New software library of geometrical primitives for modelling of solids used in Monte Carlo detector simulations

Marek Gayer, John Apostolakis, Gabriele Cosmo, Andrei Gheata, Jean-Marie Guyader, Tatiana Nikitina
CERN PH/SFT

The International Conference on Computing in High Energy and Nuclear Physics (CHEP), New York, May 21-25, 2012

Motivations for a common solids library

- Optimize and guarantee better long-term maintenance of Root and Gean4 solids libraries
  - A rough estimation indicates that about 70-80% of code investment for the geometry modeler concerns solids, to guarantee the required precision and efficiency in a huge variety of combinations
- Create a single library of high quality implementations
  - Starting from what exists today in Geant4 and Root
  - Adopt a single type for each shape
  - Create a new Multi-Union solid
  - Aims to replace solid libraries in Geant4 and Root
  - Allowing to reach complete conformance to GDML solids schema
- Create extensive testing suite

Navigation functionality and library services for each solid

- **Performance critical methods:**
  - Location of point either inside, outside or on surface
  - Shortest distance to surface for outside points
  - Shortest distance to surface for inside points
  - Distance to surface for inside points with given direction
  - Distance to surface for outside points with given direction
  - Normal vector for closest surface from given point

- **Additional methods:** Bounding Box, Capacity, Volume, Generating points on surface/edge/inside of solid, creating mesh / polyhedra for visualization

Topics presented next:

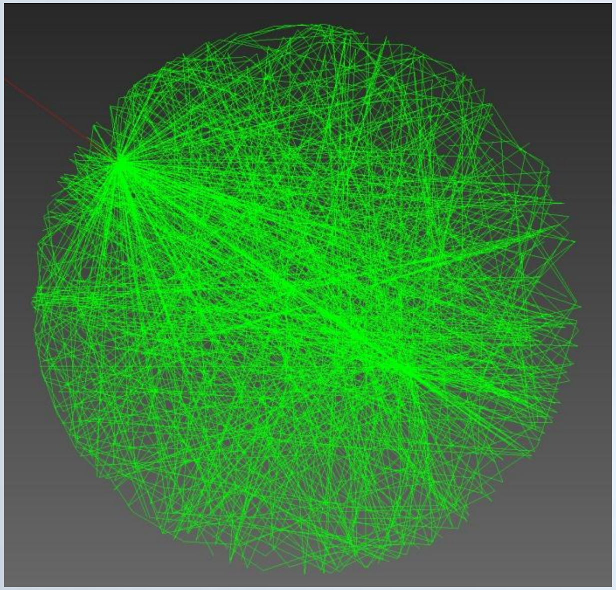
- Testing suite
- New Multi Union Solid

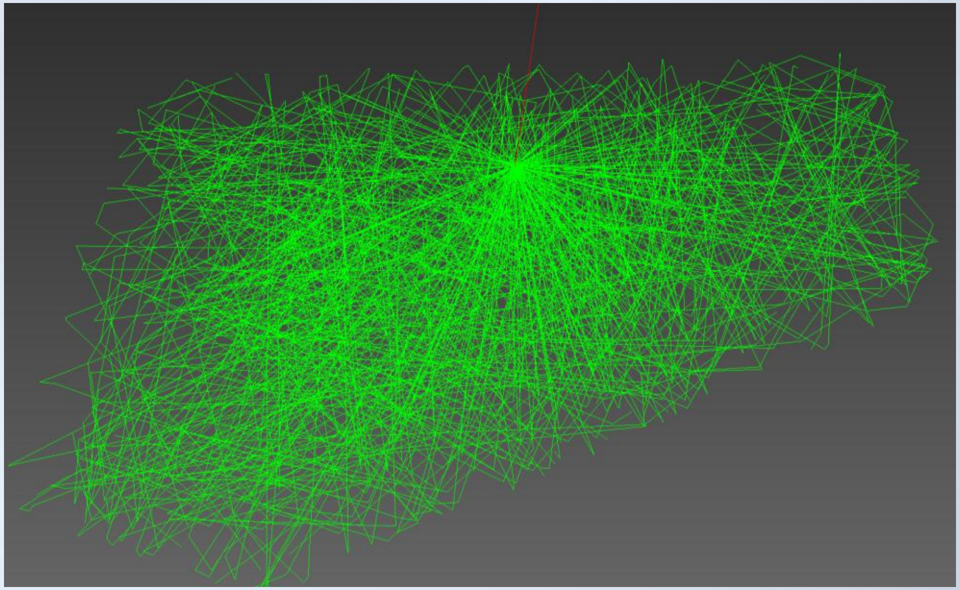
Testing Suite

- Solid Batch Test
- Optical Escape
- Data analysis and performance (SBT DAP)
- Specialized tests (e.g. quick performance scalability test for multi-union)

Optical Escape Test

- Optical photons are generated inside a solid
- Repeatedly bounce on the reflecting inner surface
- Particles must not escape the solid





Solids Batch Test (SBT)

- Points and vectors test
  - Generating groups of inside, outside and surface points
  - Testing all distance methods with numerous checks
    - E.g. for each inside random point *p*, *SafetyFromInside(p)* must be > 0
- Voxels tests
  - Randomly sized voxels with random inside points
- Scriptable application, creates logs
- Extendible C++ framework
  - Allowing easy addition of new tests

Data Analysis and Performance (DAP)

DAP features

- Extension of the SBT framework
- Centred around testing USolids together with existing Geant4 and Root solids
- Values and their differences from different codes can be compared
- Constrain: aim to reach similar or better performance in each method
- The core part of USolids testing
- Portable: Windows, Linux, Mac
- Two phases
  - Sampling phase (generation of data sets, implemented as C++ app.)
  - Analysis phase (data post-processing, implemented as MATLAB scripts)

