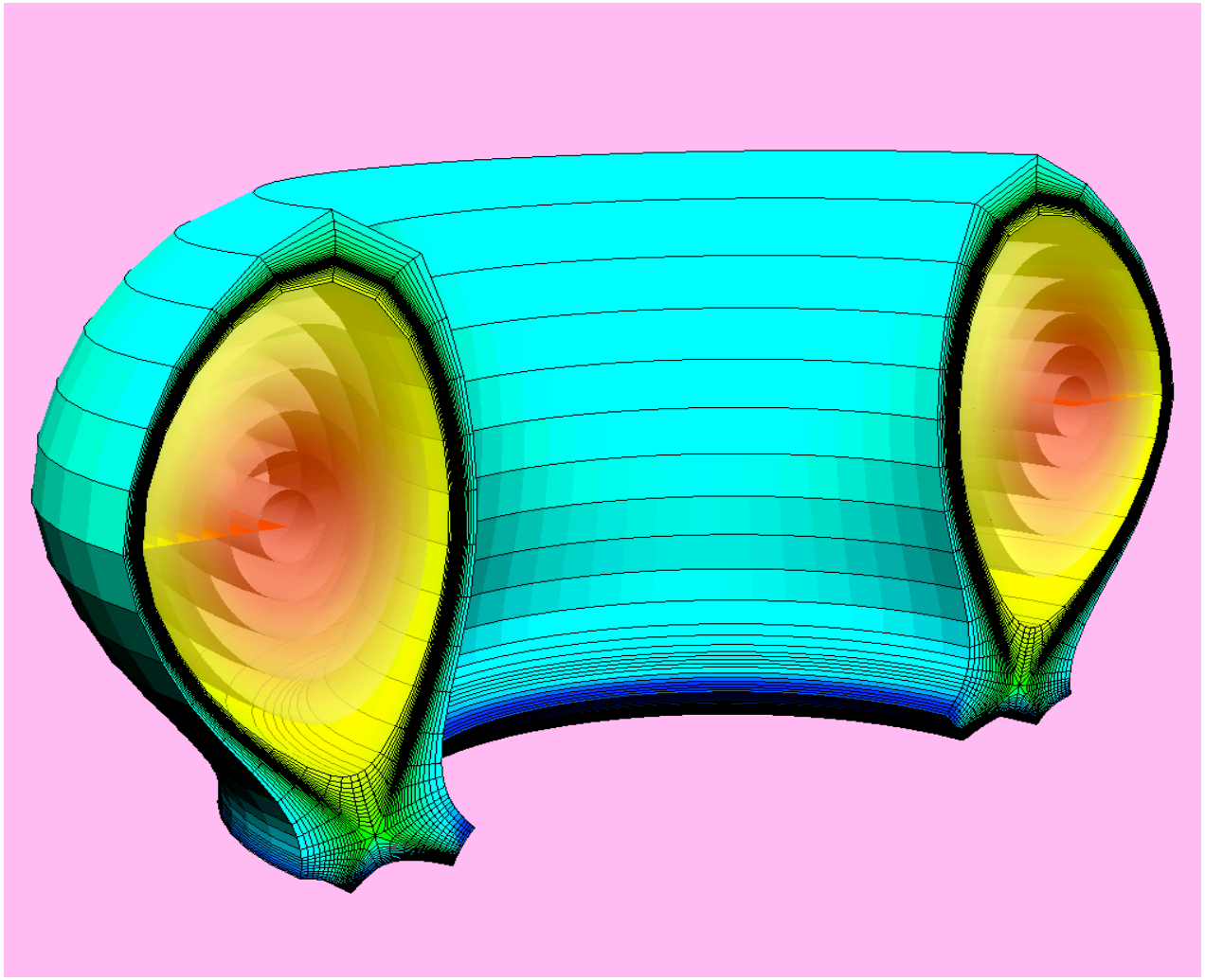
Beam Particles (p**2 in SYNERGIA)



