Scientific Visualization Suites: VisIt

2019 International HPC Summer School

Marcelo Ponce

July 2019 - Kobe, Japan

Outline

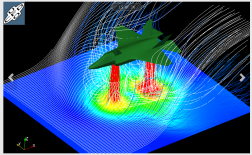
1. Scientific Visualization
   - 2D/3D Visualization Generalities
   - Visualization Pipeline
2. VisIt
   - Generalities
   - Viz. Pipelines Basics
   - Movies Generation
   - Remote Visualization
   - Scripting
3. Summary
4. Further Resources
   - References

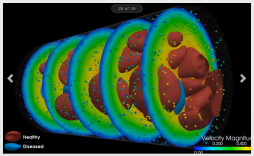
1. Scientific Visualization
   - 2D/3D Visualization Generalities
   - Visualization Pipeline
2. VisIt
   - Generalities
   - Viz. Pipelines Basics
   - Movies Generation
   - Remote Visualization
   - Scripting
3. Summary
4. Further Resources
   - References

Outline

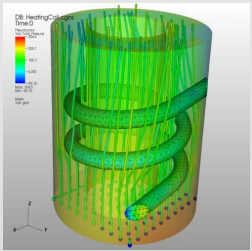
1. Scientific Visualization
   - 2D/3D Visualization Generalities
   - Visualization Pipeline
2. VisIt
   - Generalities
   - Viz. Pipelines Basics
   - Movies Generation
   - Remote Visualization
   - Scripting
3. Summary
4. Further Resources
   - References

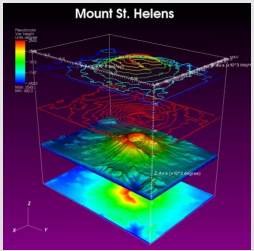
- Visualization is the process of mapping scientific data into "*visual form*"
- Much easier to understand images than a large set of numbers
- For interactive data exploration, debugging, communication with peers





ParaView

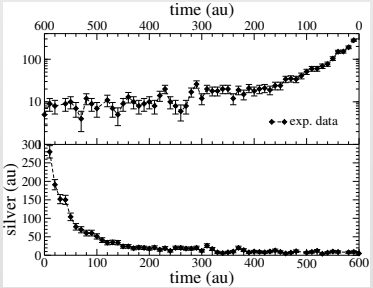




1D plotting vs. 2D/3D visualization

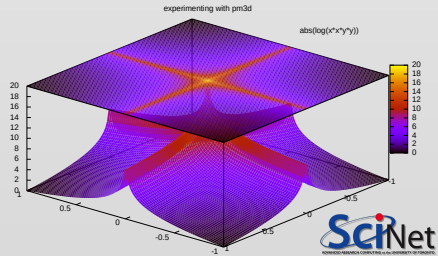
1D plotting

plotting functions of one variable, 1D tabulated data (eg. `gnuplot`, `xmgr`, or Python's `matplotlib` library)

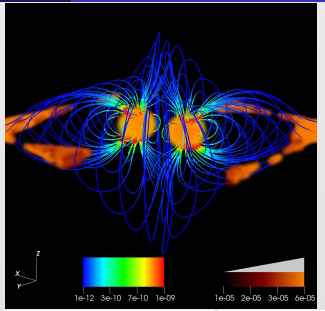


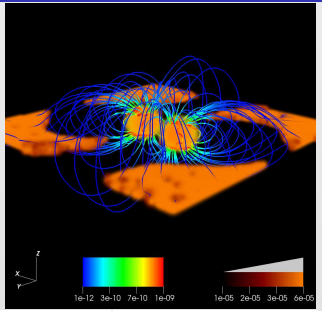
2D/3D visualization

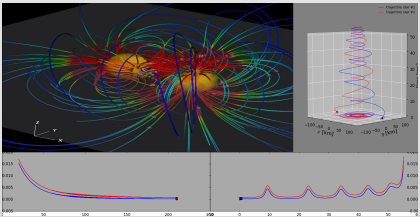
displaying multi-dimensional datasets, typically data on 2D/3D structured grids or on unstructured meshes (that have some topology in 2D/3D)

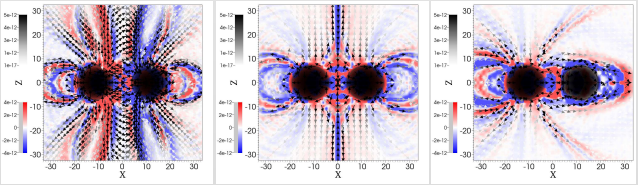


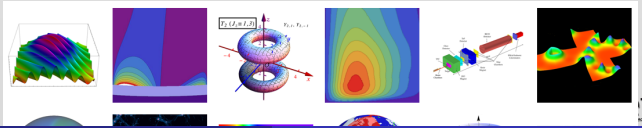
- represent data
- plotting
- visualization techniques
- analyze/explore
- communicate (publications/talks)
- make it look nice?











2D/3D visualization packages

- `gnuplot`: command-driven interactive 2d and 3d plotting program
- `GraphViz`: represent strucrural information as diagrams of abstract graphs and networks
- `HDFview`: visual tool for browsing and editing HDF4 and HDF5 files
- `ImageMagick`: manipulation of image
- `Molden`: pre/post-processing for molecular and electronic structures
- `OpenCV`: library for real time computer vision
- `ParaView`: Parallel visualization application
- `SciLab`: open source platform for numerical computation
- `VisIt`: Visualization Tool for HPC
- `XCrysDen`: Crystaline and Molecular Structure Visualization
- `yt`: python-based package for visualization of AMR datasets
- `VMD`: Visualization for Molecular Dynamics
- `openDX`: very old, not mantained package, but really nice approach to visualization process (modules)
- `blender`: . . .
- `tecPlot`: . . .

2D/3D visualization packages

General Features

- visualize scalar and vector fields
- structured and unstructured meshes in 2D and 3D, particle data, polygonal data, irregular topologies
- ability to handle very large datasets (GBs to TBs)
- ability to scale to large ($10^3 - 10^5$ cores) computing facilities
- iteractive manipulation
- support for scripting, common data formats, parallel I/O
- open-source, multi-platform, and general-purpose

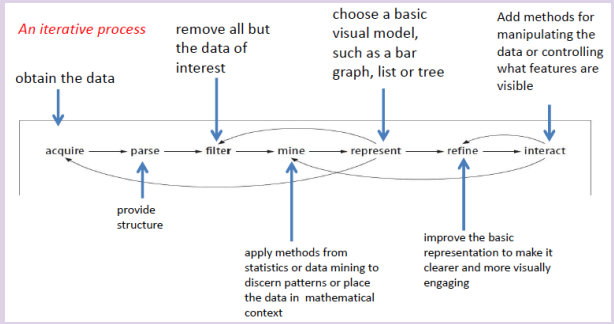
ParaView

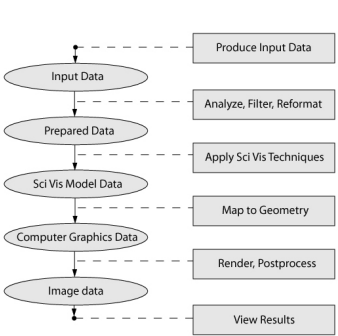
SciNet

Outline

1. Scientific Visualization
   - 2D/3D Visualization Generalities
   - Visualization Pipeline
2. VisIt
   - Generalities
   - Viz. Pipelines Basics
   - Movies Generation
   - Remote Visualization
   - Scripting
3. Summary
4. Further Resources
   - References

Data Visualization Process





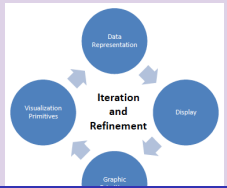
Scientific Visualization Techniques

- contours/isosurfaces, clips, thresholds, glyphs, streamlines, pseducolors, ...

Map to Geometry

- scalars, vectors, tensors
- 1D, 2D, 3D
- mesh/grid

Render/PostProcess



Visualization Toolkit (VTK)

- Open source, multiplatform
- Supports distributed computation models
- Extensible modular architecture
- Available for 3D computer graphics, image processing and visualization
- Collection of C++ libraries

- Leveraged by many applications
- Divided into logical areas
  - Filtering
  - Information Visualization
  - Volume Rendering
- Cross platform, using OpenGL
- Wrapped in Python, Tool Command Language (Tcl) and Java

ParaView and **VisIt** are end-user applications with support:

- *parallel* Data Archiving/Reading/Processing/Rendering
- single node, *Client-Server*, *MPI Cluster* Rendering

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

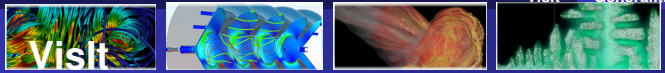
Outline

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

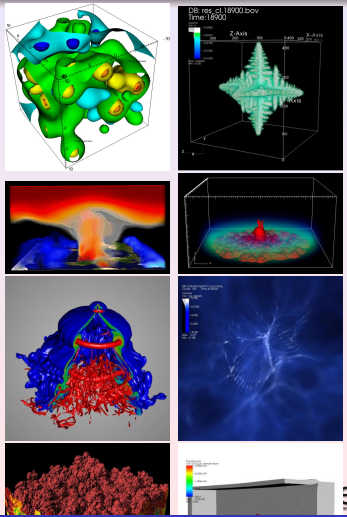
3 Summary

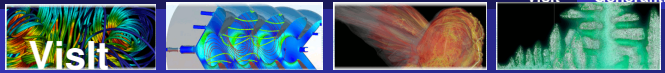
4 Further Resources
- References



https://wci.llnl.gov/simulation/computer-codes/visit

- Developed by the *DOE Advanced Simulation and Computing Initiative*, to visualize results of terascale simulations, first release fall of 2002 – mantained by *LLNL*
- For this class, **we will use v2.12.x** available as source and binary for Linux/Mac/Windows
- Latest version available is 3.0
- Over 80 visualization features (contour, mesh, slice, volume, molecule, ...)
- Reads over 110 different file formats
- Interfaces with C++, Python, and Java
- Uses MPI for distributed-memory parallelism on HPC clusters



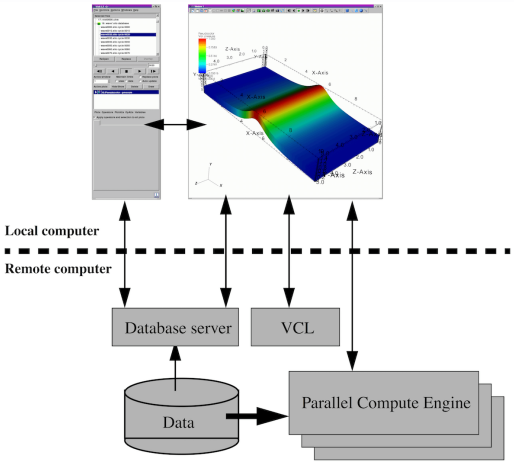


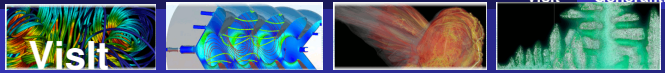
- can run: locally, remotelly, client/server mode
- interface pretty much looks the same on each platform
- can read over a 100 different data formats
- new database plugin readers can be developed

Supported Mesh Types

- 1D Curves
- 2D/3D meshes: Rectilinear, Curvilinear, Unstructured, Points, AMR, Molecular, CSG

VisIt Architecture



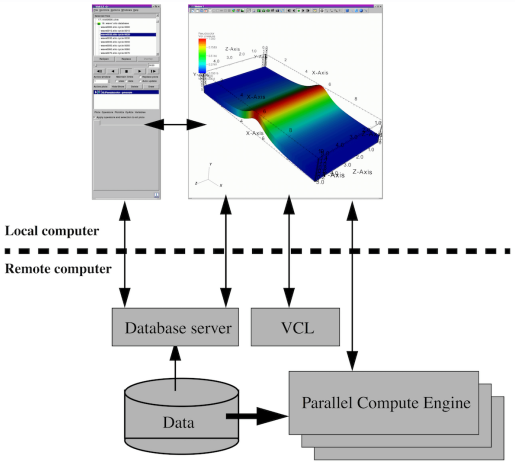


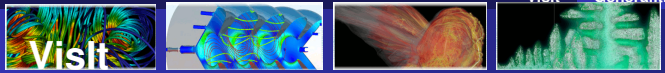
- can run: locally, remotelly, client/server mode
- interface pretty much looks the same on each platform
- can read over a 100 different data formats
- new database plugin readers can be developed

➡ Supported Mesh Types

- 1D Curves
- 2D/3D meshes: Rectilinear, Curvilinear, Unstructured, Points, AMR, Molecular, CSG

➡ VisIt Architecture



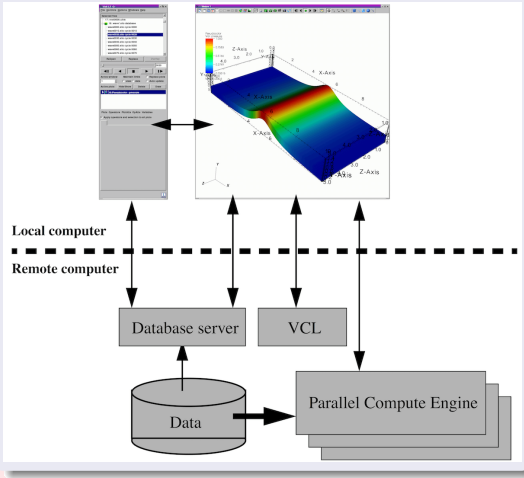


- can run: locally, remotelly, client/server mode
- interface pretty much looks the same on each platform
- can read over a 100 different data formats
- new database plugin readers can be developed

➡ Supported Mesh Types

- 1D Curves
- 2D/3D meshes: Rectilinear, Curvilinear, Unstructured, Points, AMR, Molecular, CSG

➡ VisIt Architecture



VisIt: Multiple Interfaces

- GUI (graphical user interface)
- Python programming interface
- Java programming interface
- C++ programming interface

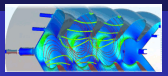
- Use multiple interfaces simultaneously
- Use VisIt as an application or a library
- C++, Python, Java interfaces allow other applications to control VisIt

VisIt: Multiple Interfaces

- GUI (graphical user interface)
- Python programming interface
- Java programming interface
- C++ programming interface

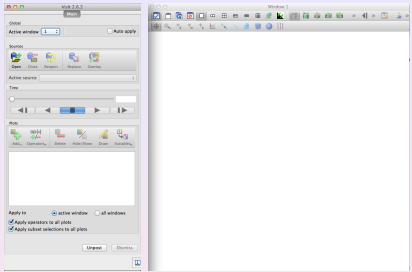
◗ Use multiple interfaces simultaneously

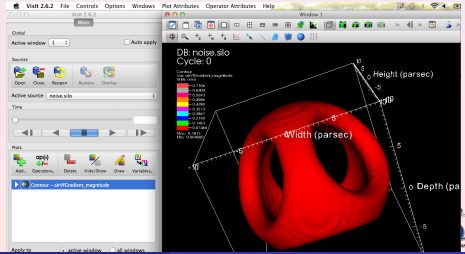
➨ Use VisIt as an application or a library
➨ C++, Python, Java interfaces allow other applications to control VisIt

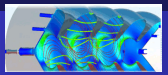


VisIt: GUI

- GUI
  - Select files to visualize
  - Create and manage plots
  - Set plot attributes
  - Add operators
  - Set look and props. for visualization
- Viewer
  - display all of the data being visualized
  - Mouse navigation
  - up to 16 vis windows
  - Popup menu