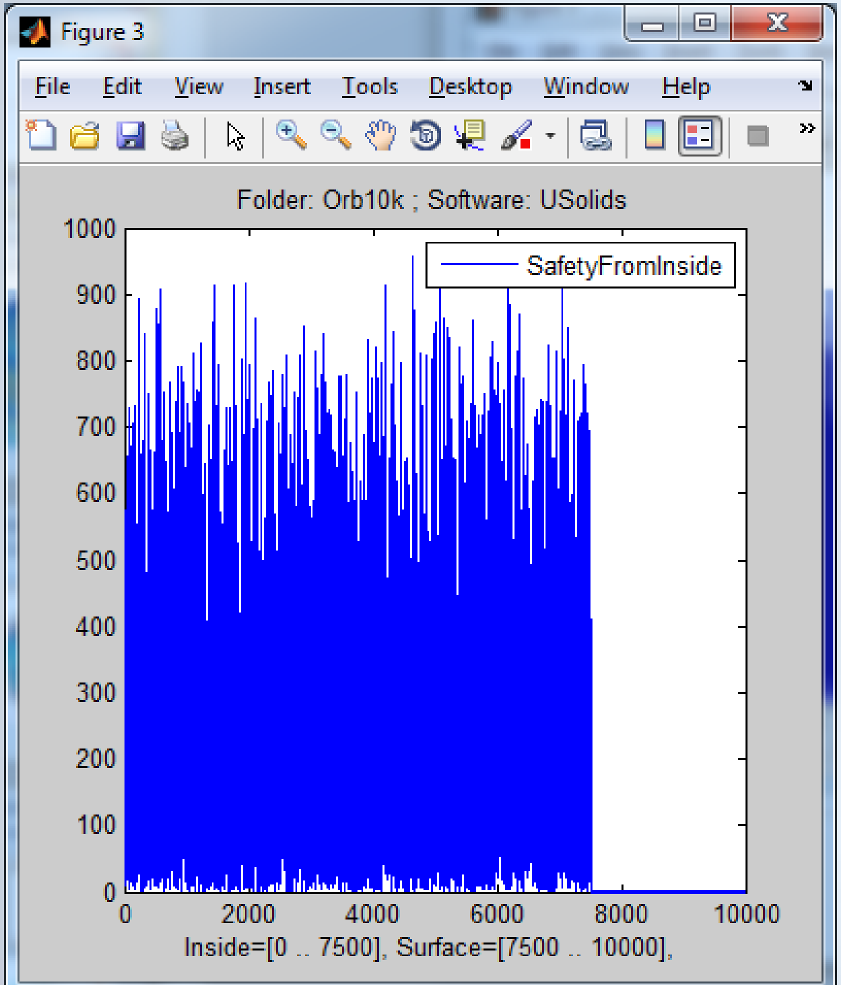
DAP - Sampling phase

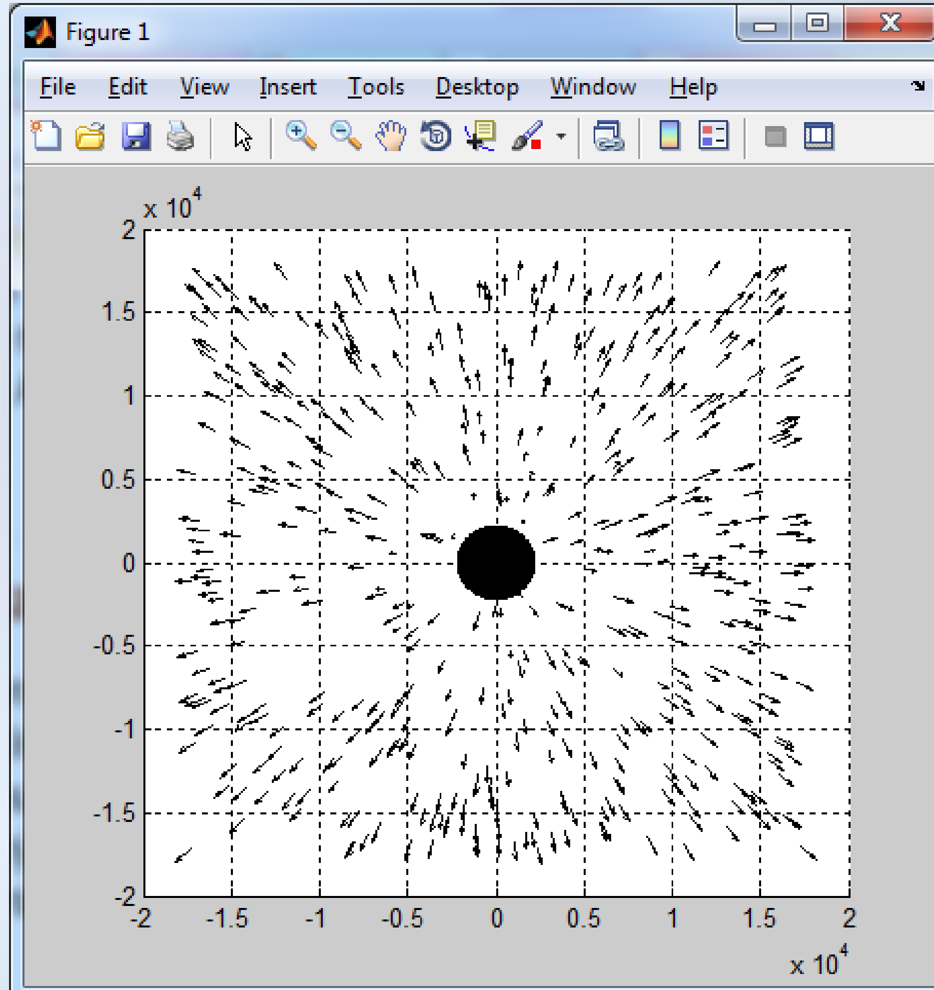
- Tests with solids from three libraries: Geant4, Root and USolids
- Tests with pre-calculated, randomly generated sets of points and vectors
- Storing of results data sets to disk
- Measurement of performance
- Support for batch scripting
  - Detailed configuration of conditions in the tests
  - Invoking several tests sequentially
- Rich debugging possibilities in Visual Studio

DAP - Analysis phase

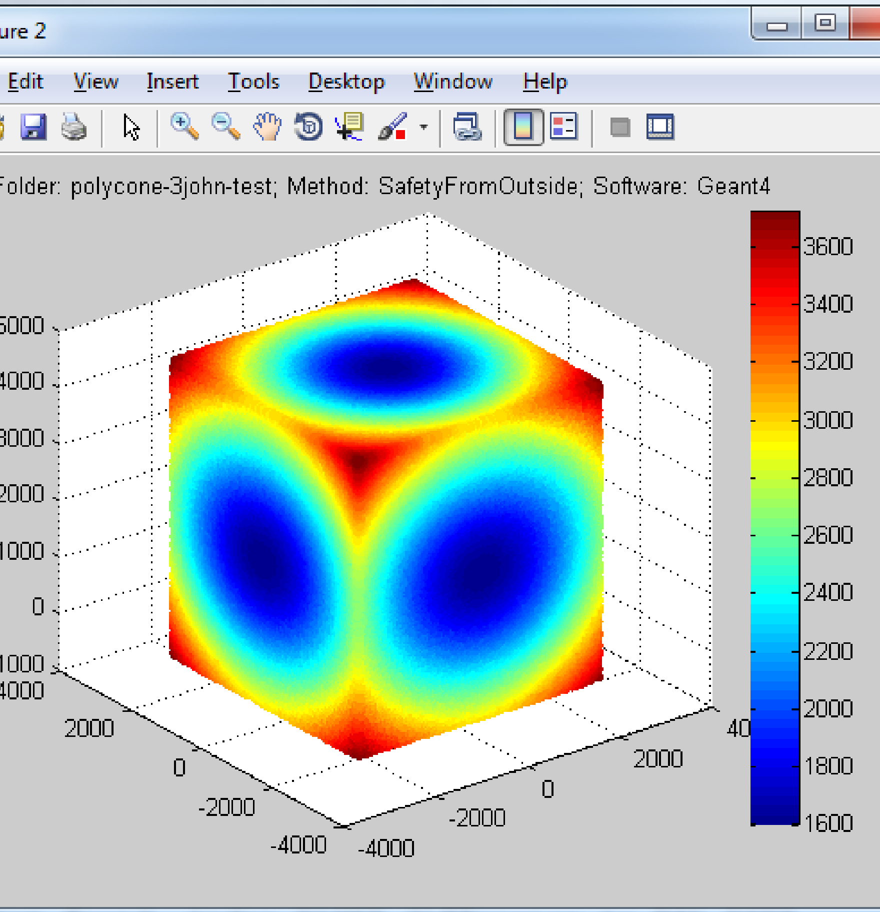
- Visualization of scalar and vector data sets and shapes
- Visual analysis of differences
- Graphs with comparison of performance and scalability
- Inspection of values and differences of data sets

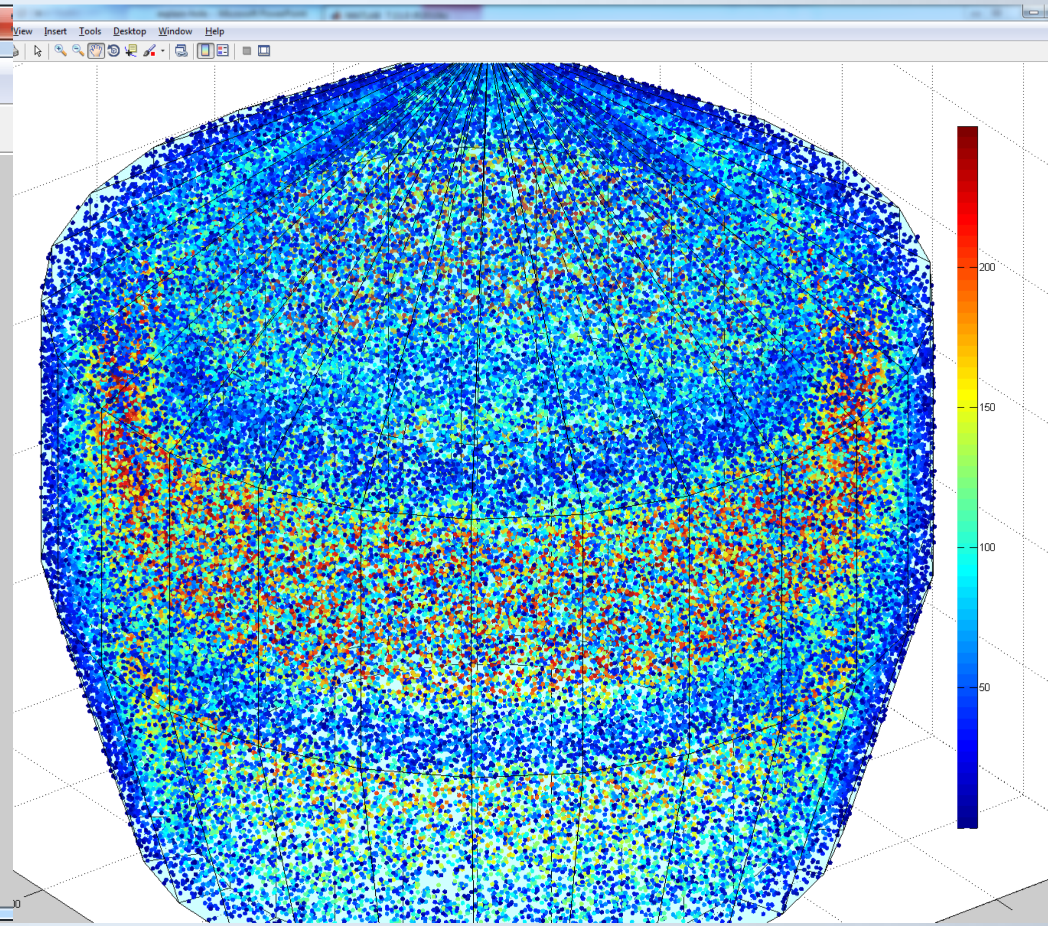
Visualization of scalar and vector data sets