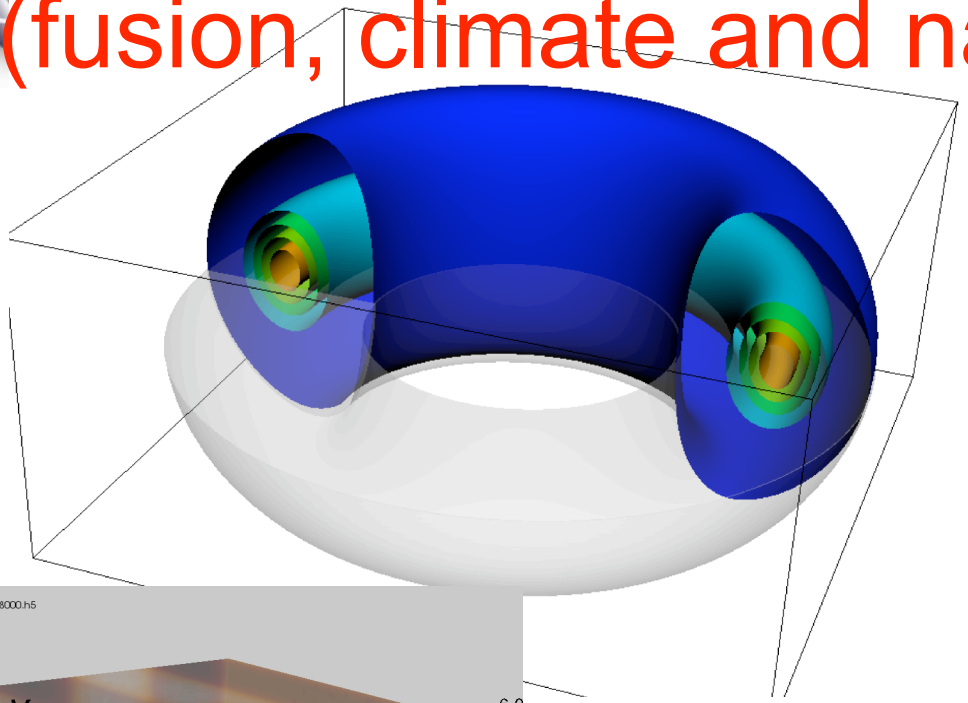
Testing Multipacting For Aerospace Applications (VORPAL)

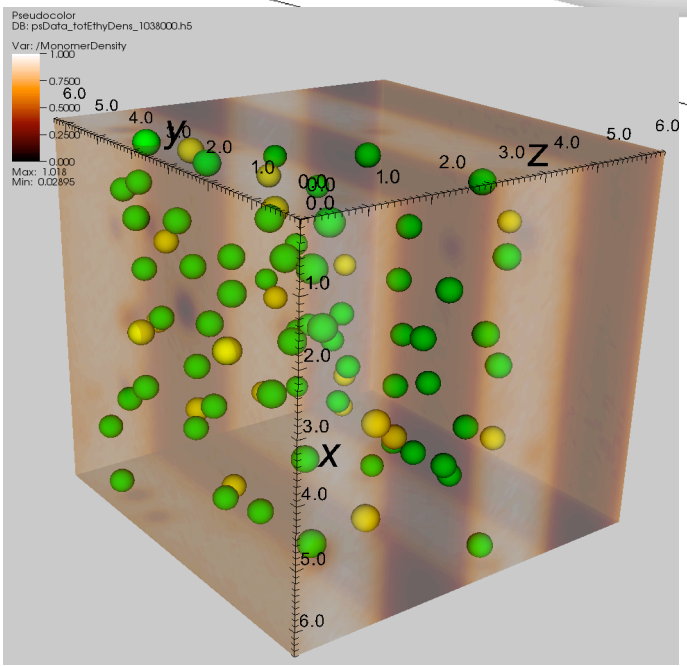


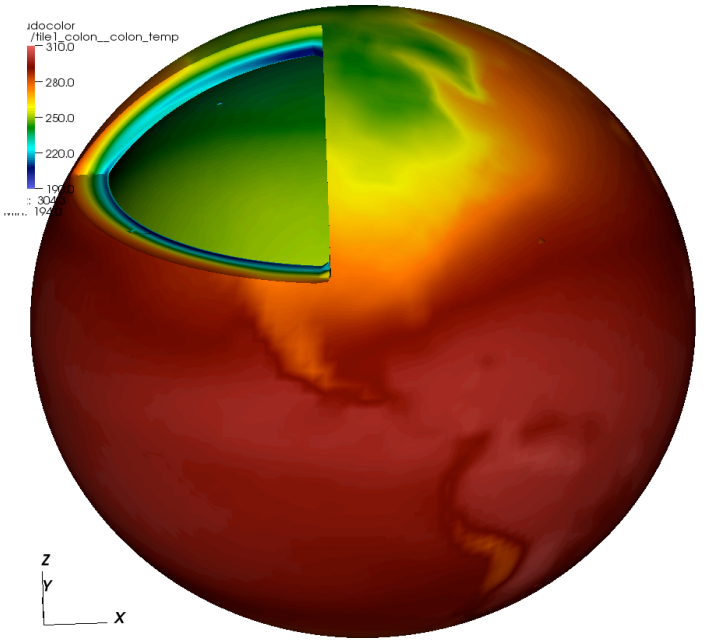
VizSchema Beyond Accelerators (FACETS – fusion)



VizSchema Beyond Accelerators (fusion, climate and nanotechnology)







TECH-X CORPORATION

Future Directions

- Finish parallel plugin (started by G. Weber)
- More codes and updating schema as needed
- VORPAL and FACETS skin
- NetCDF?
- Auto-generation of correct markup from a text or XML document

Information

- VizSchema information and download:

https://ice.txcorp.com/trac/vizschema/wiki/WikiStart

- Soon will be come as one of standard VisIt plugins
- sveta@txcorp.com