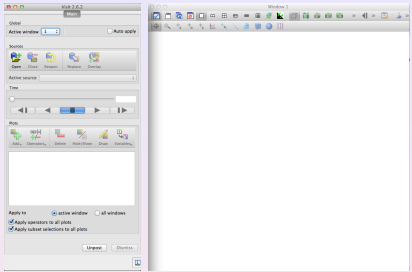


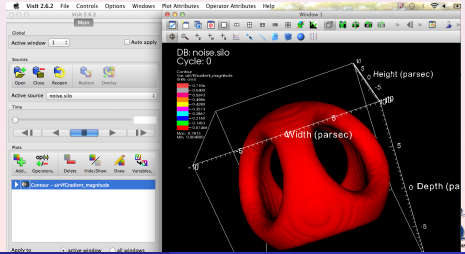


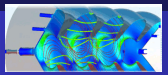


VisIt: GUI

- GUI
  - Select files to visualize
  - Create and manage plots
  - Set plot attributes
  - Add operators
  - Set look and props. for visualization
- Viewer
  - display all of the data being visualized
  - Mouse navigation
  - up to 16 vis windows
  - Popup menu

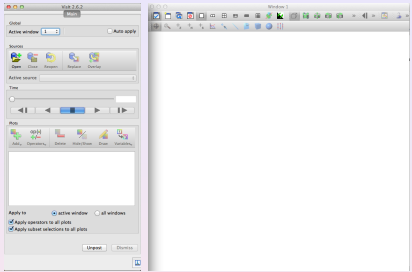


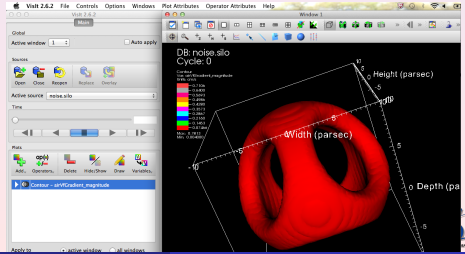




VisIt: GUI

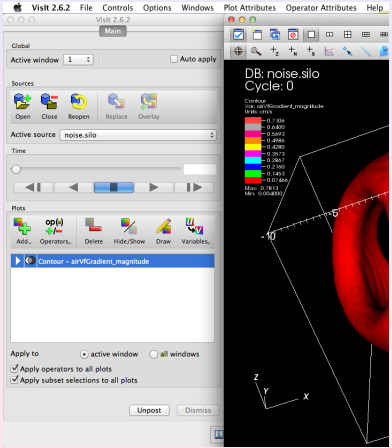
- GUI
  - Select files to visualize
  - Create and manage plots
  - Set plot attributes
  - Add operators
  - Set look and props. for visualization
- Viewer
  - display all of the data being visualized
  - Mouse navigation
  - up to 16 vis windows
  - Popup menu





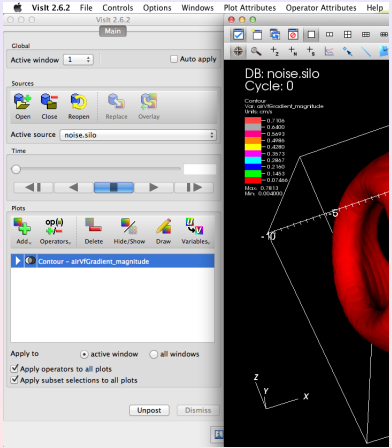
VisIt: GUIs

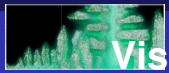
- Main window in GUI
  - Access other important windows
  - Open files
  - Set animation time state
  - Set active window
  - Create and manage filters (pipeline)
  - Displays progress from compute engine
- Really useful ones
  - Appy to... check-boxes



VisIt: GUIs

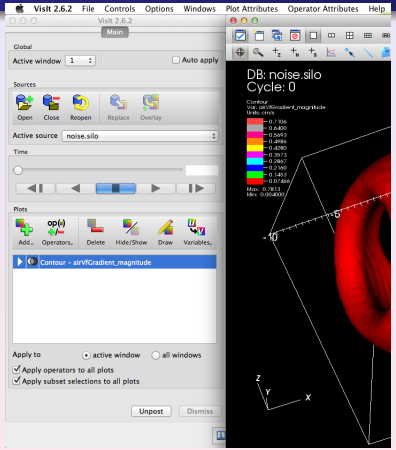
- Main window in GUI
  - Access other important windows
  - Open files
  - Set animation time state
  - Set active window
  - Create and manage filters (pipeline)
  - Displays progress from compute engine
- Really useful ones
  - Appy to... check-boxes





VisIt: GUIs

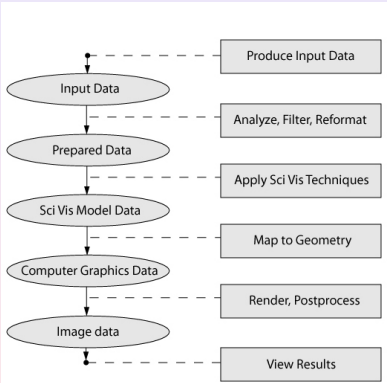
- Main Menu
  - File
  - Controls
  - Options
  - Windows
  - Plot Attributes
  - Operator Attributes
  - Help



- Visualization Toolbar

VisIt: Visualization Pipeline

1. Open database (or file)
2. Create a plot
3. Set plot attributes
4. Apply operators to plot to modify data
5. Set operator attributes
6. Compute engine generates plot
7. Plot displayed in vis window
8. iterate//repeat...



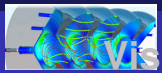
Outline

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

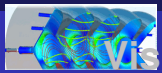


VisIt: Import Data

- File → **Open file...**
  ↦ choose data file (“noise.silo”)

- Become available
  - Active source: “noise.silo”
  - Add button

- File → File information...

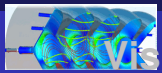


VisIt: Import Data

- File → **Open file...**
  ↦ choose data file (“noise.silo”)

- Become available
  - Active source: “noise.silo”
  - Add button

- File → File information...



VisIt: Import Data

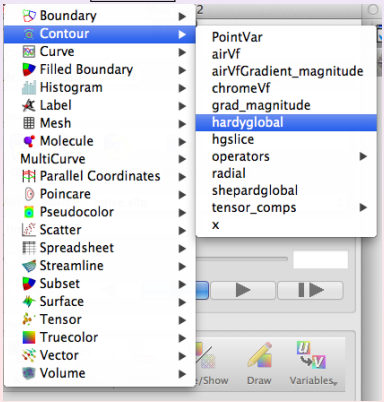
- File → **Open file...**
  ↦ choose data file (“noise.silo”)

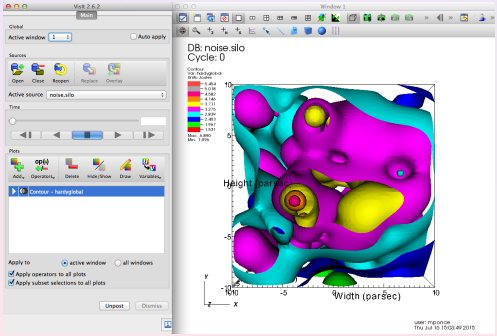
- Become available
  - Active source: “noise.silo”
  - Add button

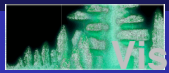
- File → **File information...**

VisIt: Contours

- Add → Contours
  - ⇝ *hardyglobal*
  - → Draw

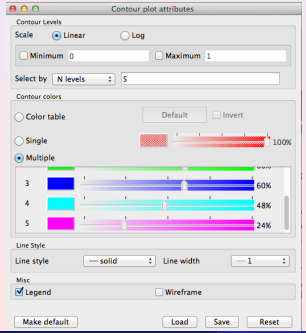


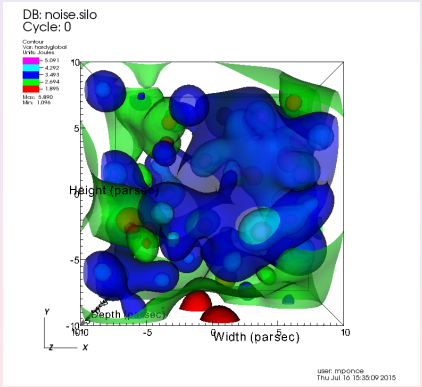


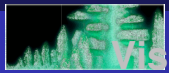


VisIt: Contours

- double-click on Contour-hardyglobal
- under Select by, choose N levels = 5 [Enter]
- Change opacity levels, eg: 100%, 60%, 60%, 48%, 24%
- Apply & Dismiss
- Hide/Show or Delete

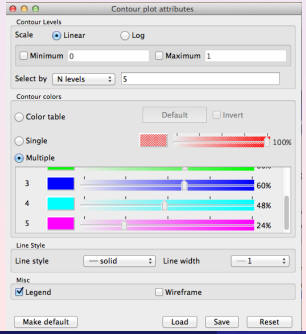


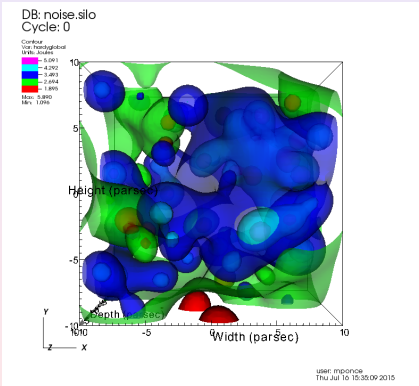


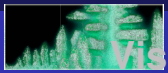


VisIt: Contours

- double-click on Contour-hardyglobal
- under Select by, choose N levels = 5 Enter
- Change opacity levels, eg: 100%, 60%, 60%, 48%, 24%
- Apply & Dismiss
- Hide/Show or Delete

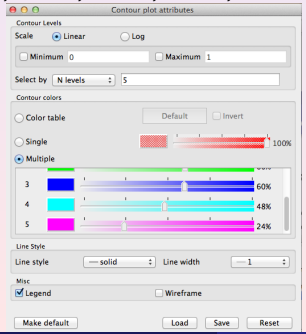




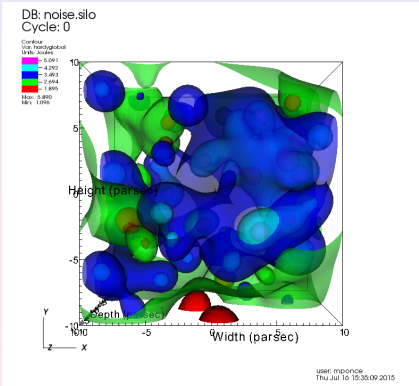


VisIt: Contours

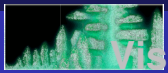
- double-click on Contour-hardyglobal
- under Select by, choose N levels = 5 Enter
- Change opacity levels, eg: 100%, 60%, 60%, 48%, 24%



- Apply & Dismiss

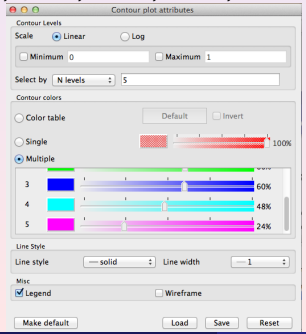


- Hide/Show or Delete

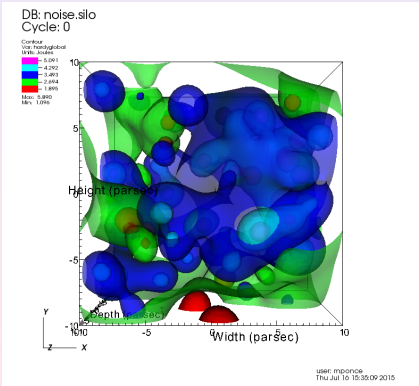


VisIt: Contours

- double-click on Contour-hardyglobal
- under Select by, choose N levels = 5 Enter
- Change opacity levels, eg: 100%, 60%, 60%, 48%, 24%



- Apply & Dismiss



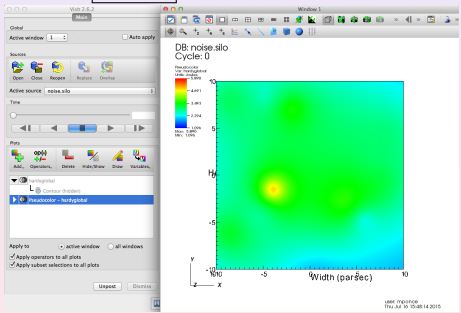
- Hide/Show or Delete

VisIt: PseudoColor & IsoSurfaces

➠ Add ▾ → **Pseudocolor**

⇝ *hardyglobal*

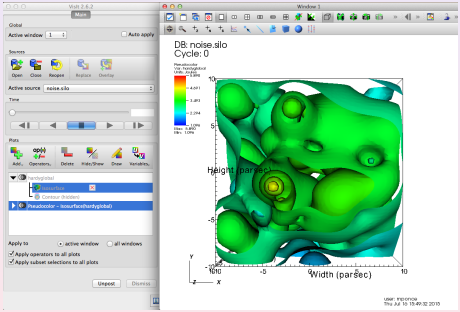
→ Draw



➠ Operators ▾ → Slicing

→ Isosurface

→ Draw

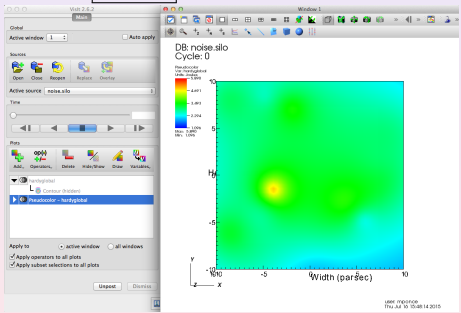


➠ click ▸ to expand, double click on [Isosurface]

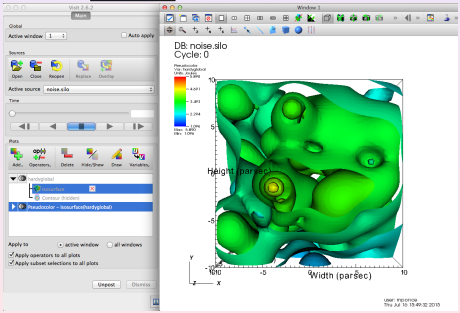
➠ under Select by, choose Percent (s) = 50 Enter & Dismiss

VisIt: PseudoColor & IsoSurfaces

- Add → **Pseudocolor**
  - ⇝ *hardyglobal*
  - → Draw



- Operators → **Slicing**
  - → **Isosurface**
  - → Draw

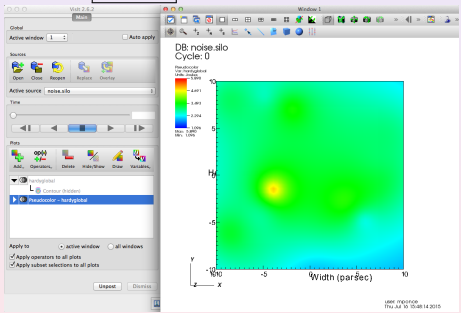


➟ click ▸ to expand, double click on [Isosurface]

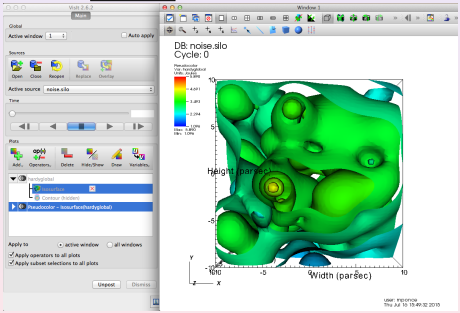
➟ under Select by, choose Percent (s) = 50 Enter & Dismiss

VisIt: PseudoColor & IsoSurfaces

- Add → **Pseudocolor**
  - ⇝ *hardyglobal*
  - → Draw



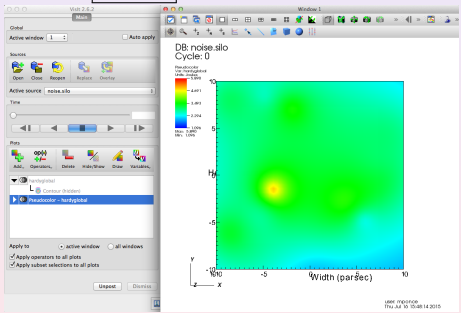
- Operators → **Slicing**
  - → **Isosurface**
  - → Draw