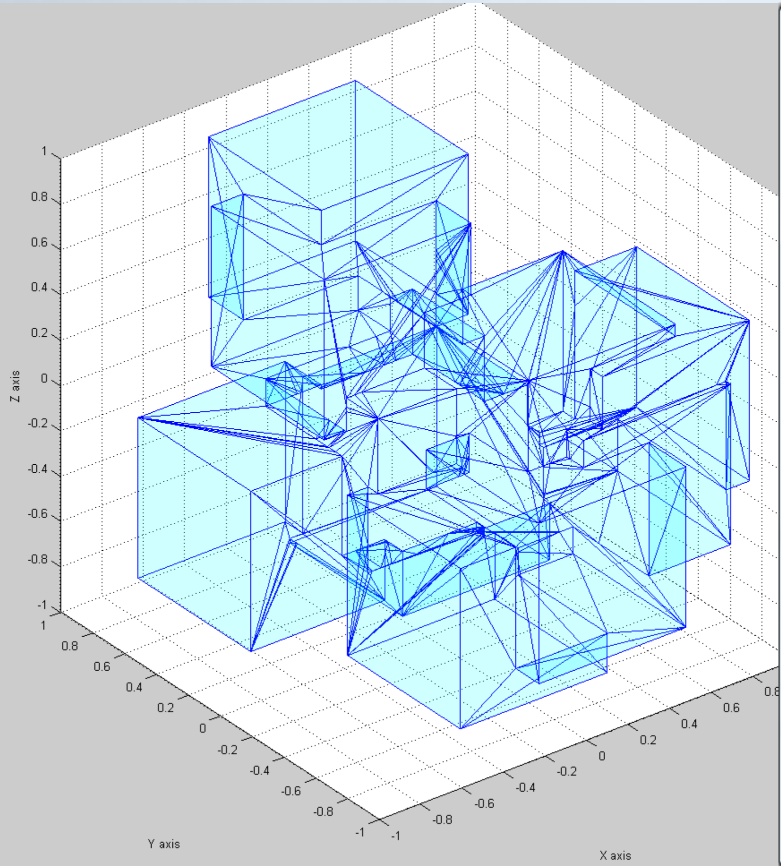


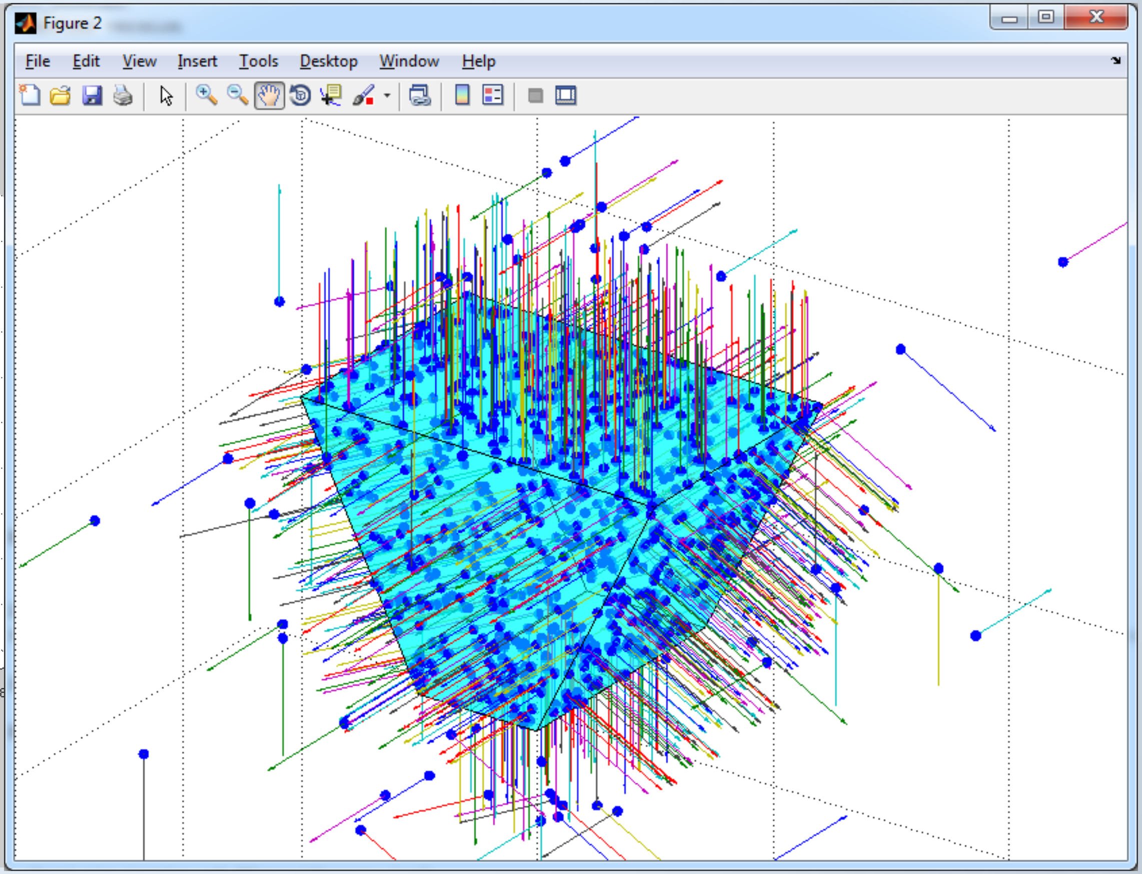
3D plots allowing to overview data sets



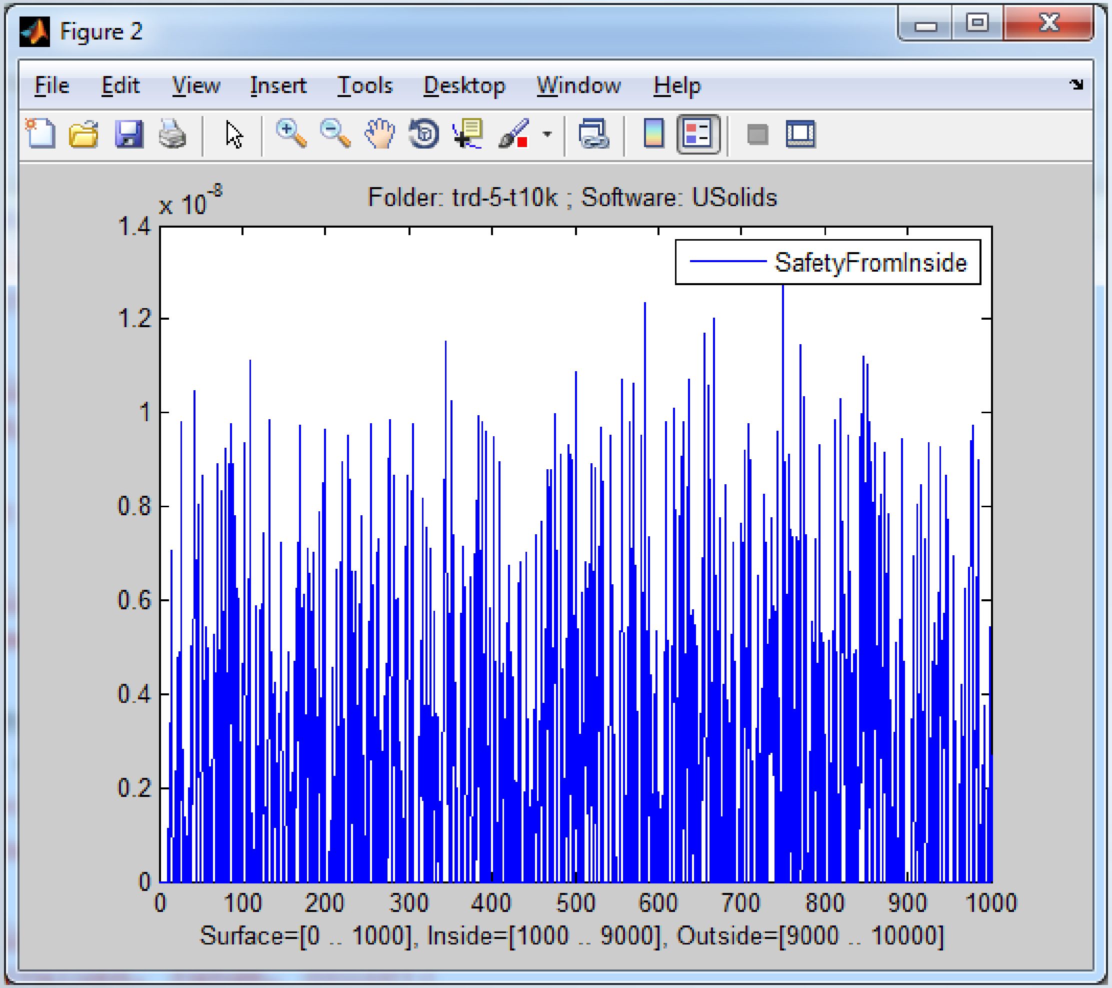


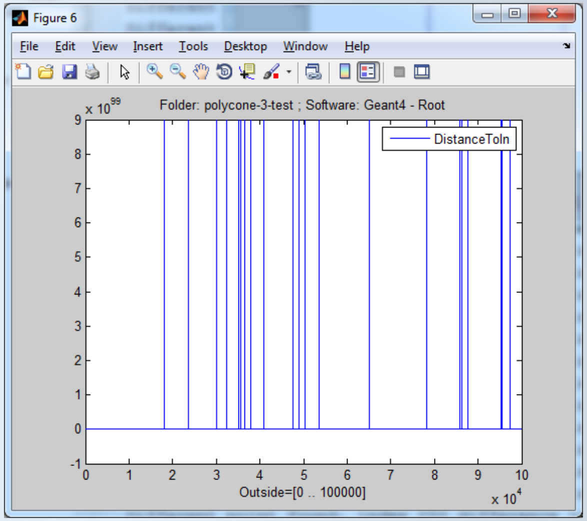
3D visualization of investigated shapes

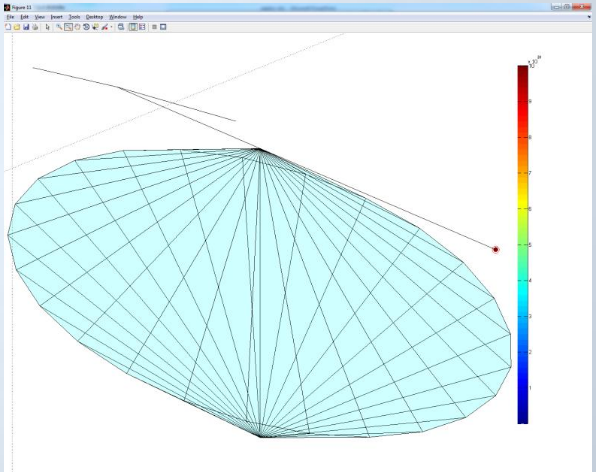




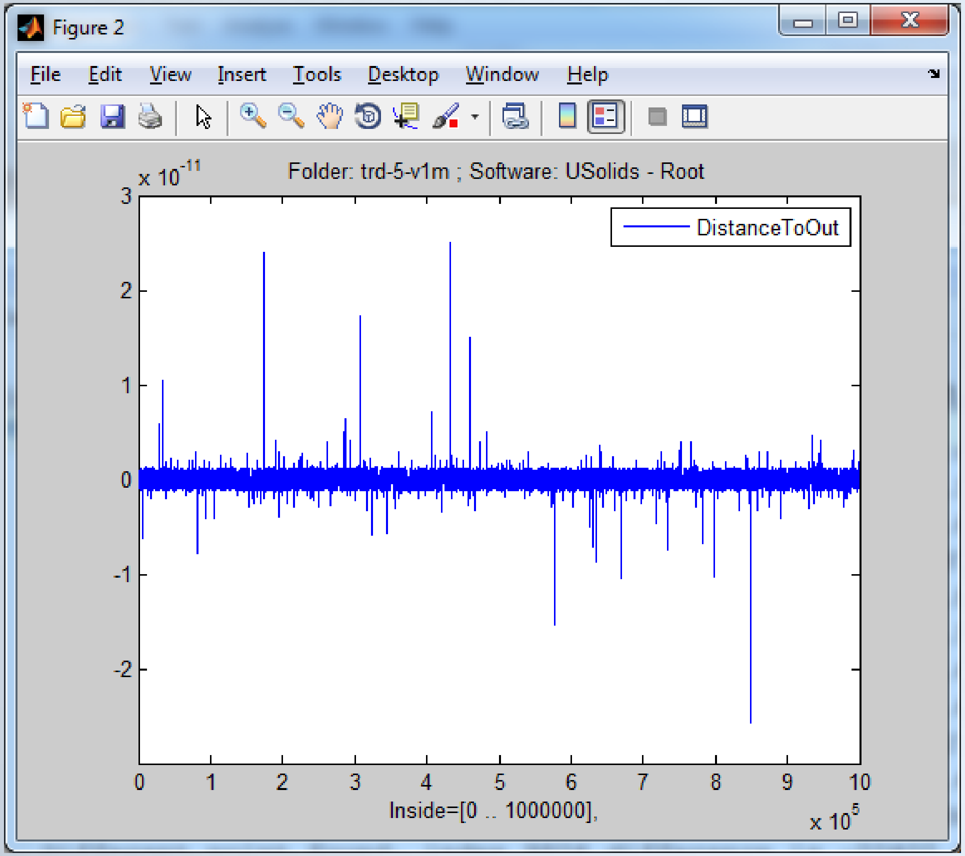
Support for regions of data, focusing on sub-parts