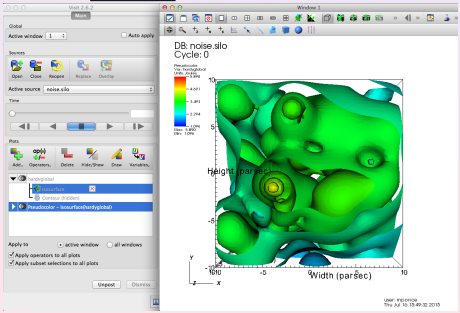
- click ▸ to expand, double click on [Isosurface]
  - under Select by, choose Percent (s) = 50 Enter & Dismiss

VisIt: PseudoColor & IsoSurfaces

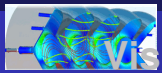
- Add → **Pseudocolor**
  - ⇝ *hardyglobal*
  - → Draw



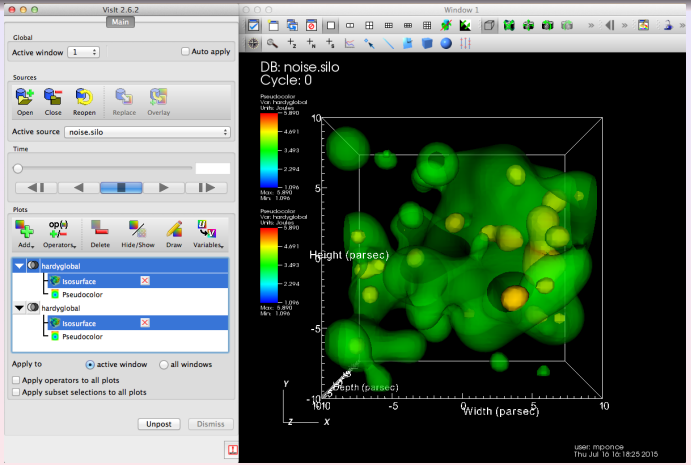
- Operators → **Slicing**
  - → **Isosurface**
  - → Draw

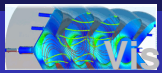


- click ▸ to expand, double click on [Isosurface]
  - under Select by, choose Percent (s) = 50 Enter & Dismiss

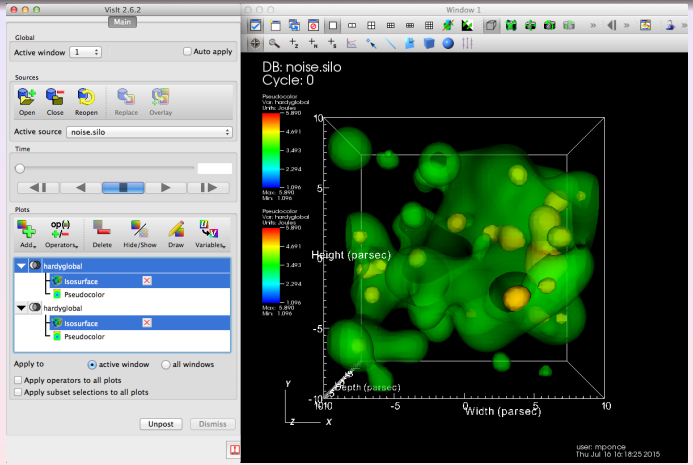


VisIt: PseudoColor & IsoSurfaces



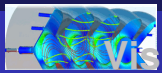


VisIt: PseudoColor & IsoSurfaces

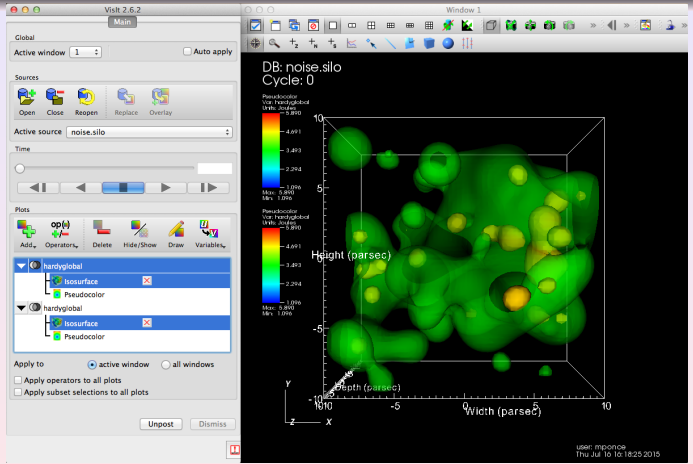


- unselect the [Apply ...] check-boxes
- add 1 more Pseudocolor +**Isosurface**, w/Percent(s)=80 & adjust its opacity

- clipping → select/check the [Apply...] boxes
- Operators → **Selection** → Clip



VisIt: PseudoColor & IsoSurfaces



- unselect the [Apply ...] check-boxes
- add 1 more Pseudocolor +**Isosurface**, w/Percent(s)=80 & adjust its opacity

- clipping → select/check the [Apply...] boxes
- Operators → **Selection** → **Clip**

VisIt: General Remarks

- operators/plots can be removed **Delete** –or– hidden **Hide/Show**
- save your work **frequently**: File → **Save session...**

* Restoring a previous session

- File → **Restore session...**

loads the previous state of the given session (**that needs to be specifically saved**)

- File → **Restore session w/sources...**

extremely useful for re-identifiying datasets that could have been moved or renamed

Be aware, that VisIt, by default won't save your work (session) nor ask you when you try to exit the program!

VisIt: General Remarks

- operators/plots can be removed **Delete** –or– hidden **Hide/Show**
- save your work **frequently**: File → **Save session...**

* Restoring a previous session

- File → **Restore session...**

loads the previous state of the given session (**that needs to be specifically saved**)

- File → **Restore session w/sources...**

extremely useful for re-identifiying datasets that could have been moved or renamed

Be aware, that VisIt, by default won't save your work (session) nor ask you when you try to exit the program!

VisIt: General Remarks

- operators/plots can be removed **Delete** –or– hidden **Hide/Show**
- save your work **frequently**: File → **Save session...**

* Restoring a previous session

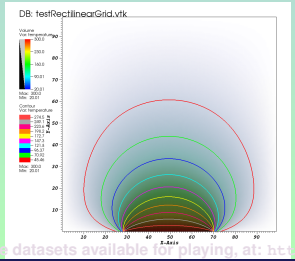
- File → **Restore session...**
loads the previous state of the given session (**that needs to be specifically saved**)
- File → **Restore session w/sources...**
extremely useful for re-identifiying datasets that could have been moved or renamed

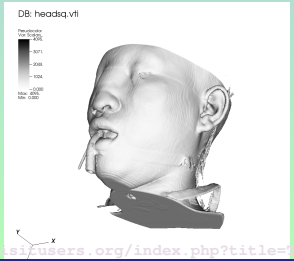
Be aware, that VisIt, by default won't save your work (session) nor ask you when you try to exit the program!

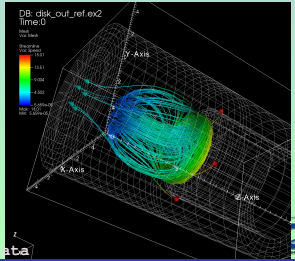
VisIt: Intermesso i

Hands-on...

- Load some of the other datasets: [eg. "testRectilinearGrid.vtk", "headsq.vtk", "disk_out_ref.ex2", ...] –or– *your own data*!!!
- Try to explore the data and visualize it, using some of the tools we have discussed
- If you have used other visualization packages, compare how easy and whether it is possible or not, to obtain similar results, let's say, with ParaView or VAPOR for instance...
- also which one, results more intuitive, elegant, useful for you and your research





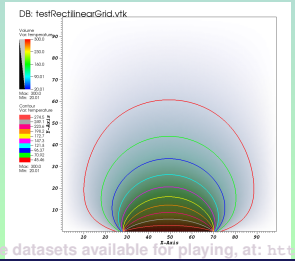


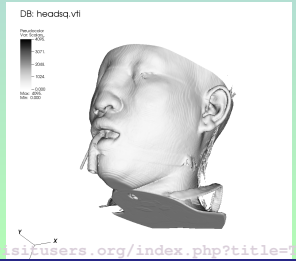
More datasets available for playing, at: https://www.visitusers.org/index.php?title=Tutorial_Data

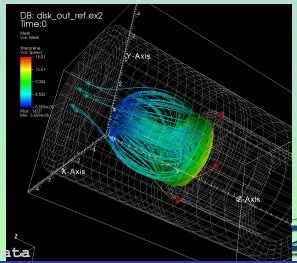
VisIt: Intermesso i

Hands-on...

- Load some of the other datasets: [eg. "testRectilinearGrid.vtk", "headsq.vtk", "disk_out_ref.ex2", ...] –or– *your own data*!!!
- Try to explore the data and visualize it, using some of the tools we have discussed
- If you have used other visualization packages, compare how easy and whether it is possible or not, to obtain similar results, let's say, with ParaView or VAPOR for instance...
- also which one, results more intuitive, elegant, useful for you and your research





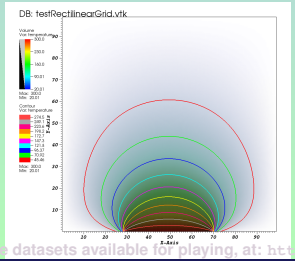


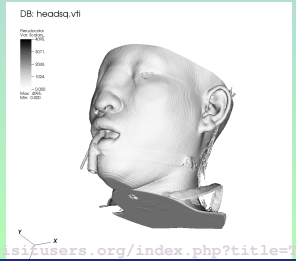
More datasets available for playing, at: http://www.visitusers.org/index.php?title=Tutorial_Data

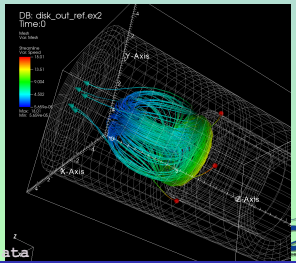
VisIt: Intermesso i

Hands-on...

- Load some of the other datasets: [eg. "testRectilinearGrid.vtk", "headsq.vtk", "disk_out_ref.ex2", ...] –or– *your own data*!!!
- Try to explore the data and visualize it, using some of the tools we have discussed
- If you have used other visualization packages, compare how easy and whether it is possible or not, to obtain similar results, let's say, with ParaView or VAPOR for instance...
- also which one, results more intuitive, elegant, useful for you and your research





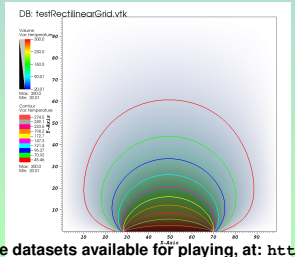


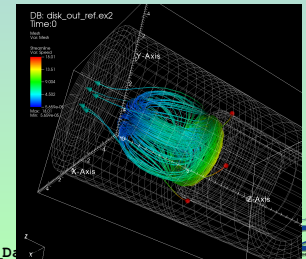
More datasets available for playing, at: http://www.visitusers.org/index.php?title=Tutorial_Data

VisIt: Intermesso i

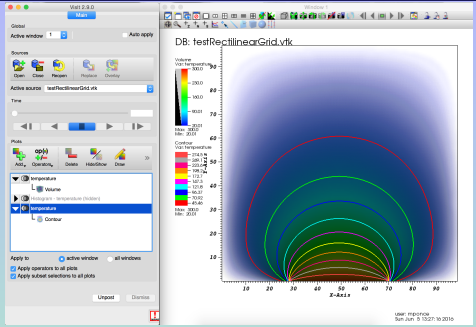
Hands-on...

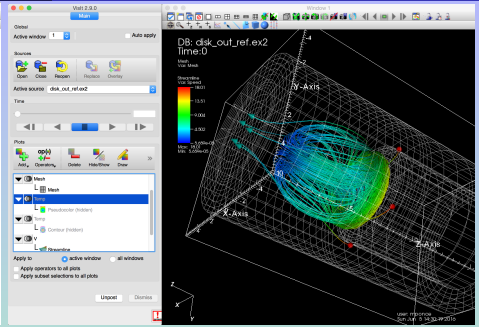
- Load some of the other datasets: [eg. "testRectilinearGrid.vtk", "headsq.vtk", "disk_out_ref.ex2", ...] –or– *your own data*!!!
- Try to explore the data and visualize it, using some of the tools we have discussed
- If you have used other visualization packages, compare how easy and whether it is possible or not, to obtain similar results, let's say, with ParaView or VAPOR for instance...
- also which one, results more intuitive, elegant, useful for you and your research

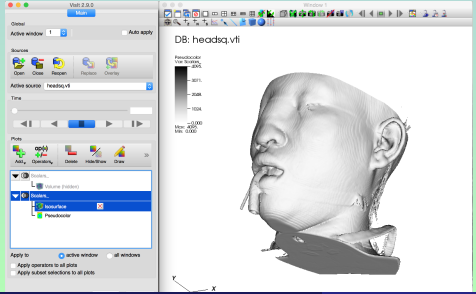


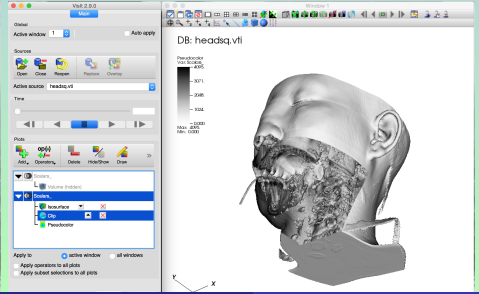


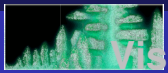
More datasets available for playing, at: `http://www.visitusers.org/index.php?title=Tutorial_Da`


DB: testRectilinearGrid.vtk
X-Axis


DB: disk_out_ref.ex2
Time:0
Y-Axis
X-Axis
Z-Axis


DB: headsq.vti

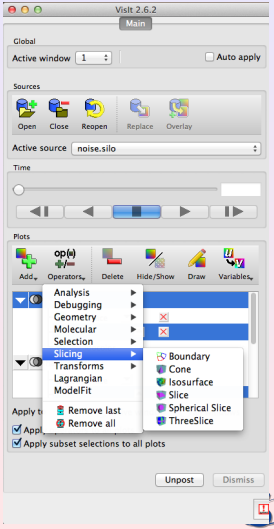

DB: headsq.vti



VisIt: Slices

Slicing Isosurfaces

- Operators → **Slicing** → **Slice**
- double-click on [Slice]
  - try choosing different axes & Project to 2D
  - try also the other types of slices



VisIt: Vector Field representations – Glyphs

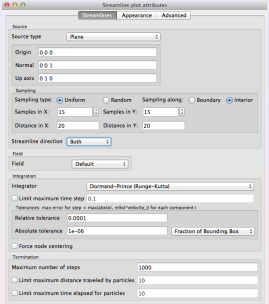
- Glyphs
  - Add$_{\blacktriangledown}$ → **Vector** ⇝ *airVfGradient*
    - Apply & Dismiss
  - double-click on [Vector]
    - set Vector amount ⇝ 1000
    - more *properties* in [Data] & [Glyph]

VisIt: Vector Field representations – Streamlines

- Streamlines

Add → Pseudocolor → operators → **IntegralCurve** ⇝ grad

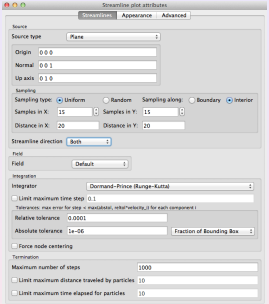
- double-click on [IntegralCurve]
  - set Source type ⇝ plane
  - increase samples/distance along axes ⇝ 15/20
  - Integration direction ⇝ both



- double-click on [Pseudocolor]
  - in [Geommetry], Line type ⇝ Tubes, w/Radius ⇝ 0.005
  - [Data] ⇝ bluehot → Apply