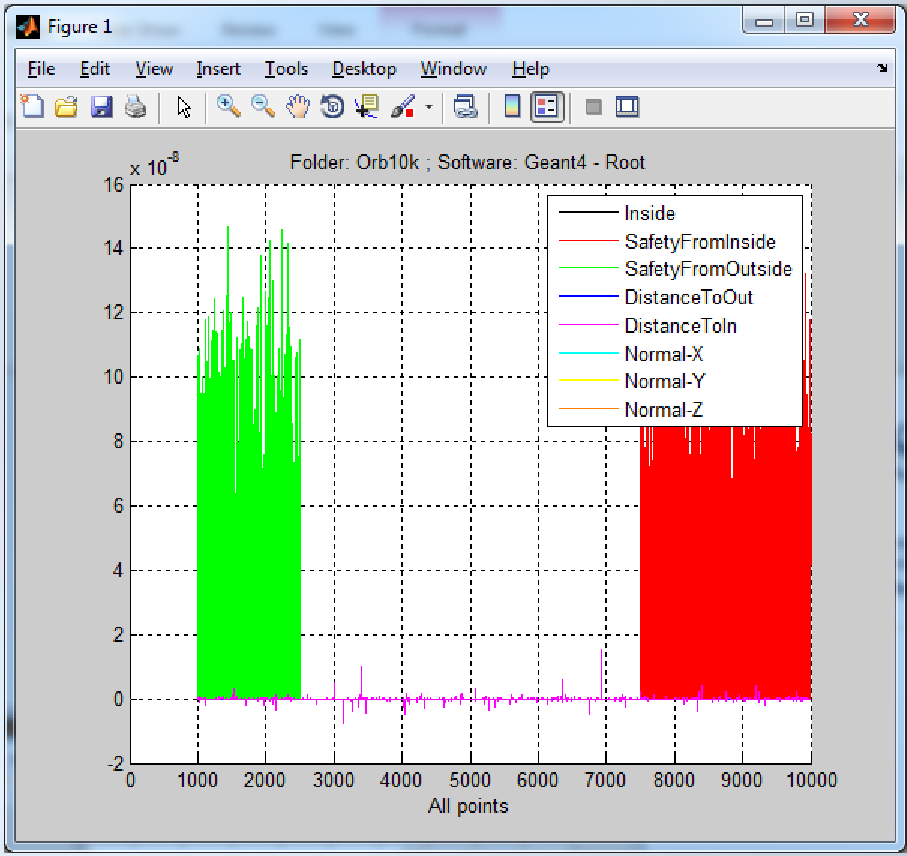




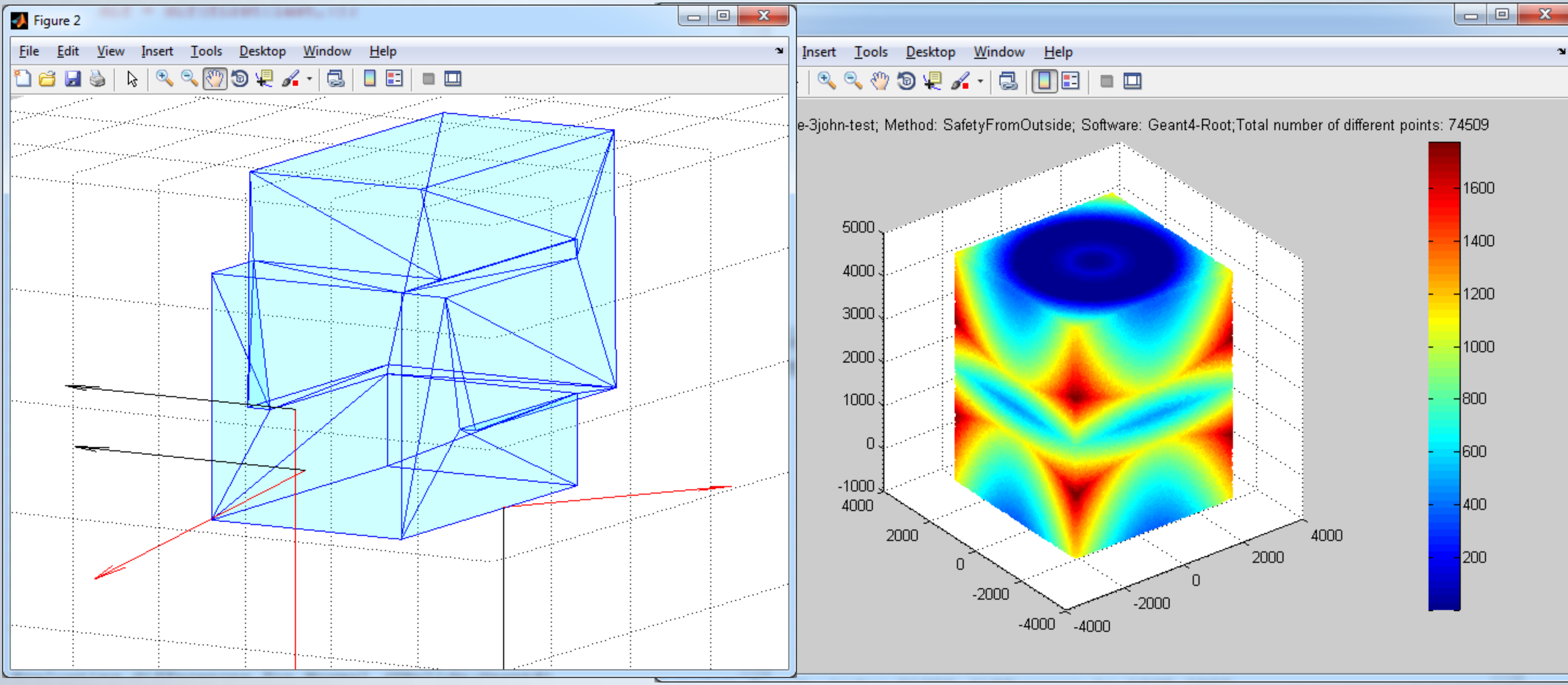


Visual analysis of differences

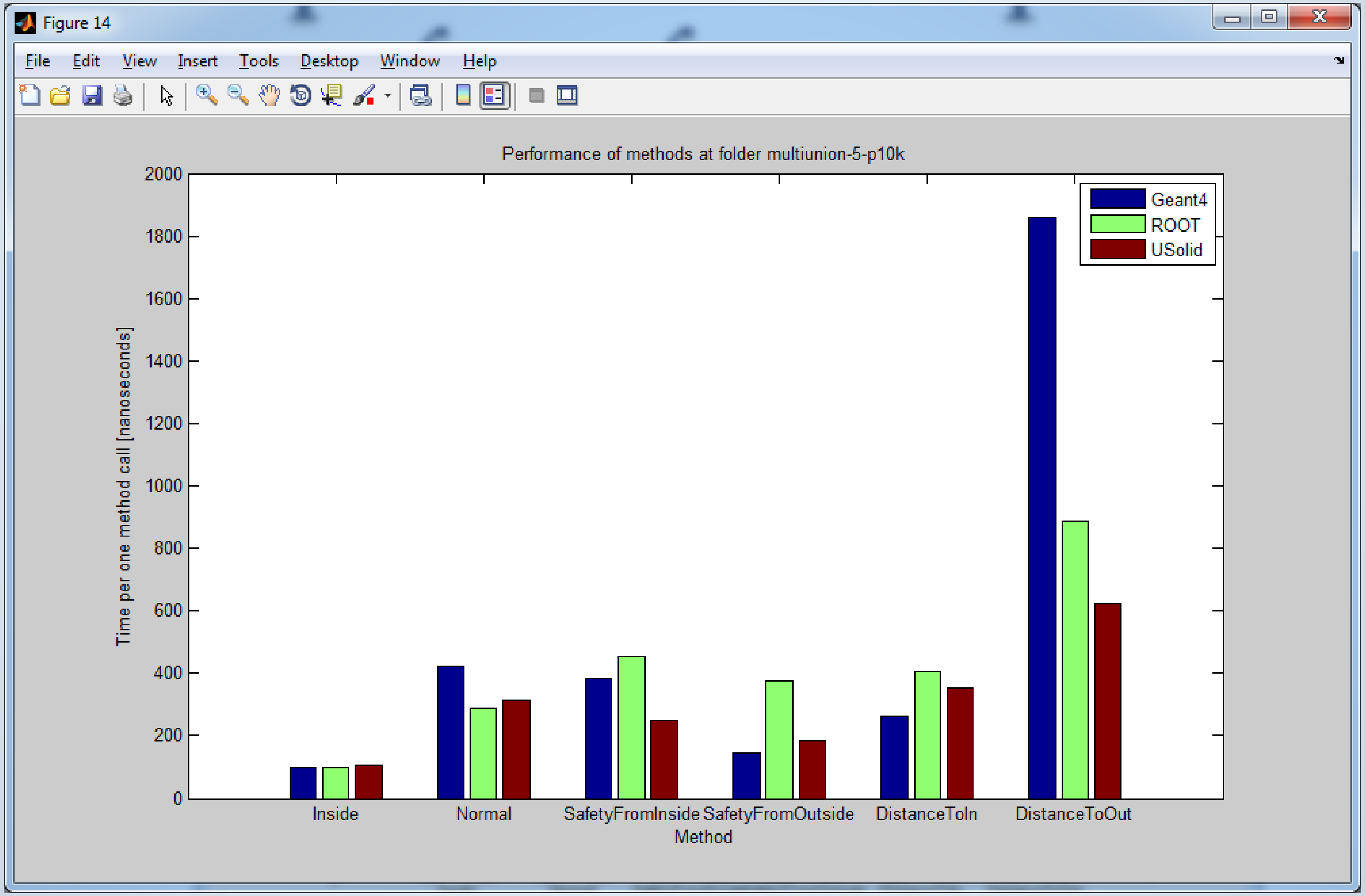




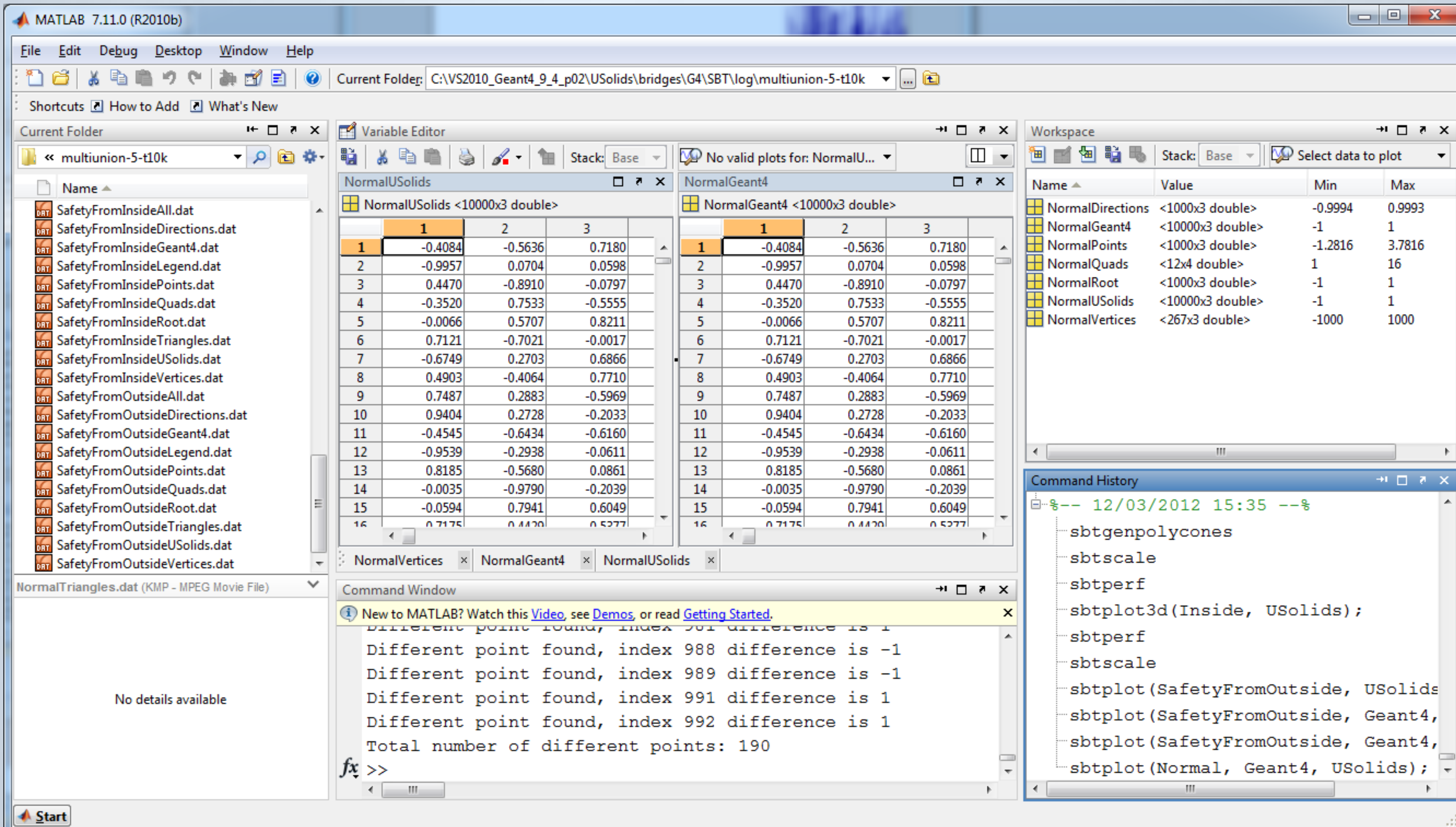
Visual analysis of differences in 3D



Graphs with comparison of performance



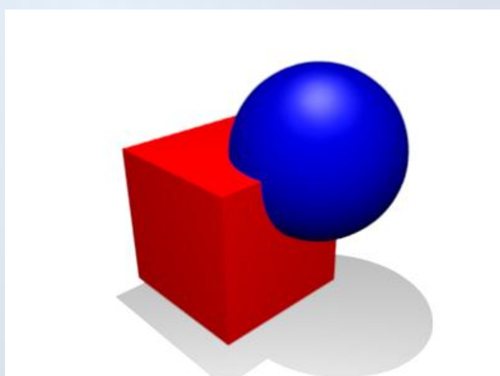
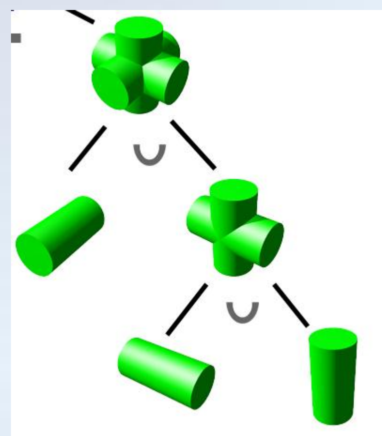
Inspection of values and differences of scalar and vector data sets



New Multi-Union solid