VisIt: Vector Field representations – Streamlines

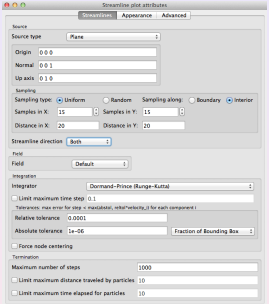
- Streamlines
  - Add → Pseudocolor → operators → **IntegralCurve** ⇝ grad
  - double-click on [IntegralCurve]
    - set Source type ⇝ plane
    - increase samples/distance along axes ⇝ 15/20
    - Integration direction ⇝ both



- double-click on [Pseudocolor]
  - in [Geommetry], Line type ⇝ Tubes, w/Radius ⇝ 0.005
  - [Data] ⇝ bluehot → Apply

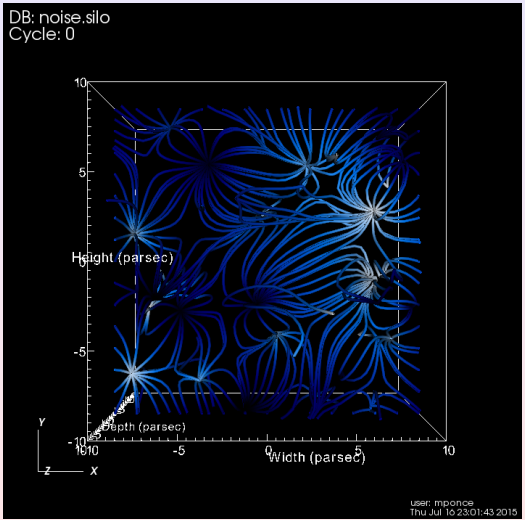
VisIt: Vector Field representations – Streamlines

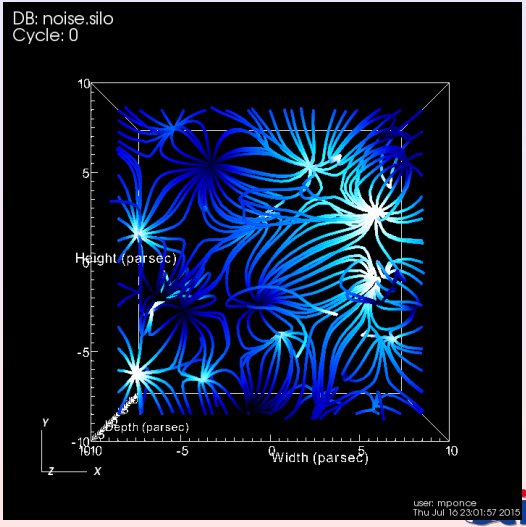
- Streamlines
  - Add → Pseudocolor → operators → **IntegralCurve** ⇝ grad
  - double-click on [IntegralCurve]
    - set Source type ⇝ plane
    - increase samples/distance along axes ⇝ 15/20
    - Integration direction ⇝ both



- double-click on [Pseudocolor]
  - in [Geommetry], Line type ⇝ Tubes, w/Radius ⇝ 0.005
  - [Data] ⇝ bluehot → Apply

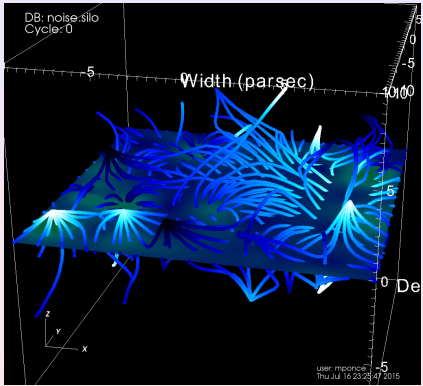
VisIt: Vector Field representations – Streamlines





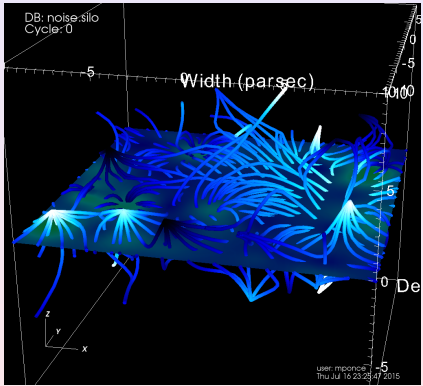
VisIt: Slices

- Add → **Pseudocolor**
  - ⇝ grad_magnitude
- Operator → **Slicing** ⇝ **Slice**
- double-click on [Slice]
  - select ⇝ Z-axis
  - unselect Project to 2D
  - Apply
  - Draw
  - Hide/Show



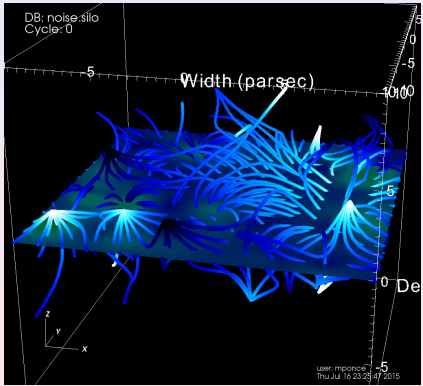
VisIt: Slices

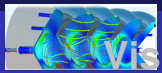
- Add ▾ → **Pseudocolor**
  - ⇝ grad_magnitude
- Operator ▾ → **Slicing** ⇝ **Slice**
- double-click on [Slice]
  - select ⇝ Z-axis
  - unselect Project to 2D
  - Apply
  - Draw
  - Hide/Show



VisIt: Slices

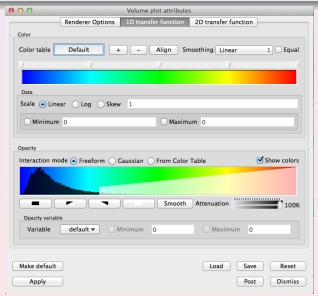
- Add ▾ → **Pseudocolor**
  - ⇝ grad_magnitude
- Operator ▾ → **Slicing** ⇝ **Slice**
- double-click on [Slice]
  - select ⇝ Z-axis
  - unselect Project to 2D
  - Apply
  - Draw
  - Hide/Show



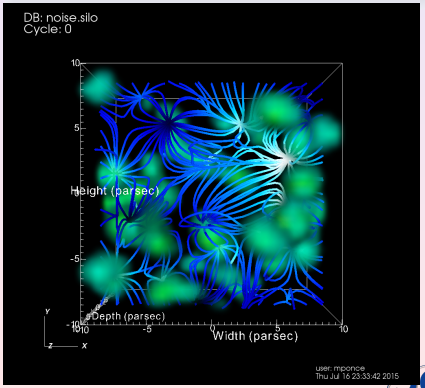


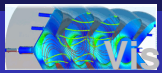
VisIt: Volume Rendering

- Add → **Volume**
  - ⇝ grad_magnitude
- double-click on [Volume]
  - click on 1D Transfer Function
  - change Transfer Fn/**Opacity**



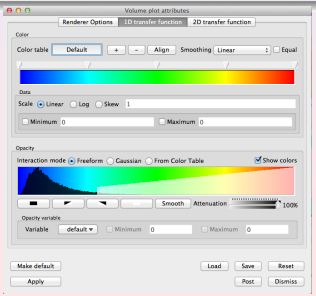
- Apply
- Draw



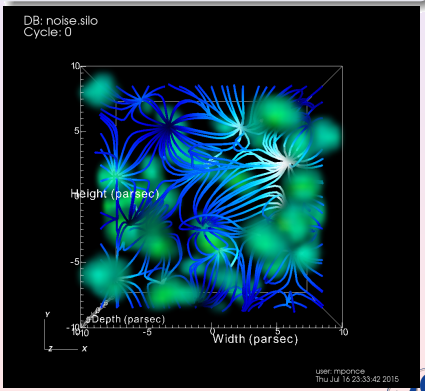


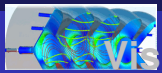
VisIt: Volume Rendering

- Add → **Volume**
  - ⇝ grad_magnitude
- double-click on [Volume]
  - click on 1D Transfer Function
  - change Transfer Fn/**Opacity**



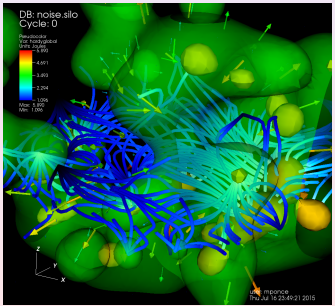
- Apply
- Draw

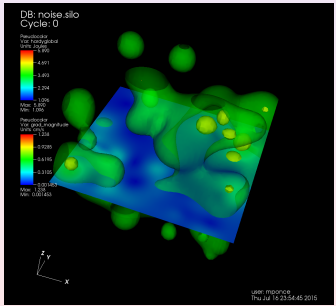


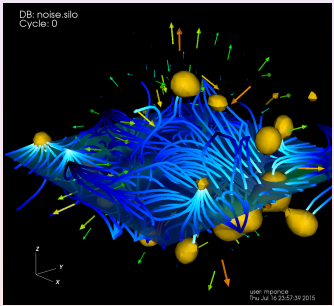


VisIt: *Aesthetics* & Final Products

- Legends, axes, ...
  - Controls → **Annotation...**
- "hard-copies": images, ...
  - File → **Save Window**







Outline

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

Movie Generation

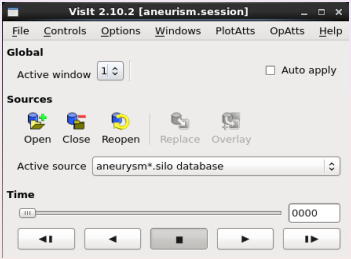
Movies & Animations

- sequence in time (*evolution*)
- motion through space (usually done through *scripting*)

Basic *timestep* Animation

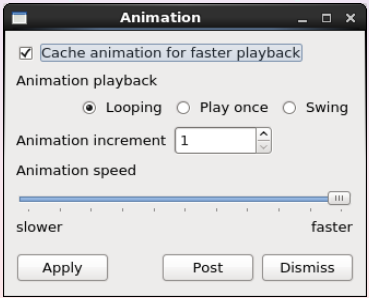
- Simplest case: “static animation”, in which only the *database timestep* changes
- Allows database behaviour over time to be quickly inspected (without the complexity of scripting)
- Controled through [VCR]-typeof buttons on the *time-framming* panel of the main windows

[Time Slider]



[Main Window]

[Animation Window]

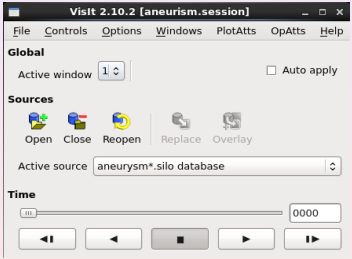


Controls → Animation...

Basic *timestep* Animation

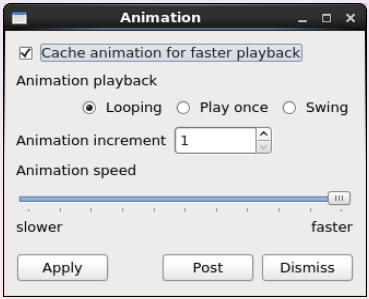
- Simplest case: “static animation”, in which only the *database timestep* changes
- Allows database behaviour over time to be quickly inspected (without the complexity of scripting)
- Controled through [VCR]-typeof buttons on the *time-framming* panel of the main windows

[Time Slider]



[Main Window]

[Animation Window]

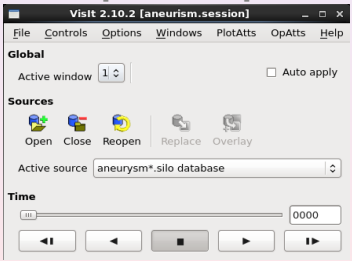


Controls → Animation...

Basic *timestep* Animation

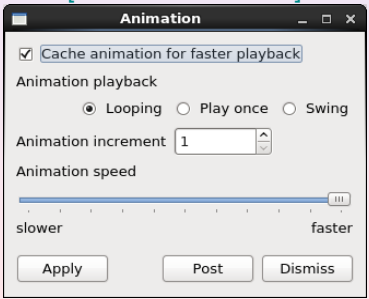
- Simplest case: “static animation”, in which only the *database timestep* changes
- Allows database behaviour over time to be quickly inspected (without the complexity of scripting)
- Controled through [VCR]-typeof buttons on the *time-framming* panel of the main windows

[Time Slider]



[Main Window]

[Animation Window]



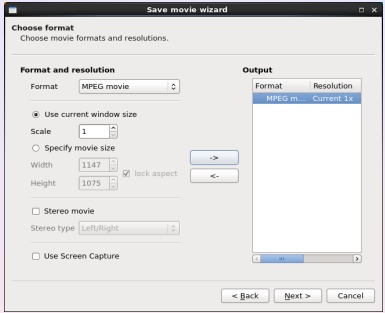
Controls → **Animation...**

Movie Wizard

File → **Save Movie...**

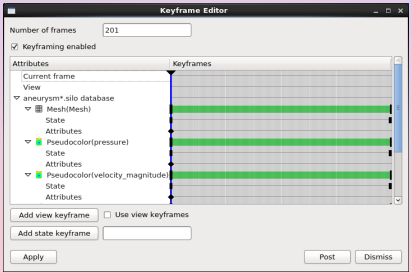
Guided Movie Generation

- Produce several formats and resolutions, at the same time
- Stereo movies
- Can handle currently allocated processors or spawn another VisIt session for movie-generation
- Can use movie templates to assemble complex sequence of frames



Keyframing

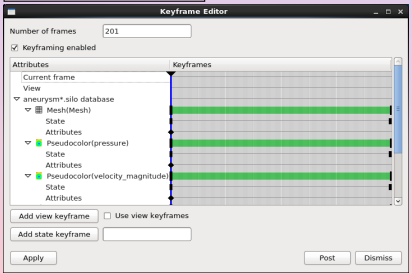
- Advanced form of animation, that allows attributes to change as the animation progresses
- Attributes that can be keyframed: Plots attributes, Database states, View
- Eg. make a plot fade out as the animation progresses, make view slowly change, ...



Keyframing

- Advanced form of animation, that allows attributes to change as the animation progresses
- Attributes that can be keyframed: Plots attributes, Database states, View
- Eg. make a plot fade out as the animation progresses, make view slowly change, ...

➠ Controls → **Keyframing...**



➠ [time slider]: "Keyframing Animation"

➠ Choose the desired **active** time slider from the pull-down menu

➠ Enable keyframing mode

➠ Adjust number of frames

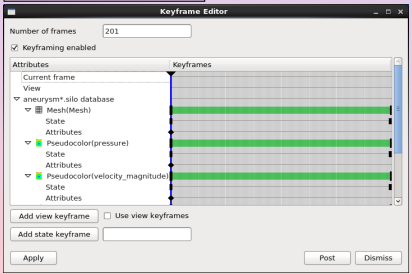
- a keyframe is created for each plot attrib.
- Plot attribs are calculated for each frame

➠ Using the keyframe indicators ◆, set the time range for each attribute

Keyframing

- Advanced form of animation, that allows attributes to change as the animation progresses
- Attributes that can be keyframed: Plots attributes, Database states, View
- Eg. make a plot fade out as the animation progresses, make view slowly change, ...