Boolean Union solids

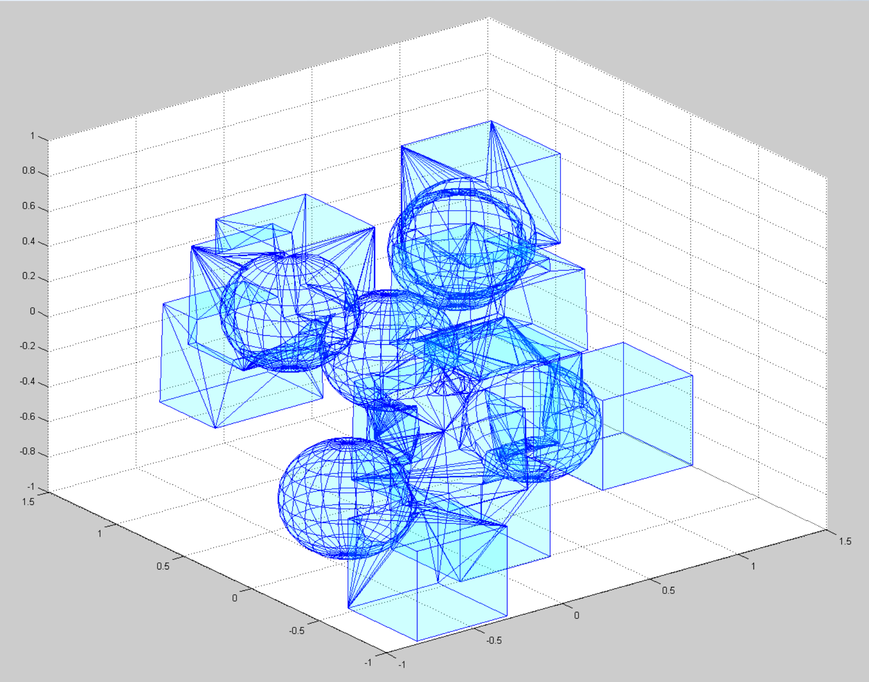
- Existing CSG Boolean solids (Root and Geant4) represented as binary trees
  - To solve navigation requests, most of the solids composing a complex one have to be checked
  - Scalability is typically linear => low performance for solids composed of many parts

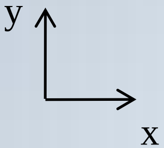
Boolean Union solid:
is composite of two solids, either primitive or Boolean

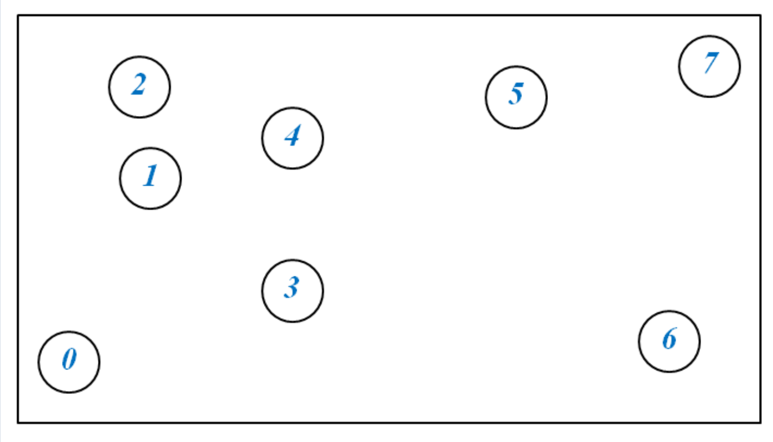
Multi-Union solid

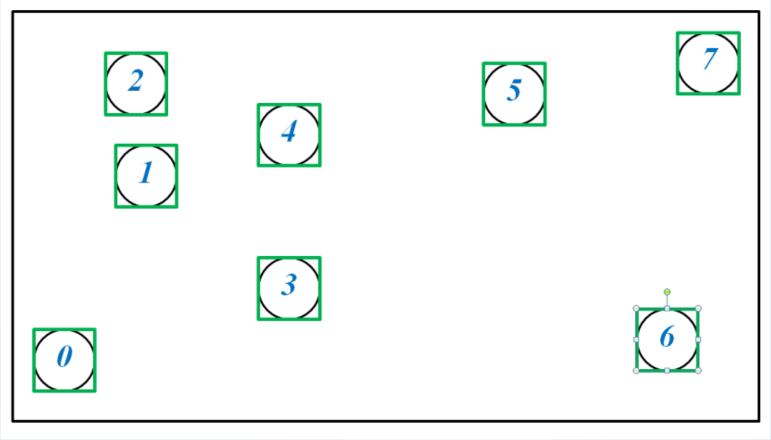
- We implemented a new solid as a union of many solids using voxelization technique to optimize the speed
  - 3D space partition for fast localization of components
  - Aiming for a log(n) scalability
- Useful also for several complex composites made of many solids with regular patterns

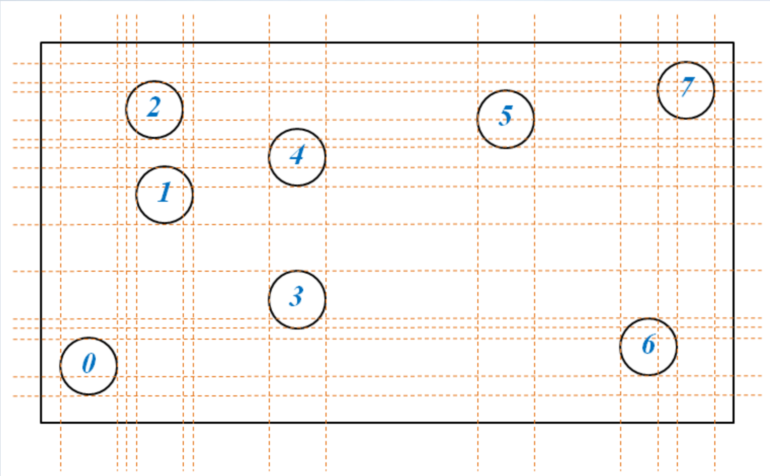


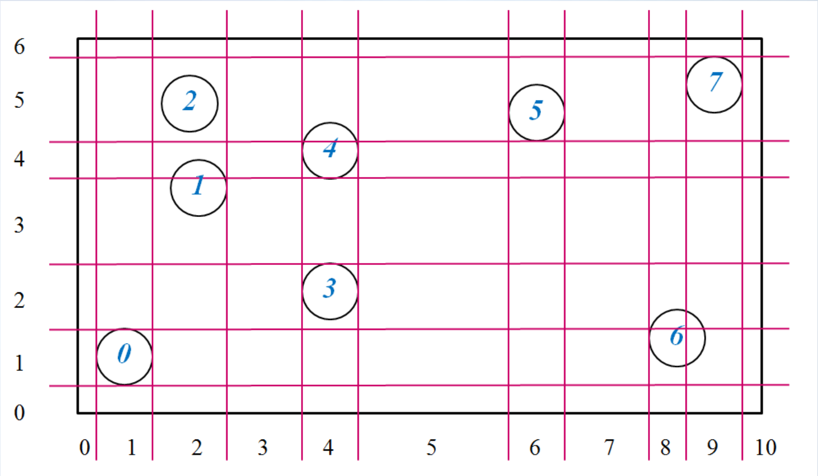
1. Create voxel space (2D simplification)