Controls → **Keyframing...**

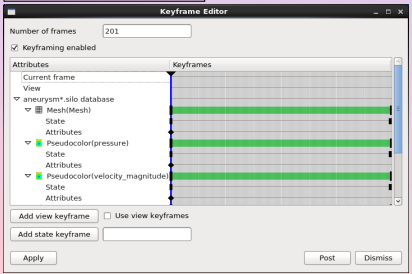


- Enable "keyframing mode"
- Adjust number of frames
  - a keyframe is created for each plot attrib.
- Plot attribs are calculated for each frame
- Using the keyframe indicators ◆, set the time range for each attribute

- [time slider]: "Keyframing Animation"
- Choose the desired active time slider from the pull-down menu

Keyframing

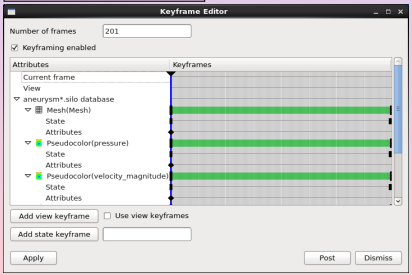
- Advanced form of animation, that allows attributes to change as the animation progresses
- Attributes that can be keyframed: Plots attributes, Database states, View
- Eg. make a plot fade out as the animation progresses, make view slowly change, ...
  - Controls → **Keyframing...**



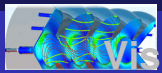
  - Enable "keyframing mode"
  - Adjust number of frames
    - a keyframe is created for each plot attrib.
    - Plot attribs. are calculated for each frame
  - Using the keyframe indicators ♦, set the time range for each attribute
  - [time slider]: "Keyframing Animation"
  - Choose the desired active time slider from the pull-down menu

Keyframing

- Advanced form of animation, that allows attributes to change as the animation progresses
- Attributes that can be keyframed: Plots attributes, Database states, View
- Eg. make a plot fade out as the animation progresses, make view slowly change, ...
  - Controls → **Keyframing...**



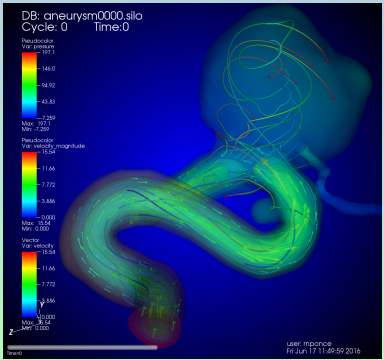
  - Enable "keyframing mode"
  - Adjust number of frames
    - a keyframe is created for each plot attrib.
    - Plot attribs. are calculated for each frame
  - Using the keyframe indicators ♦, set the time range for each attribute
- [time slider]: "Keyframing Animation"
- Choose the desired **active** time slider from the pull-down menu

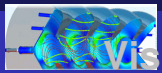


VisIt: Intermesso ii

Hands-on...

- Using the "`aneurysm`" dataset –or– *your own data*, generate a time sequence movie
- Experiment with keyframming, lighting, ... or any of the other techniques we have been discussing

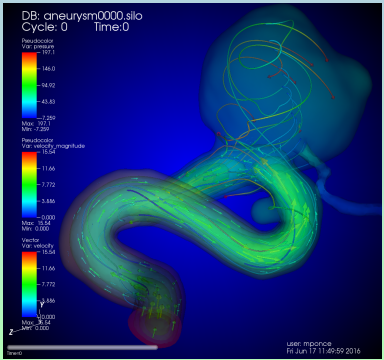


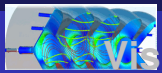


VisIt: Intermesso ii

Hands-on...

- Using the "`aneurysm`" dataset –or– *your own data*, generate a time sequence movie
- Experiment with keyframming, lighting, ... or any of the other techniques we have been discussing

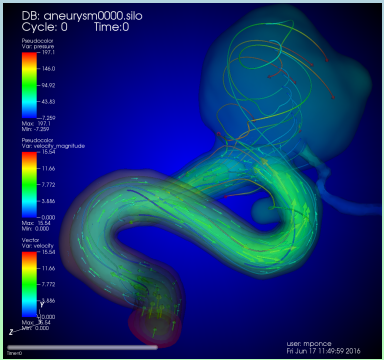


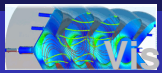


VisIt: Intermesso ii

Hands-on...

- Using the "`aneurysm`" dataset –or– *your own data*, generate a time sequence movie
- Experiment with keyframming, lighting, ... or any of the other techniques we have been discussing

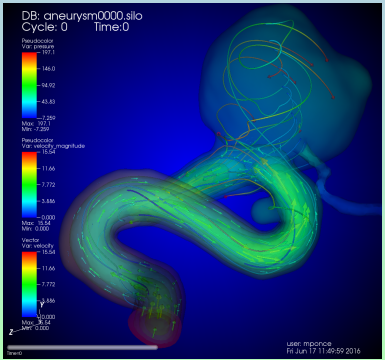




VisIt: Intermesso ii

Hands-on...

- Using the "`aneurysm`" dataset –or– *your own data*, generate a time sequence movie
- Experiment with keyframming, lighting, ... or any of the other techniques we have been discussing



Outline

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

Remote Visualization

Thre may be different reasons why one would want to visualize data remotelly:

- data locality: data is located in a remote site, eg. cluster
- the remote server is more powerful and can generate more demanding visualizations
- ...

Ways to Visualize Remotely

- X-forwarding
- VNC
- Server-Client protocol, natively provided by the Viz.Suite

X-forwarding

- the viz. software will run on the remote host
- all the graphics are forwarded to the local computer
- it is usually slow! Alternatively one could use VNC instead...

```
ssh -X USERNAME@bridges.psc.edu
```

```
module load visit
visit
# for executing in parallel with 28
processors
module load intel
visit -np 28
```

Try running,
`visit -help`
`visit -fullhelp`
for more command line arguments...

However, `bridges` does "not" provide interactive VisIt via x-forwarding, hence this method won't work on bridges!

X-forwarding

- the viz. software will run on the remote host
- all the graphics are forwarded to the local computer
- it is usually slow! Alternatively one could use VNC instead...

```
ssh -X USERNAME@bridges.psc.edu
```

```
module load visit
visit
# for executing in parallel with 28
processors
module load intel
visit -np 28
```

Try running,
`visit -help`
`visit -fullhelp`
for more command line arguments...

However, bridges does "not" provide interactive VisIt via x-forwarding, hence this method won't work on bridges!

X-forwarding

- the viz. software will run on the remote host
- all the graphics are forwarded to the local computer
- it is usually slow! Alternatively one could use VNC instead...

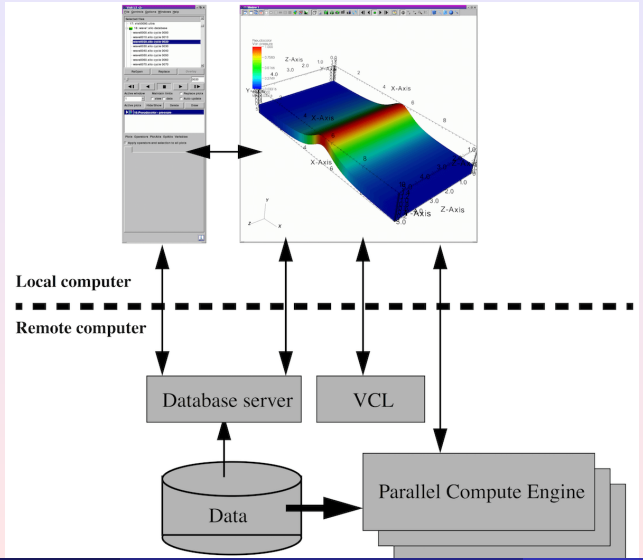
```
ssh -X USERNAME@bridges.psc.edu
```

```
module load visit
visit
# for executing in parallel with 28
processors
module load intel
visit -np 28
```

Try running,
`visit -help`
`visit -fullhelp`
for more command line arguments...

However, `bridges` does *not* provide interactive VisIt via x-forwarding, hence this method won't work on bridges!

Client-Server Protocol



Remote Visualization – Client-Server Mode

VisIt supports “remote visualization” protocols, which includes:

- accessing data remotely, ie. stored on the cluster
- rendering visualizations using the compute nodes as rendering engines
- or both

For allowing VisIt connect to the Bridges cluster you need to set up a "Host Configuration"

1. use the Bridges Host configuration file:
   `https://support.scinet.utoronto.ca/~mponce/viz/host_bridges.xml`
2. configure the host manually

* OBS: VisIt requires to use the same version in both the local client and the remote host!

Remote Visualization – Client-Server Mode

VisIt supports “remote visualization” protocols, which includes:

- accessing data remotely, ie. stored on the cluster
- rendering visualizations using the compute nodes as rendering engines
- or both

For allowing VisIt connect to the Bridges cluster you need to set up a "Host Configuration"

1. use the Bridges Host configuration file:
   `https://support.scinet.utoronto.ca/~mponce/viz/host_bridges.xml`
2. configure the host manually

* OBS: VisIt requires to use the same version in both the local client and the remote host!

Using the Bridges Host configuration file

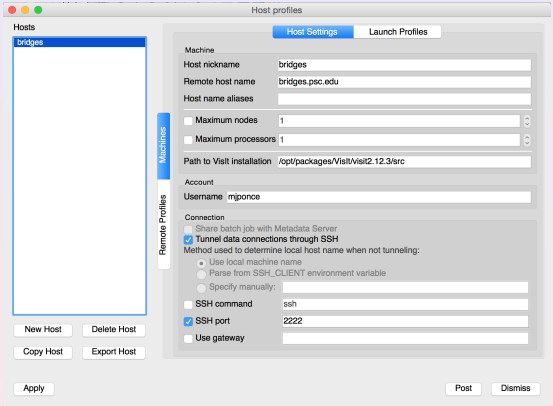
1. Download the Bridges Host configuration file, from the following link
   `https://support.scinet.utoronto.ca/~mponce/viz/host_bridges.xml`
2. Depending on the OS you are using on your local machine:
   - on a **Linux/Mac OS** place this file in `~/.visit/hosts/`
   - on a **Windows** machine, place the file in `My Documents\VisIt 2.12.x\hosts\`
3. Restart VisIt and check that the brigdes profile should be available in your hosts [Options] ➠ Host Profiles...

Manual Host Configuration

1. [Options] ➠ "Host profiles...", click on 'New Host' and select:

   ```
   Host nickname = bridges
   Remote host name = bridges.psc.edu
   Username = YOURusername
   Path to VisIt installation =
   /opt/packages/VisIt/visit2.12.3/src/
   ```

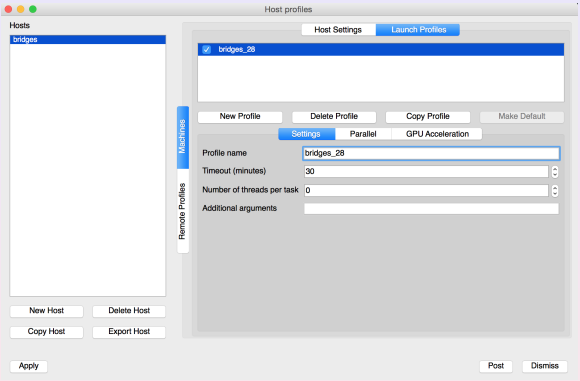
2. Click on the "Tunnel data connections through SSH", and then hit Apply!

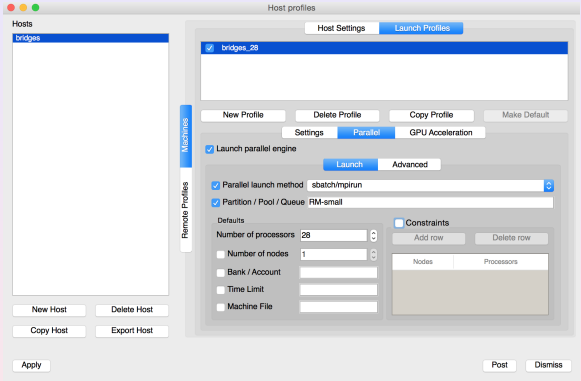


Manual Host Configuration (cont.)

3. On the top of the window click on 'Launch Profiles' tab.
   You will have to create a profile:
   - `Profile name:` `bridges_28`
4. Then click on the Parallel tab and set the "Launch parallel engine"
   - `Parallel launch method:` `sbatch/mpirun`
   - `Partition/Pool/Queue:` `RM-small`
   - `Number of processors:` `28`
5. Don't forget to select Options/Save Settings!

Manual Host Configuration (cont.)





6. Finally, go to the "Options" menu and select "Save settings", so that your changes are saved and available next time you relaunch VisIt.

Accessing Remote Hosts...

- [File] ➠ Open file... ⇝ click on "Host ▾"
- A list possible hosts should display, including "bridges" – select it!
- You will be asked to enter your password, check that the username is correct! Otherwise, click on [Change Username]!
- If the connection was sucessful, the window should be displaying the remote File System, eg. `Path: /home/USERNAME`

- ssh into bridges: `ssh USERNAME@bridges.psc.edu`
- Try copying the datasets, either directly from the internet, `curl -L -O https://support.ceinet.ucdavis.edu/~rgponce/courses/datasets/visit/datasets.tar.gz`
  or copy it from my homedir, `cp -rv /home/rgponce/visit-examples ~`
- uncompress your datafiles: `tar -xvvzf datasets.tar.gz`
- Hit [Refresh] in VisIt's file browser window

Accessing Remote Hosts...

- [File] ➠ Open file... ⇝ click on "Host ▾"
- A list possible hosts should display, including "bridges" – select it!
- You will be asked to enter your password, check that the username is correct! Otherwise, click on [Change Username]!
- If the connection was sucessful, the window should be displaying the remote File System, eg. `Path: /home/USERNAME`

- ssh into bridges: `ssh USERNAME@bridges.psc.edu`
- Try copying the datasets, either directly from the internet, `curl -L -O https://support.scinet.utoronto.ca/~mponce/courses/datasets/visit/datasets.tar.gz`
  or copy it from my homedir, `cp -rv /home/mjponce/visit-examples ~`
- uncompress your datafiles: `tar -xvvzf datasets.tar.gz`
- Hit [Refresh] in VisIt's file browser window

Outline

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

Python scripting in VisIt I

- It is possible to run Python scripts in VisIt, from the command line without the GUI
  ```
  $ /path/to/VisIt -nowin -cli -s YOURscript.py
  ```
  it can be very useful for running batch rendering jobs on a supercomputer
- VisIt has a built-in Python 2.7 shell
  Controls → Launch CLI...
  It'll start VisIt's Python interpreter in a terminal
- Alternatively,
  Controls → Command...
  provides a text editor with Python syntax highlighting and an Execute button that tells VisIt to execute the script;
- GUI translation tool into Python code:
  Controls → Command...
  lets you record your GUI actions into Python code that you can use in your scripts

Python scripting in VisIt II

- In addition, you can use Python in
  Controls → Expressions... → Python Expression Editor
- Also,
  Controls → Query → Python Query Editor

Python Scripting: creating plots

Let's start from scratch...

1. launch a Python shell
   Controls → Launch CLI...
2. lets type the following commands (YOU NEED TO ADJUST THE PATH!):

```
1 # this opens the datafile, "noise.silo"
2 OpenDatabase("PATHtoDATA/noise.silo")
3 # this defines which type of plot we want to add
4 AddPlot("Pseudocolor" , "hardyglobal")
5 # this last command "draws" the plot, like pressing the draw button
6 DrawPlots()
```

Controlling Plots Attributes

- Each plot in VisIt has a number of attributes that control the appearance of the plot
- To access them, first create a plot attributes object by calling a function `PlotNameAttributes()`, e.g., `PseudocolorAttributes()`, or `VolumeAttributes()`
- If changing attributes, pass the object to the `SetPlotOptions()`
- If setting new defaults, pass the object to `SetDefaultPlotOptions`

```
p = PseudocolorAttributes()
p # will print out all attributes

p.min, p.max = 1, 3 # colour map range
p.minFlag, p.maxFlag = 1, 1 # turn it on
SetPlotOptions(p) # set active plot attributes

help(SetPlotOptions)
```

```
#Revert to the original colour map range:
p.minFlag , p.maxFlag = 0,0 # turn it off
SetPlotOptions(p)

#Pick a different colour map:
p. colorTableName = "Greens" # new colour map
SetPlotOptions(p)
```

Controlling Plots Attributes

- Each plot in VisIt has a number of attributes that control the appearance of the plot
- To access them, first create a plot attributes object by calling a function `PlotNameAttributes()`, e.g., `PseudocolorAttributes()`, or `VolumeAttributes()`
- If changing attributes, pass the object to the `SetPlotOptions()`
- If setting new defaults, pass the object to `SetDefaultPlotOptions`

```
1 p = PseudocolorAttributes()
2 p # will print out all attributes
3
4 p.min, p.max = 1, 3 # colour map range
5 p.minFlag, p.maxFlag = 1, 1 # turn it on
6 SetPlotOptions(p) # set active plot attributes
7
8 help(SetPlotOptions)
```

```
#Revert to the original colour map range:
p.minFlag , p.maxFlag = 0,0 # turn it off
SetPlotOptions(p)

#Pick a different colour map:
p. colorTableName = "Greens" # new colour map
SetPlotOptions(p)
```

Controlling Plots Attributes

- Each plot in VisIt has a number of attributes that control the appearance of the plot
- To access them, first create a plot attributes object by calling a function `PlotNameAttributes()`, e.g., `PseudocolorAttributes()`, or `VolumeAttributes()`
- If changing attributes, pass the object to the `SetPlotOptions()`
- If setting new defaults, pass the object to `SetDefaultPlotOptions`

```
p = PseudocolorAttributes()
p # will print out all attributes

p.min, p.max = 1, 3 # colour map range
p.minFlag, p.maxFlag = 1, 1 # turn it on
SetPlotOptions(p) # set active plot attributes

help(SetPlotOptions)
```

```
#Revert to the original colour map range:
p.minFlag , p.maxFlag = 0,0 # turn it off
SetPlotOptions(p)

#Pick a different colour map:
p. colorTableName = "Greens" # new colour map
SetPlotOptions(p)
```

Running scripts from inside the GUI

- Option 1: paste the code into the command window (Controls → Command...) and click Execute
- inside the Python shell, change to the directory containing the scripts and run the script(s)

```
os.getcwd()   # get the current directory}

os.chdir("/Users/marcelo/Downloads/datasets/scripts")

# os.chdir("C:\Users\marcelo\Desktop\datasets")    # Windows example

Source("scriptName.py")
```

Some examples...