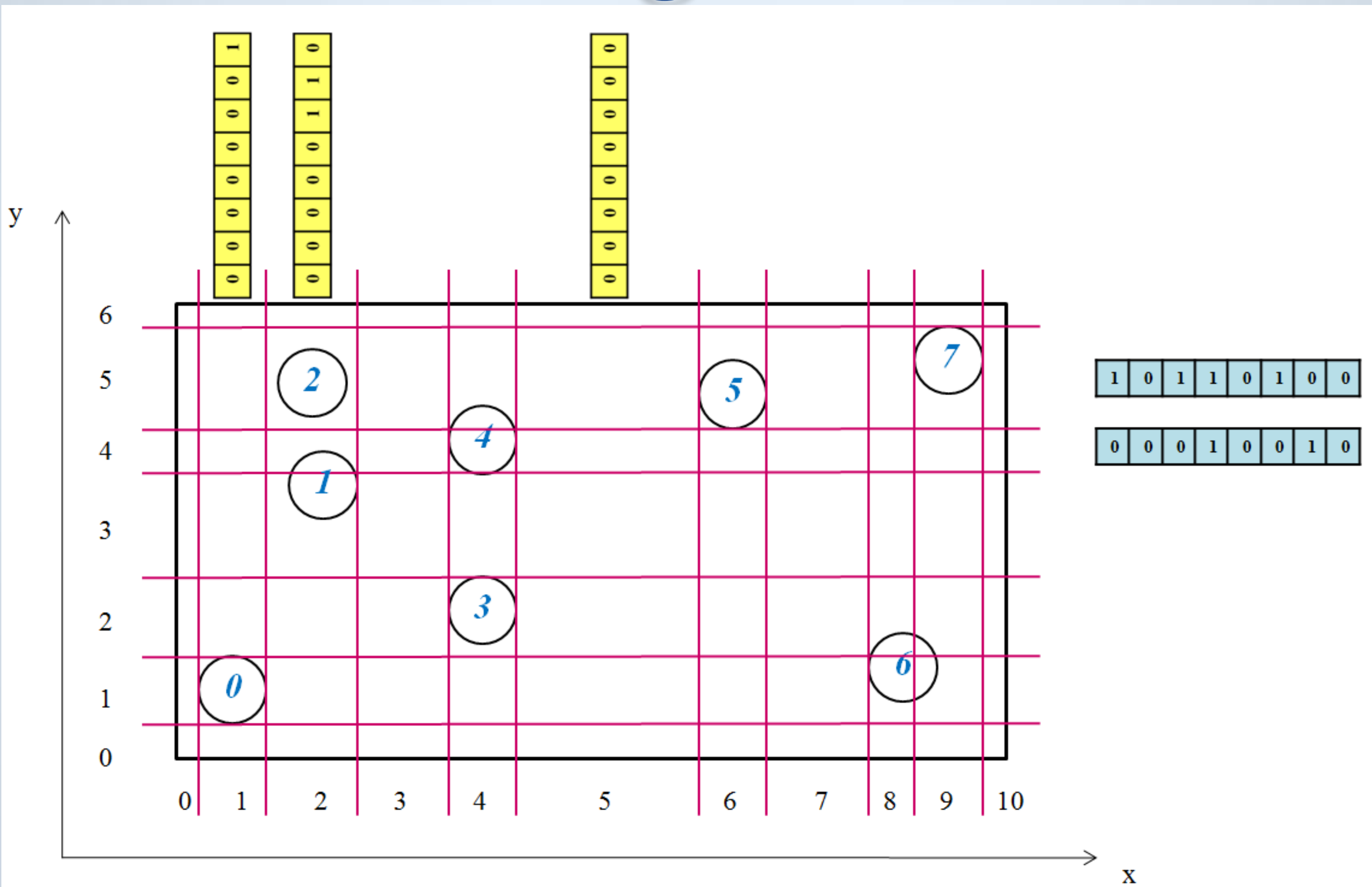




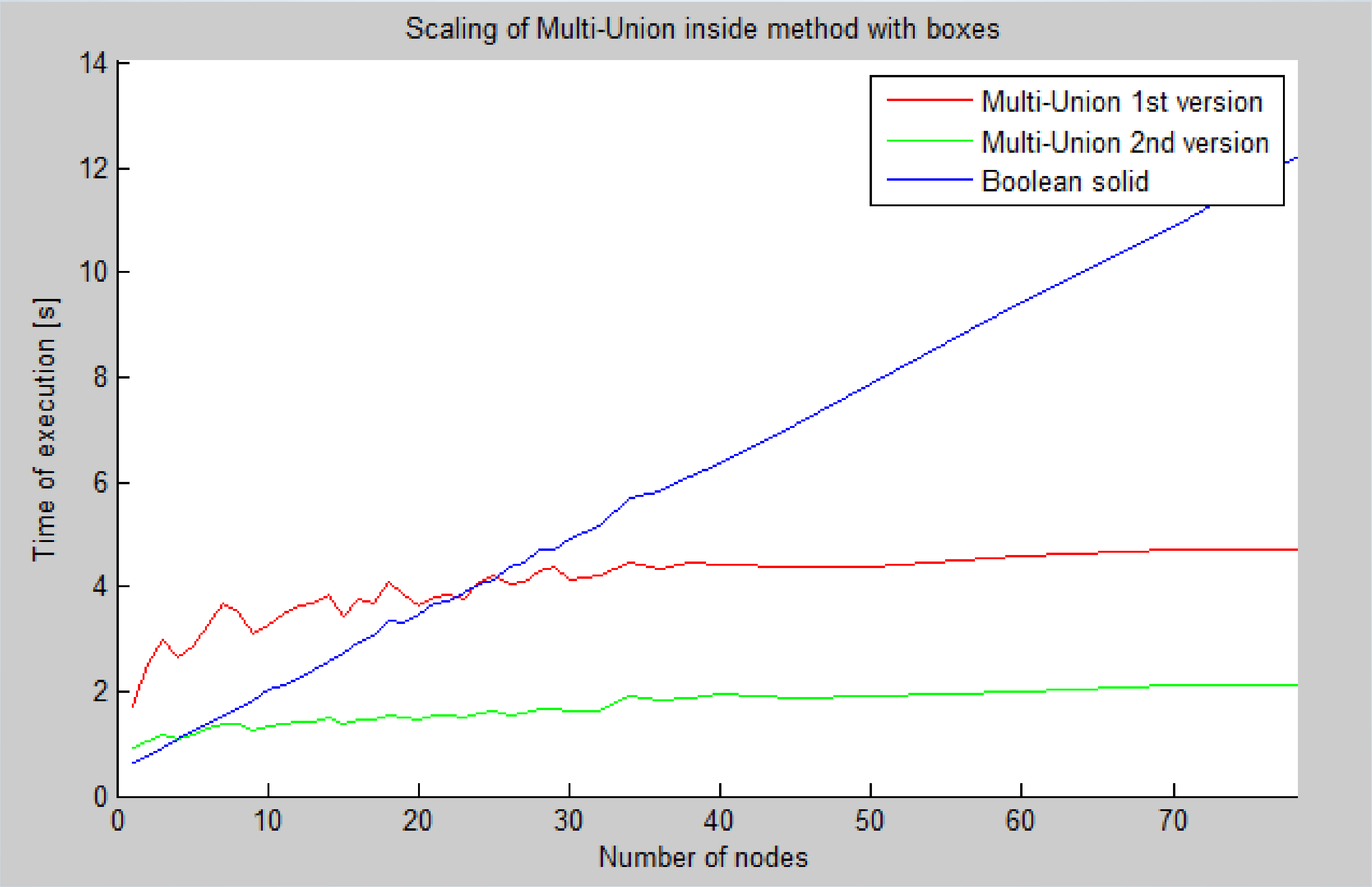




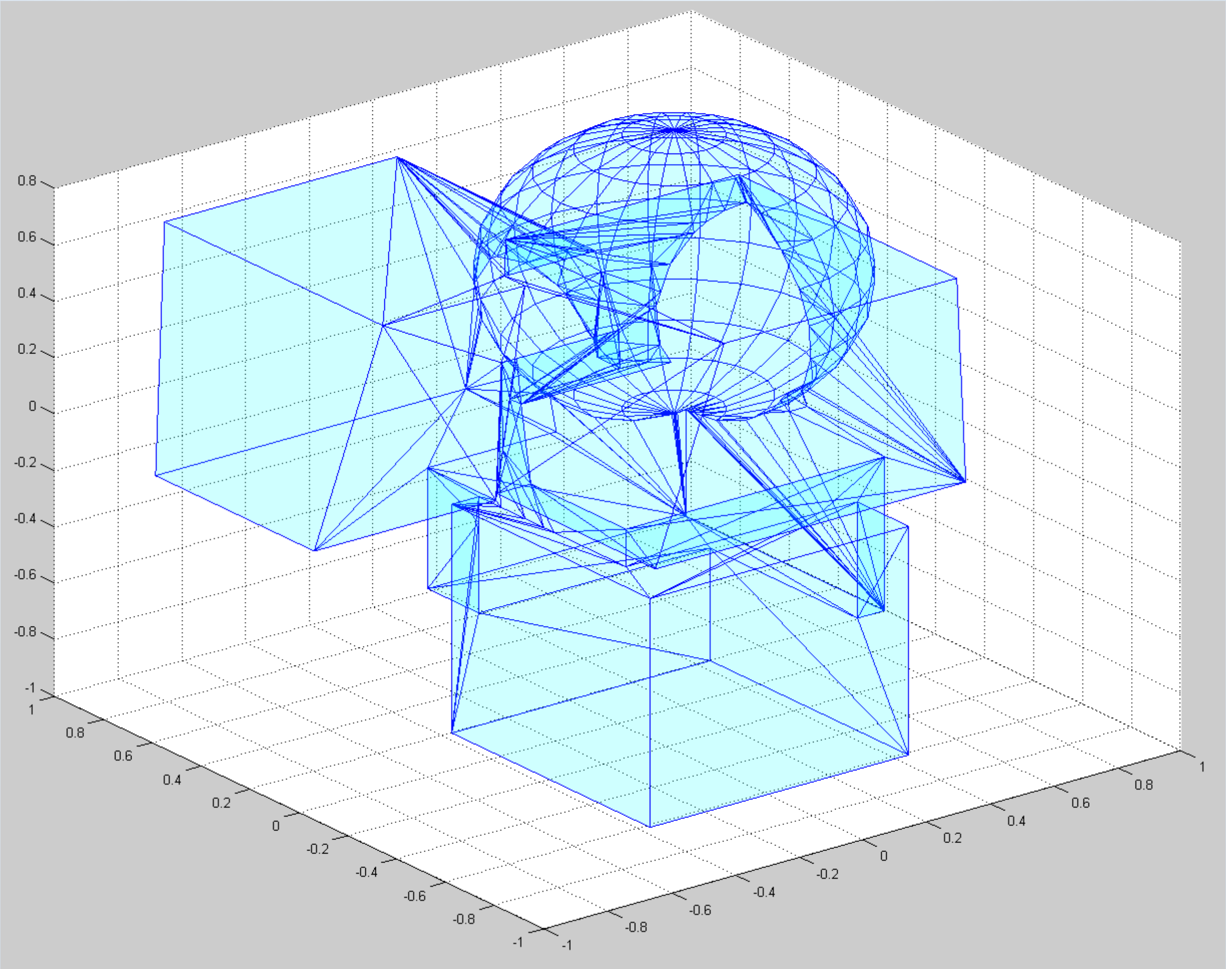
2. Usage of bit masks for storing voxels



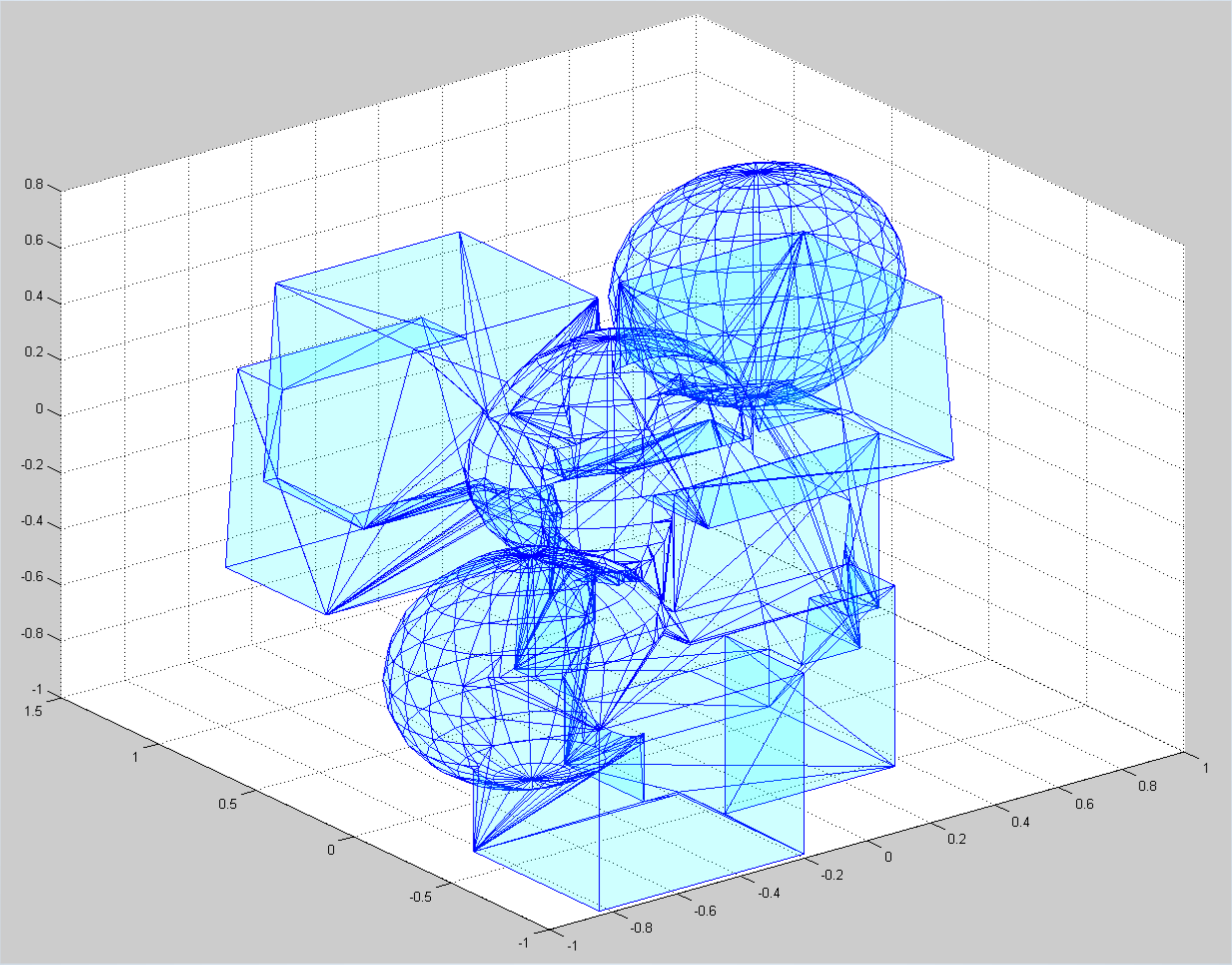
Scaling of Multi-Union vs. Boolean solid



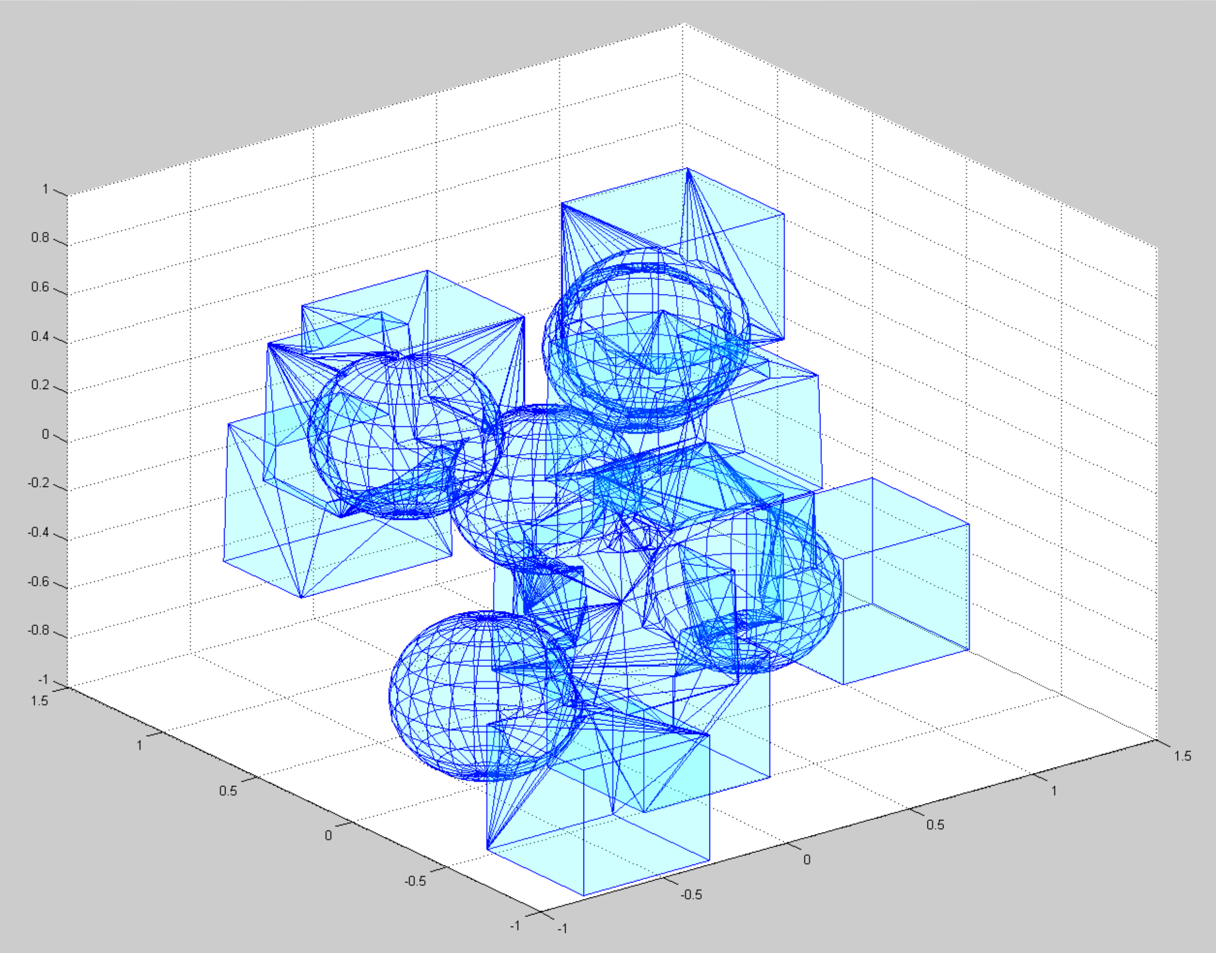
Test union solids for scalability measurements



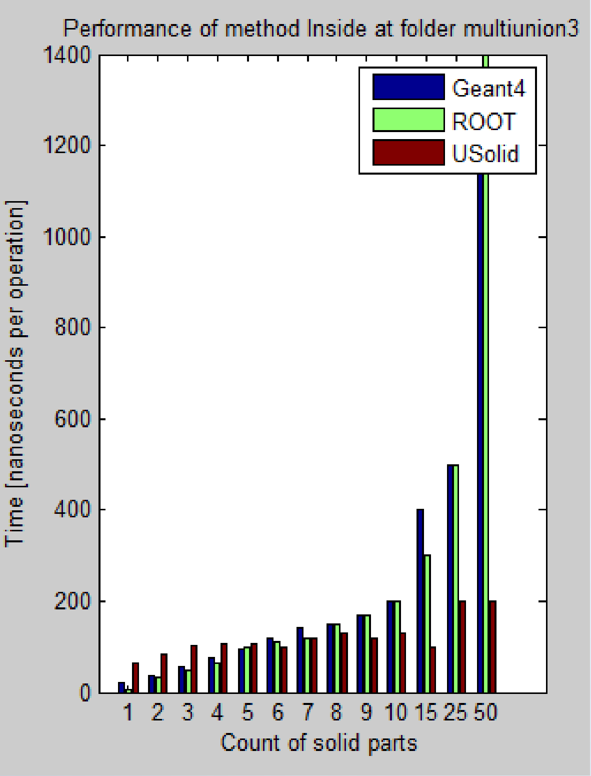
Test union solids for scalability measurements

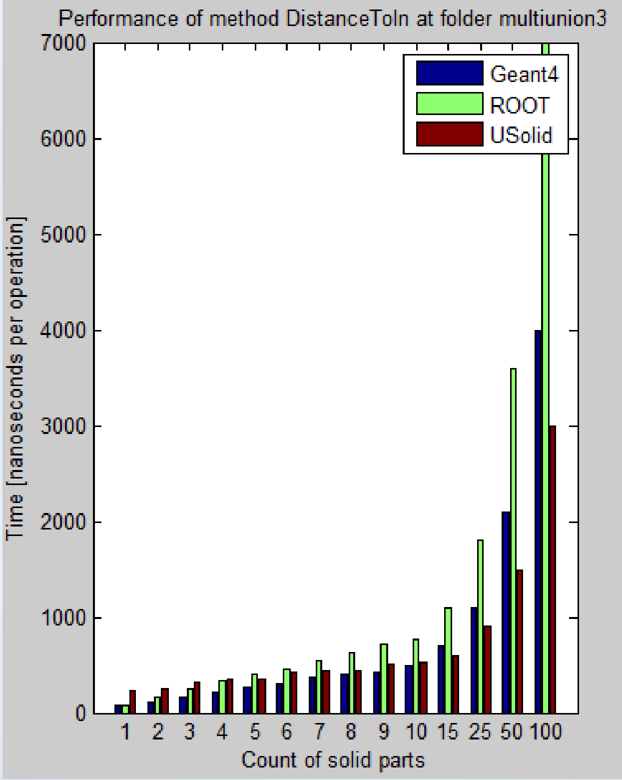


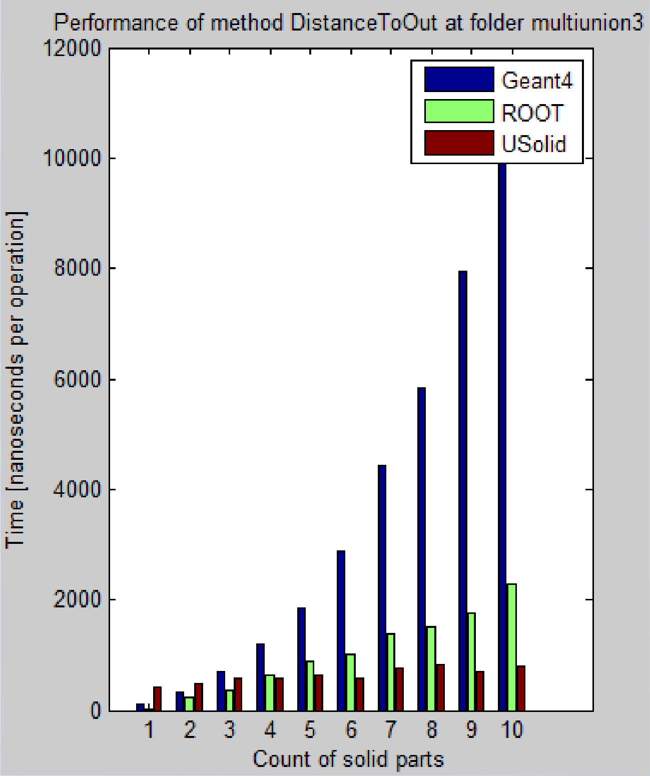
Test union solids for scalability measurements



The most performance critical methods







Status of work

- ✓ Types and USolid interface are defined
- ✓ Bridge classes defined and implemented for both Geant4 and Root
- ✓ Testing suite defined and deployed
- ✓ Implementation of Multi-Union solid completed and performance optimized
- ✓ Started implementation of primitives:
  - ✓ First implementation of Box, Orb (simple full sphere) and Trd (simple trapezoid)

Future work

- Give priority to the most critical solids and those where room for improvement can be easily identified
- Systematically analyze and implement remaining solids in the new library

Thank you for your attention.

???

Questions ?