We will assume that "noise.silo" is open!

```
# this is scratch.py
DeleteAllPlots()
AddPlot("Pseudocolor", "hardyglobal")
p = PseudocolorAttributes()
p.colorTableName = "Oranges"
SetPlotOptions(p)
DrawPlots()
```

```
>>> Source("scratch.py")
```

```
# this is addOperator.py
DeleteAllPlots()
AddPlot("Pseudocolor", "hardyglobal")
AddOperator("Isosurface")
isoAtts = IsosurfaceAttributes()
isoAtts.contourMethod = isoAtts.Value  # contour by
    value(s)
isoAtts.variable = "hardyglobal"
DrawPlots()
print isoAtts   # default is 10 isosurface levels
```

```
>>> Source("addOperator.py")
```

```
# this is threeSurfaces.py
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
```

```
>>> Source("threeSurfaces.py")
```

```
# this is saveSurfaces.py
s = SaveWindowAttributes()
s.format, s.fileName = s.PNG, 'iso'
s.outputToCurrentDirectory = 0
s.outputDirectory = "."
SetSaveWindowAttributes(s)
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
    name = SaveWindow()
```

```
>>> Source("saveSurfaces.py")
```

Some examples...

We will assume that "noise.silo" is open!

```
1 # this is scratch.py
2 DeleteAllPlots()
3 AddPlot("Pseudocolor", "hardyglobal")
4 p = PseudocolorAttributes()
5 p.colorTableName = "Oranges"
6 SetPlotOptions(p)
7 DrawPlots()
```

```
>>> Source("scratch.py")
```

```
1 # this is addOperator.py
2 DeleteAllPlots()
3 AddPlot("Pseudocolor", "hardyglobal")
4 AddOperator("Isosurface")
5 isoAtts = IsosurfaceAttributes()
6 isoAtts.contourMethod = isoAtts.Value  # contour by
      value(s)
7 isoAtts.variable = "hardyglobal"
8 DrawPlots()
9 print isoAtts   # default is 10 isosurface levels
```

```
>>> Source("addOperator.py")
```

```
1 # this is threeSurfaces.py
2 for i in range(3):
3     isoAtts.contourValue = 2. + i*1.5
4     SetOperatorOptions(isoAtts)
```

```
>>> Source("threeSurfaces.py")
```

```
1 # this is saveSurfaces.py
2 s = SaveWindowAttributes()
3 s.format, s.fileName = s.PNG, 'iso'
4 s.outputToCurrentDirectory = 0
5 s.outputDirectory = "."
6 SetSaveWindowAttributes(s)
7 for i in range(3):
8     isoAtts.contourValue = 2. + i*1.5
9     SetOperatorOptions(isoAtts)
10    name = SaveWindow()
```

```
>>> Source("saveSurfaces.py")
```

Some examples...

We will assume that "noise.silo" is open!

```
1  # this is scratch.py
2  DeleteAllPlots()
3  AddPlot("Pseudocolor", "hardyglobal")
4  p = PseudocolorAttributes()
5  p.colorTableName = "Oranges"
6  SetPlotOptions(p)
7  DrawPlots()
```

```
>>> Source("scratch.py")
```

```
1  # this is addOperator.py
2  DeleteAllPlots()
3  AddPlot("Pseudocolor", "hardyglobal")
4  AddOperator("Isosurface")
5  isoAtts = IsosurfaceAttributes()
6  isoAtts.contourMethod = isoAtts.Value  # contour by
       value(s)
7  isoAtts.variable = "hardyglobal"
8  DrawPlots()
9  print isoAtts   # default is 10 isosurface levels
```

```
>>> Source("addOperator.py")
```

```
# this is threeSurfaces.py
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
```

```
>>> Source("threeSurfaces.py")
```

```
# this is saveSurfaces.py
s = SaveWindowAttributes()
s.format, s.fileName = s.PNG, 'iso'
s.outputToCurrentDirectory = 0
s.outputDirectory = "."
SetSaveWindowAttributes(s)
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
    name = SaveWindow()
```

```
>>> Source("saveSurfaces.py")
```

Some examples...

We will assume that "noise.silo" is open!

```
# this is scratch.py
DeleteAllPlots()
AddPlot("Pseudocolor", "hardyglobal")
p = PseudocolorAttributes()
p.colorTableName = "Oranges"
SetPlotOptions(p)
DrawPlots()
```

```
>>> Source("scratch.py")
```

```
# this is addOperator.py
DeleteAllPlots()
AddPlot("Pseudocolor", "hardyglobal")
AddOperator("Isosurface")
isoAtts = IsosurfaceAttributes()
isoAtts.contourMethod = isoAtts.Value  # contour by
    value(s)
isoAtts.variable = "hardyglobal"
DrawPlots()
print isoAtts   # default is 10 isosurface levels
```

```
>>> Source("addOperator.py")
```

```
# this is threeSurfaces.py
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
```

```
>>> Source("threeSurfaces.py")
```

```
# this is saveSurfaces.py
s = SaveWindowAttributes()
s.format, s.fileName = s.PNG, 'iso'
s.outputToCurrentDirectory = 0
s.outputDirectory = "."
SetSaveWindowAttributes(s)
for i in range(3):
    isoAtts.contourValue = 2. + i*1.5
    SetOperatorOptions(isoAtts)
    name = SaveWindow()
```

```
>>> Source("saveSurfaces.py")
```

Some examples...

We will assume that "noise.silo" is open!

```
1  # this is scratch.py
2  DeleteAllPlots()
3  AddPlot("Pseudocolor", "hardyglobal")
4  p = PseudocolorAttributes()
5  p.colorTableName = "Oranges"
6  SetPlotOptions(p)
7  DrawPlots()
```

```
>>> Source("scratch.py")
```

```
1  # this is addOperator.py
2  DeleteAllPlots()
3  AddPlot("Pseudocolor", "hardyglobal")
4  AddOperator("Isosurface")
5  isoAtts = IsosurfaceAttributes()
6  isoAtts.contourMethod = isoAtts.Value  # contour by
       value(s)
7  isoAtts.variable = "hardyglobal"
8  DrawPlots()
9  print isoAtts   # default is 10 isosurface levels
```

```
>>> Source("addOperator.py")
```

```
1  # this is threeSurfaces.py
2  for i in range(3):
3      isoAtts.contourValue = 2. + i*1.5
4      SetOperatorOptions(isoAtts)
```

```
>>> Source("threeSurfaces.py")
```

```
1  # this is saveSurfaces.py
2  s = SaveWindowAttributes()
3  s.format, s.fileName = s.PNG, 'iso'
4  s.outputToCurrentDirectory = 0
5  s.outputDirectory = "."
6  SetSaveWindowAttributes(s)
7  for i in range(3):
8      isoAtts.contourValue = 2. + i*1.5
9      SetOperatorOptions(isoAtts)
10     name = SaveWindow()
```

```
>>> Source("saveSurfaces.py")
```

Some examples... Camera Motion

```
# this is printView.py
print GetView3D()   # print all its attributes of the
    current view
# GetView3D().viewNormal   # can also print a single
    attribute
```

```
>> Source("printView.py")
```

```
# this is setControlPoint.py
from math import *
c0 = View3DAttributes()
phi = 0   # 0 <= phi <= 2*pi
theta = 0   # -pi/2 <= theta <= pi/2
c0.viewNormal = (cos(theta)*cos(phi),cos(theta)*sin(
    phi),sin(theta))
c0.focus = (0, 0, 0)
c0.viewUp = (0, 0, 1)
c0.viewAngle = 30
c0.parallelScale = 17.3205
c0.nearPlane = -34.641
c0.farPlane = 34.641
c0.perspective = 1
SetView3D(c0)
```

```
>> Source("setControlPoint.py")
```

```
# this is rotateAroundVertical.py
nsteps = 300
for i in range(nsteps):
    phi = float(i)/float(nsteps-1)*2.*pi
    c0.viewNormal = (cos(theta)*cos(phi), cos(theta)*
        sin(phi),
                     sin(theta))
    SetView3D(c0)
```

```
>> Source("rotateAroundVertical.py")
```

```
# this is flyInOut.py
nsteps = 100
xfirst = 0
xlast = -30
for i in range(nsteps):
    x = xfirst + float(i)/float(nsteps-1)*(xlast-
        xfirst)
    c0.focus = (x, 0, 0)
    SetView3D(c0)
for i in range(nsteps):
    x = xlast + float(i)/float(nsteps-1)*(xfirst-xlast
        )
    c0.focus = (x, 0, 0)
    SetView3D(c0)
```

```
>> Source("flyInOut.py")
```

Some examples... Camera Motion

```
# this is printView.py
print GetView3D()   # print all its attributes of the
      current view
# GetView3D().viewNormal   # can also print a single
      attribute
```

```
>> Source("printView.py")
```

```
# this is setControlPoint.py
from math import *
c0 = View3DAttributes()
phi = 0   # 0 <= phi <= 2*pi
theta = 0   # -pi/2 <= theta <= pi/2
c0.viewNormal = (cos(theta)*cos(phi),cos(theta)*sin(
    phi),sin(theta))
c0.focus = (0, 0, 0)
c0.viewUp = (0, 0, 1)
c0.viewAngle = 30
c0.parallelScale = 17.3205
c0.nearPlane = -34.641
c0.farPlane = 34.641
c0.perspective = 1
SetView3D(c0)
```

```
>> Source("setControlPoint.py")
```

```
# this is rotateAroundVertical.py
nsteps = 300
for i in range(nsteps):
    phi = float(i)/float(nsteps-1)*2.*pi
    c0.viewNormal = (cos(theta)*cos(phi), cos(theta)*
        sin(phi),
                     sin(theta))
    SetView3D(c0)
```

```
>> Source("rotateAroundVertical.py")
```

```
# this is flyInOut.py
nsteps = 100
xfirst = 0
xlast = -30
for i in range(nsteps):
    x = xfirst + float(i)/float(nsteps-1)*(xlast-
        xfirst)
    c0.focus = (x, 0, 0)
    SetView3D(c0)
for i in range(nsteps):
    x = xlast + float(i)/float(nsteps-1)*(xfirst-xlast
        )
    c0.focus = (x, 0, 0)
    SetView3D(c0)
```

```
>> Source("flyInOut.py")
```

Some examples... Camera Motion

```
# this is printView.py
print GetView3D()   # print all its attributes of the
      current view
# GetView3D().viewNormal   # can also print a single
      attribute
```

```
>> Source("printView.py")
```

```
# this is setControlPoint.py
from math import *
c0 = View3DAttributes()
phi = 0   # 0 <= phi <= 2*pi
theta = 0   # -pi/2 <= theta <= pi/2
c0.viewNormal = (cos(theta)*cos(phi),cos(theta)*sin(
      phi),sin(theta))
c0.focus = (0, 0, 0)
c0.viewUp = (0, 0, 1)
c0.viewAngle = 30
c0.parallelScale = 17.3205
c0.nearPlane = -34.641
c0.farPlane = 34.641
c0.perspective = 1
SetView3D(c0)
```

```
>> Source("setControlPoint.py")
```

```
# this is rotateAroundVertical.py
nsteps = 300
for i in range(nsteps):
    phi = float(i)/float(nsteps-1)*2.*pi
    c0.viewNormal = (cos(theta)*cos(phi), cos(theta)*
          sin(phi),
                     sin(theta))
    SetView3D(c0)
```

```
>> Source("rotateAroundVertical.py")
```

```
# this is flyInOut.py
nsteps = 100
xfirst = 0
xlast = -30
for i in range(nsteps):
    x = xfirst + float(i)/float(nsteps-1)*(xlast-
          xfirst)
    c0.focus = (x, 0, 0)
    SetView3D(c0)
for i in range(nsteps):
    x = xlast + float(i)/float(nsteps-1)*(xfirst-xlast
          )
    c0.focus = (x, 0, 0)
    SetView3D(c0)
```

```
>> Source("flyInOut.py")
```

Some examples... Camera Motion

```
# this is printView.py
print GetView3D()   # print all its attributes of the
      current view
# GetView3D().viewNormal   # can also print a single
      attribute
```

```
>> Source("printView.py")
```

```
# this is setControlPoint.py
from math import *
c0 = View3DAttributes()
phi = 0   # 0 <= phi <= 2*pi
theta = 0   # -pi/2 <= theta <= pi/2
c0.viewNormal = (cos(theta)*cos(phi),cos(theta)*sin(
      phi),sin(theta))
c0.focus = (0, 0, 0)
c0.viewUp = (0, 0, 1)
c0.viewAngle = 30
c0.parallelScale = 17.3205
c0.nearPlane = -34.641
c0.farPlane = 34.641
c0.perspective = 1
SetView3D(c0)
```

```
>> Source("setControlPoint.py")
```

```
# this is rotateAroundVertical.py
nsteps = 300
for i in range(nsteps):
    phi = float(i)/float(nsteps-1)*2.*pi
    c0.viewNormal = (cos(theta)*cos(phi), cos(theta)*
          sin(phi),
                     sin(theta))
    SetView3D(c0)
```

```
>> Source("rotateAroundVertical.py")
```

```
# this is flyInOut.py
nsteps = 100
xfirst = 0
xlast = -30
for i in range(nsteps):
    x = xfirst + float(i)/float(nsteps-1)*(xlast-
          xfirst)
    c0.focus = (x, 0, 0)
    SetView3D(c0)
for i in range(nsteps):
    x = xlast + float(i)/float(nsteps-1)*(xfirst-xlast
          )
    c0.focus = (x, 0, 0)
    SetView3D(c0)
```

```
>> Source("flyInOut.py")
```

Some examples... Camera Motion

```
# this is printView.py
print GetView3D()   # print all its attributes of the
      current view
# GetView3D().viewNormal   # can also print a single
      attribute
```

```
>> Source("printView.py")
```

```
# this is setControlPoint.py
from math import *
c0 = View3DAttributes()
phi = 0   # 0 <= phi <= 2*pi
theta = 0   # -pi/2 <= theta <= pi/2
c0.viewNormal = (cos(theta)*cos(phi),cos(theta)*sin(
      phi),sin(theta))
c0.focus = (0, 0, 0)
c0.viewUp = (0, 0, 1)
c0.viewAngle = 30
c0.parallelScale = 17.3205
c0.nearPlane = -34.641
c0.farPlane = 34.641
c0.perspective = 1
SetView3D(c0)
```

```
>> Source("setControlPoint.py")
```

```
# this is rotateAroundVertical.py
nsteps = 300
for i in range(nsteps):
    phi = float(i)/float(nsteps-1)*2.*pi
    c0.viewNormal = (cos(theta)*cos(phi), cos(theta)*
          sin(phi),
                     sin(theta))
    SetView3D(c0)
```

```
>> Source("rotateAroundVertical.py")
```

```
# this is flyInOut.py
nsteps = 100
xfirst = 0
xlast = -30
for i in range(nsteps):
    x = xfirst + float(i)/float(nsteps-1)*(xlast-
          xfirst)
    c0.focus = (x, 0, 0)
    SetView3D(c0)
for i in range(nsteps):
    x = xlast + float(i)/float(nsteps-1)*(xfirst-xlast
          )
    c0.focus = (x, 0, 0)
    SetView3D(c0)
```

```
>> Source("flyInOut.py")
```

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

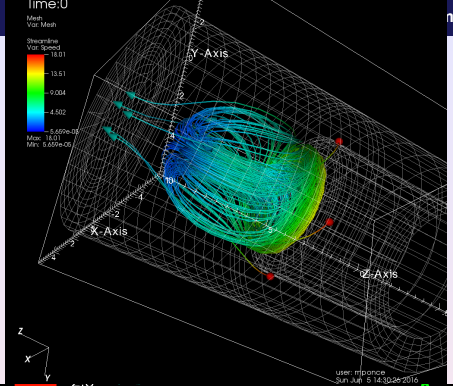
2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

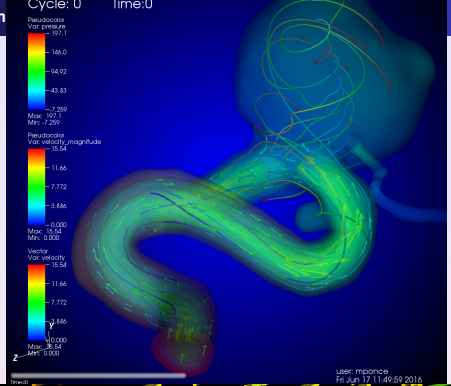
3 Summary

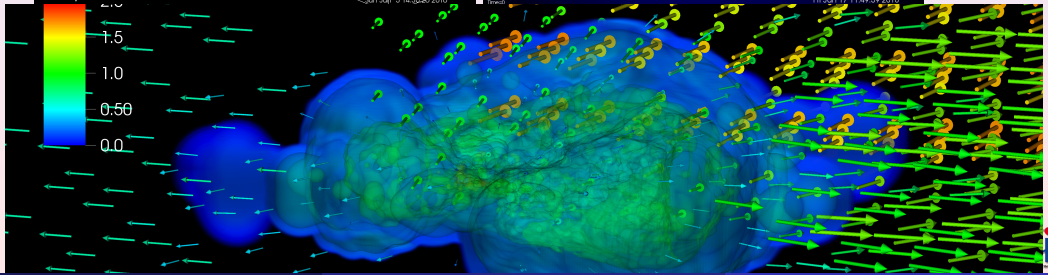
4 Further Resources
- References

Section 3

Summary


Time:0
Mesh
Var: Mesh
Streamline
Var: Speed
18.01
13.51
9.004
4.502
Max: 18.01
Min: 8.659e-05
Y-Axis
X-Axis
Z-Axis
user: mponce
Sun Jun 5 14:39:26 2016


Cycle: 0
Time:0
Pseudocolor
Var: pressure
197.1
146.0
94.92
43.83
-7.259
Max: 197.1
Min: -7.259
Pseudocolor
Var: velocity_magnitude
15.54
11.66
7.772
3.886
0.000
Max: 15.54
Min: 0.000
Vector
Var: velocity
15.54
11.66
7.772
3.886
0.000
Max: 15.54
Min: 0.000
user: mponce
Fri Jun 17 11:49:59 2016


2.0
1.5
1.0
0.50
0.0

1 Scientific Visualization
- 2D/3D Visualization Generalities
- Visualization Pipeline

2 VisIt
- Generalities
- Viz. Pipelines Basics
- Movies Generation
- Remote Visualization
- Scripting

3 Summary

4 Further Resources
- References

Outline

1. Scientific Visualization
   - 2D/3D Visualization Generalities
   - Visualization Pipeline
2. VisIt
   - Generalities
   - Viz. Pipelines Basics
   - Movies Generation
   - Remote Visualization
   - Scripting
3. Summary
4. Further Resources
   - References

    VisIt

- Website:
  https://wci.llnl.gov/simulation/computer-codes/visit/
  https://wci.llnl.gov/codes/visit/
- Documentation:
  https://wci.llnl.gov/simulation/computer-codes/visit/manuals
- VisIt User Manual: http://visit-sphinx-user-manual.readthedocs.io
- Gallery: https://wci.llnl.gov/simulation/computer-codes/visit/gallery
- Visit Users wiki: http://www.visitusers.org
- Tutorials: http://www.visitusers.org/index.php?title=VisIt_Tutorial
- Examples Datasets: http://www.visitusers.org/index.php?title=Tutorial_Data

- ParaView Tutorial
  - http://www.paraview.org/Wiki/The_ParaView_Tutorial

- CI-Tutor: Cyber Infrastructure Tutor
  - http://www.citutor.org "Introduction to Visualization"

- specifics or general questions about viz...
  - courses_AT_scinet•utoronto•ca
  - support_AT_scinet•utoronto•ca

SciNet

5 Additional Material

- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

Outline

5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

VTK legacy format

- in the shell, try

```
$ file testRectilinearGrid.vtk
```

```
testRectilinearGrid.vtk:  ASCII C program text, with very long lines, with CLRF line terminators
```

  - ASCII file ...

- try to look at it...

```
$ more testRectilinearGrid.vtk
```

VTK legacy format

- in the shell, try

```
$ file testRectilinearGrid.vtk
```

```
testRectilinearGrid.vtk:  ASCII C program text, with very long lines, with CLRF line terminators
```

  - ASCII file ...

- try to look at it...

```
$ more testRectilinearGrid.vtk
```

VTK legacy format

```
# vtk DataFile Version 2.0
Rectilinear grid of temperature values
ASCII
DATASET RECTILINEAR_GRID
DIMENSIONS 100 100 1
X_COORDINATES 100 float
0 1 2 3 4 5 6 7 8 9 10  11  12  13  14  15  16  17  18  19  20  21  22
       23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
      39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
      55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70
      71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86
      87  88  89  90  91  92  93  94  95  96  97  98  99
Y_COORDINATES 100 float
0 1 2 3 4 5 6 7 8 9 10  11  12  13  14  15  16  17  18  19  20  21  22
       23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
      39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
      55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70
      71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86
      87  88  89  90  91  92  93  94  95  96  97  98  99
Z_COORDINATES 1 float
0
POINT_DATA 10000
SCALARS temperature float
LOOKUP_TABLE default
20.18 20.36 20.54 20.73 20.93 21.13 21.35 21.58 21.82 22.09 22.38 22.70
       23.06 23.46 23.92 24.44 25.05 25.77 26.63 27.68 28.99 30.68
      32.90 35.99 40.50 47.61 60.00 84.65 142.03 300.00 300.00 300.00
      300.00 300.00 300.00 300.00 300.00 300.00 300.00 300.00 300.00
```

(1) `header`

(2) `title` (upto 256 characters)

(3) `Data type`:
ASCII or Binary

(4) `Geometry/Toplogy`,
5 types:
STRUCTURED_POINTS,
STRUCTURED_GRID,
UNSTRUCTURED_GRID,
POLYDATA,
RECTILINEAR_GRID,
FIELD

(5) `Dataset attributes`:
data

Outline

5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

Outline

5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

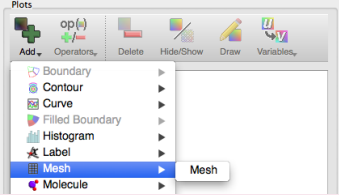
VisIt: Aneurism DataSet

This tutorial uses the aneurysm dataset

- Launch VisIt
- In VisIt's GUI, under the Sources section, click Open
- Navigate your file system to select the "aneurysm.visit" file.
- Alternatively, navigate into the aneurism directory and load all the "*.silo" files.

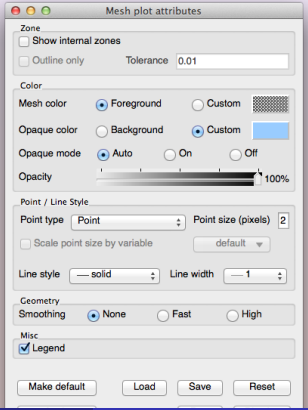
VisIt: Plotting Mesh Topology

- First we will examine the finite element mesh used in the blood flow simulation.
- Add → Mesh → **Mesh**



- click draw

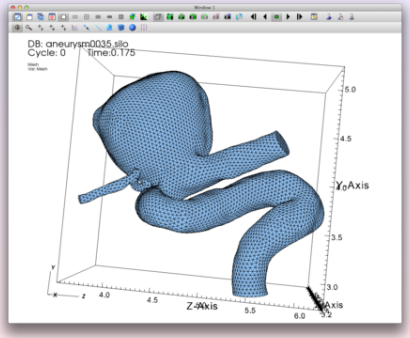
- Expand the Mesh plot object and double click to open the Mesh Plot Attributes Window



VisIt: Plotting Mesh Topology

Experiment with settings for:

- Mesh color
- Opaque color
- Opaque mode
- Show internal zones
- You will need to click Apply to commit the settings to your plot.



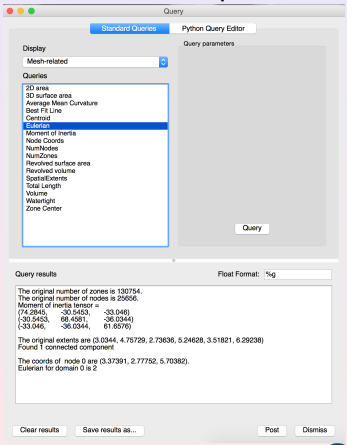
VisIt: Query Mesh Properties

VisIt's Query interface provides several quantitative data summarization operations.

- [Controls Menu] → Query
- Select NumZones and click Query
  ➠ **number of elements** in the mesh
- Select NumNodes and click Query
  ➠ **number of vertices** in the mesh

Pop quiz

- How many elements are used to construct the mesh?
- How many vertices are used to construct the mesh?



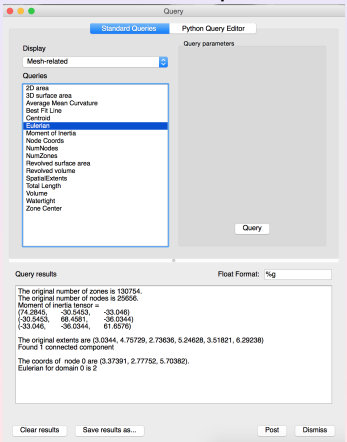
VisIt: Query Mesh Properties

VisIt's Query interface provides several quantitative data summarization operations.

- [Controls Menu] → Query
- Select NumZones and click Query
  ➠ **number of elements** in the mesh
- Select NumNodes and click Query
  ➠ **number of vertices** in the mesh

Pop quiz

- How many elements are used to construct the mesh?
- How many vertices are used to construct the mesh?



VisIt: Examining and Identifying Data Fields

In addition to the mesh topology, this dataset provides two mesh fields:

- A *scalar field* **pressure**, associated with the mesh vertices.
- A *vector field* **velocity**, associated with the mesh vertices.
- [File Menu] → File information...

VisIt automatically defines an expression that allows us to use the *magnitude of the velocity vector field* as a *scalar field* on the mesh

➠ the result of the expression is a new field named **velocity_magnitude**

VisIt: Examining and Identifying Data Fields

In addition to the mesh topology, this dataset provides two mesh fields:

- A *scalar field* **pressure**, associated with the mesh vertices.
- A *vector field* **velocity**, associated with the mesh vertices.
- [File Menu] → File information...

VisIt automatically defines an expression that allows us to use the *magnitude of the velocity vector field* as a *scalar field* on the mesh

➠ the result of the expression is a new field named **velocity_magnitude**

VisIt: Examining and Identifying Data Fields

In addition to the mesh topology, this dataset provides two mesh fields:

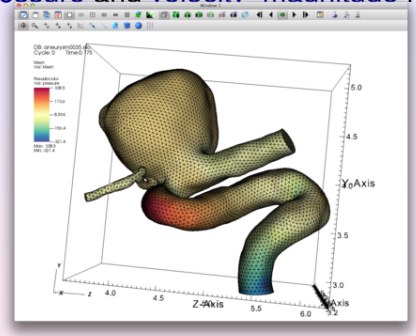
- A *scalar field* **pressure**, associated with the mesh vertices.
- A *vector field* **velocity**, associated with the mesh vertices.
- [File Menu] → File information...

VisIt automatically defines an expression that allows us to use the *magnitude of the velocity vector field* as a *scalar field* on the mesh
➠ the result of the expression is a new field named **velocity_magnitude**

VisIt: Visualizing Scalar Fields

We will use **Pseudocolor Plots** to examine the pressure and velocity magnitude fields.

- Add → Pseudocolor → Pressure
- Expand the Pseudocolor plot and double click to bring up the Pseudocolor Plot Attributes Window.
- Change the color table to **Spectral** and check the **Invert** button
- Click Apply
- [Plot List] click Draw
- [Time Slider] click Play / Stop



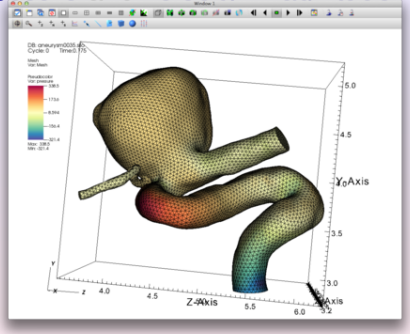
Experiment with:

- setting the Pseudocolor plot limits
- hiding/showing the Mesh plot

VisIt: Visualizing Scalar Fields

We will use **Pseudocolor Plots** to examine the pressure and velocity magnitude fields.

- Add → **Pseudocolor** → Pressure
- Expand the Pseudocolor plot and double click to bring up the Pseudocolor Plot Attributes Window.
- Change the color table to **Spectral** and check the **Invert** button
- Click Apply
- [Plot List] click Draw
- [Time Slider] click Play / Stop



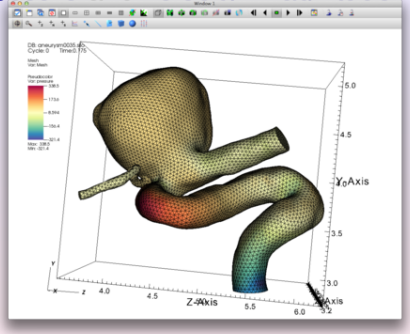
Experiment with:

- setting the Pseudocolor plot limits
- hiding/showing the Mesh plot

VisIt: Visualizing Scalar Fields

We will use **Pseudocolor Plots** to examine the pressure and velocity magnitude fields.

- Add → **Pseudocolor** → Pressure
- Expand the Pseudocolor plot and double click to bring up the Pseudocolor Plot Attributes Window.
- Change the color table to **Spectral** and check the **Invert** button
- Click Apply
- [Plot List] click Draw
- [Time Slider] click Play / Stop



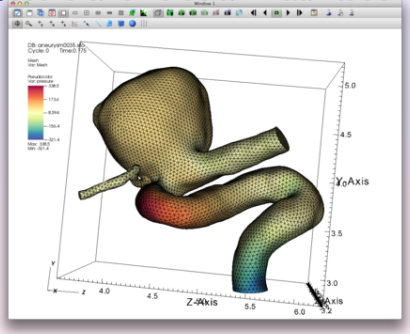
Experiment with:

- setting the Pseudocolor plot limits
- hiding/showing the Mesh plot

VisIt: Visualizing Scalar Fields

We will use **Pseudocolor Plots** to examine the pressure and velocity magnitude fields.

- Add → **Pseudocolor** → Pressure
- Expand the Pseudocolor plot and double click to bring up the Pseudocolor Plot Attributes Window.
- Change the color table to **Spectral** and check the **Invert** button
- Click Apply
- [Plot List] click Draw
- [Time Slider] click Play / Stop



Experiment with:

- setting the Pseudocolor plot limits
- hiding/showing the Mesh plot

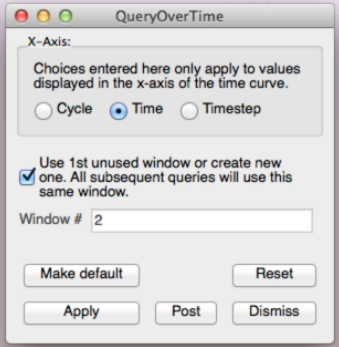
VisIt: Analizing Scalar Fields

Query the Maximum Pressure Over Time

We can use the pressure field to extract the heart beat signal.
We want to find the maximum pressure value across the mesh elements at each time step of our dataset.

➡ VisIt provides a **Query over time** mechanism that

- First, we need to set our query options to use time as the independent variable for our query: [Controls Menu] → Query over time options
- Select **time**
- click Apply and dismiss the window

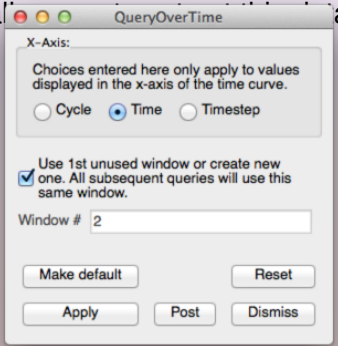


VisIt: Analizing Scalar Fields

Query the Maximum Pressure Over Time

We can use the pressure field to extract the heart beat signal.
We want to find the maximum pressure value across the mesh elements at each time step of our dataset.

- VisIt provides a **Query over time** mechanism that al a.
  - First, we need to set our query options to use time as the independent variable for our query: [Controls Menu] → Query over time options
  - Select **time**
  - click Apply and dismiss the window

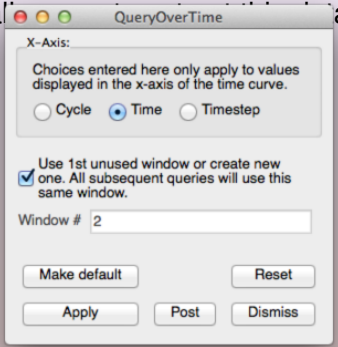


VisIt: Analizing Scalar Fields

Query the Maximum Pressure Over Time

We can use the pressure field to extract the heart beat signal.
We want to find the maximum pressure value across the mesh elements at each time step of our dataset.

- VisIt provides a **Query over time** mechanism that al… a.
  - First, we need to set our query options to use time as the independent variable for our query: [Controls Menu] → Query over time options
  - Select **time**
  - click Apply and dismiss the window

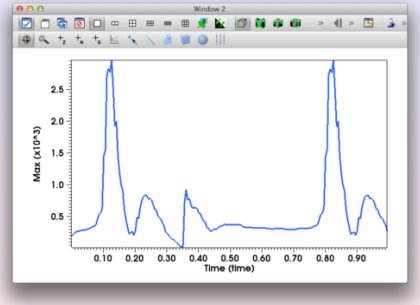


VisIt: Analizing Scalar Fields

Now we can execute the **Max query** on all the time steps and collect the results into a curve.

- [Plot List] Click to make sure the **Pseudocolor plot** is active
- [Controls Menu] → Queries
- Select **Max**
- check Do Time Query

This will process the simulation output files and create a new window with a curve that contains the maximum pressure value at each time.



Exercises

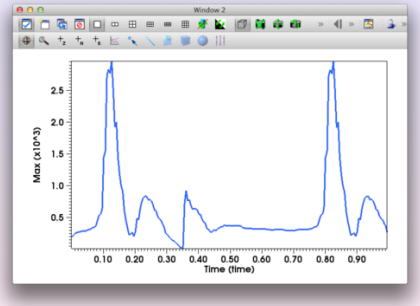
- How many heart beats does this dataset cover?
- Estimate the number of beats per minute of the simulated heart.

VisIt: Analizing Scalar Fields

Now we can execute the **Max query** on all the time steps and collect the results into a curve.

- [Plot List] Click to make sure the **Pseudocolor plot** is active
- [Controls Menu] → Queries
- Select **Max**
- check Do Time Query

This will process the simulation output files and create a new window with a curve that contains the maximum pressure value at each time.



Exercises

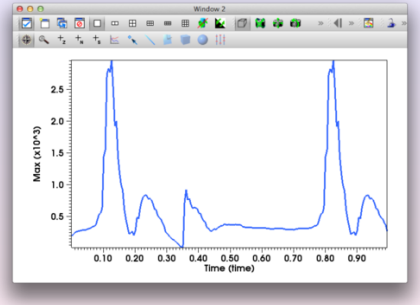
- How many heart beats does this dataset cover?
- Estimate the number of beats per minute of the simulated heart.

VisIt: Analizing Scalar Fields

Now we can execute the **Max query** on all the time steps and collect the results into a curve.

- [Plot List] Click to make sure the **Pseudocolor plot** is active
- [Controls Menu] → Queries
- Select **Max**
- check Do Time Query

This will process the simulation output files and create a new window with a curve that contains the maximum pressure value at each time.



Exercises

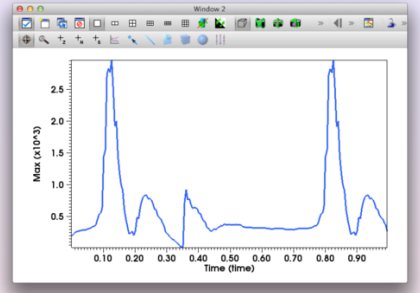
- How many heart beats does this dataset cover?
- Estimate the number of beats per minute of the simulated heart.

VisIt: Analizing Scalar Fields

Now we can execute the **Max query** on all the time steps and collect the results into a curve.

- [Plot List] Click to make sure the **Pseudocolor plot** is active
- [Controls Menu] → Queries
- Select **Max**
- check Do Time Query

This will process the simulation output files and create a new window with a curve that contains the maximum pressure value at each time.



Exercises

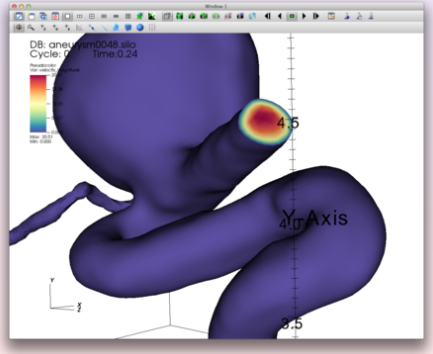
- How many heart beats does this dataset cover?
- Estimate the number of beats per minute of the simulated heart.

VisIt: Analizing Scalar Fields

Contours and Sub-volumes of High Velocity

Next, we will create a **Pseudocolor plot** to look at the *magnitude of the velocity vector field*.

- First, Hide / Delete previous plots
- Add a **Pseudocolor Plot** of the *velocity_magnitude*: [Plot List] → Add → **Pseudocolor** → velocity_magnitude
- Open (double-click) **Pseudocolor Plot Attributes Window**, and set the color table options as before
- [Plot List] → Draw

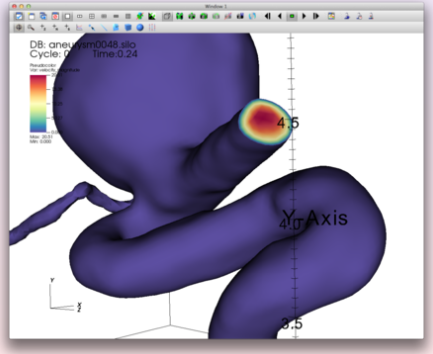


VisIt: Analizing Scalar Fields

Contours and Sub-volumes of High Velocity

Next, we will create a **Pseudocolor plot** to look at the *magnitude of the velocity vector field*.

- First, Hide / Delete previous plots
- Add a **Pseudocolor Plot** of the *velocity_magnitude*: [Plot List] → Add → **Pseudocolor** → velocity_magnitude
- Open (double-click) **Pseudocolor Plot Attributes Window**, and set the color table options as before
- [Plot List] → Draw

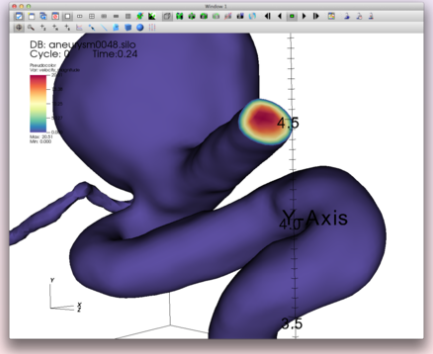


VisIt: Analizing Scalar Fields

Contours and Sub-volumes of High Velocity

Next, we will create a **Pseudocolor plot** to look at the *magnitude of the velocity vector field*.

- First, Hide / Delete previous plots
- Add a **Pseudocolor Plot** of the *velocity_magnitude*: [Plot List] → Add → **Pseudocolor** → velocity_magnitude
- Open (double-click) **Pseudocolor Plot Attributes Window**, and set the color table options as before
- [Plot List] → Draw



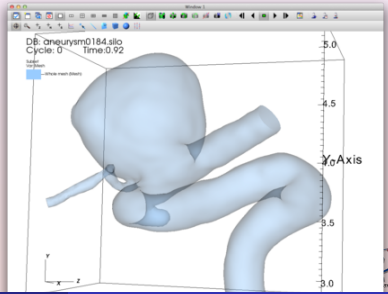
VisIt: Analizing Scalar Fields

Notice that the velocity at the surface of the mesh is **zero**. To get a better understanding of the flow inside the mesh, we will use operators to extract regions of *high blood flow*.

Creating a Semi-Transparent Exterior Mesh Plot

When looking at features inside the mesh, it helps to have a partially transparent view of the whole mesh boundary for reference.
We will add a Subset plot to create this view of the mesh boundary.

- [Plot List] → **Uncheck** Apply operators to all plots
- Add a **Subset Plot** of the *mesh*: [Plot List] → Add → **Subset** → Mesh
- Open (double-click) **Subset Plot Attributes Window**: change color to **LightBlue**, **Opacity** ⇝ 25%, hit Apply



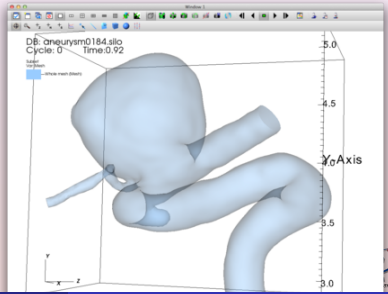
VisIt: Analizing Scalar Fields

Notice that the velocity at the surface of the mesh is **zero**. To get a better understanding of the flow inside the mesh, we will use operators to extract regions of *high blood flow*.

Creating a Semi-Transparent Exterior Mesh Plot

When looking at features inside the mesh, it helps to have a partially transparent view of the whole mesh boundary for reference.
We will add a Subset plot to create this view of the mesh boundary.

- [Plot List] → **Uncheck** Apply operators to all plots
- Add a **Subset Plot** of the *mesh*: [Plot List] → Add → **Subset** → Mesh
- Open (double-click) **Subset Plot Attributes Window**: change color to **LightBlue**, **Opacity** ⇝ 25%, hit Apply



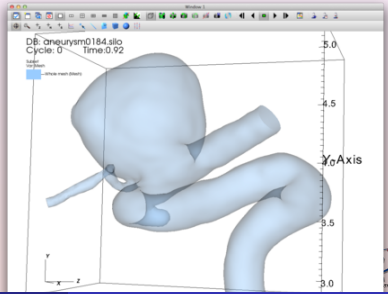
VisIt: Analizing Scalar Fields

Notice that the velocity at the surface of the mesh is **zero**. To get a better understanding of the flow inside the mesh, we will use operators to extract regions of *high blood flow*.

Creating a Semi-Transparent Exterior Mesh Plot

When looking at features inside the mesh, it helps to have a partially transparent view of the whole mesh boundary for reference.
We will add a Subset plot to create this view of the mesh boundary.

- [Plot List] → **Uncheck** Apply operators to all plots
- Add a **Subset Plot** of the *mesh*: [Plot List] → Add → **Subset** → Mesh
- Open (double-click) **Subset Plot Attributes Window**: change color to **LightBlue**, **Opacity** ⇝ **25%**, hit Apply

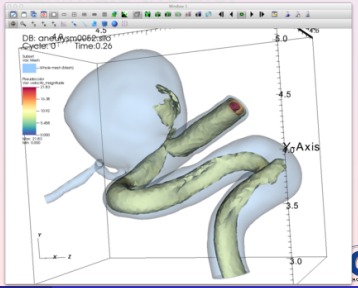


VisIt: Analizing Scalar Fields

Contours of High Velocity

Now we will extract **contour surfaces** at *high velocity* values using the **IsoSurface Operator**

- [Plot List] → Click to select the Pseudocolor Plot
- Add an **IsoSurface Operator**: [Plot List] → Operators → Slicing → IsoSurface
- Open (double-click) **IsoSurface Operator Attributes Window**: set Select by ⇝ **Value**, and use **10 15 20**; click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot

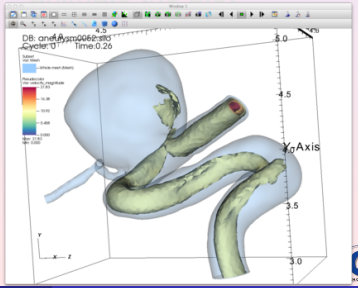


VisIt: Analizing Scalar Fields

Contours of High Velocity

Now we will extract **contour surfaces** at *high velocity* values using the **IsoSurface Operator**

- [Plot List] → Click to select the Pseudocolor Plot
- Add an **IsoSurface Operator**: [Plot List] → Operators → Slicing → **IsoSurface**
- Open (double-click) **IsoSurface Operator Attributes Window**: set Select by →**Value**, and use **10 15 20**; click Apply and dismiss the window



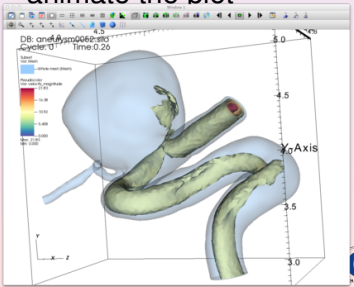
VisIt: Analizing Scalar Fields

Contours of High Velocity

Now we will extract **contour surfaces** at *high velocity* values using the **IsoSurface Operator**

- [Plot List] → Click to select the Pseudocolor Plot
- Add an **IsoSurface Operator**:
  [Plot List] → Operators → Slicing → **IsoSurface**
- Open (double-click) **IsoSurface Operator Attributes Window**:
  set Select by ⇝**Value**, and use **10 15 20**;
  click Apply and dismiss the window

- [Plot List] → Draw
- use the [Time Slider] to animate the plot

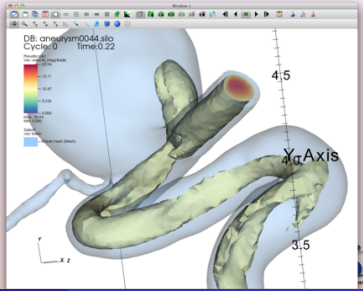


VisIt: Analizing Scalar Fields

Sub-Volumes of High Velocity

As an alternative to *contours*, we can also extract the **sub-volume** between two scalar values using the **IsoVolume Operator**

- Remove / Hide the 'IsoSurface Operator'
- Add an **IsoVolume Operator**: [Plot List] → Operators → Selection → IsoVolume
- Open (double-click) **IsoVolume Operator Attributes Window**: set the Lower Bound ⇝**10**, and Upper Bound ⇝**20**; click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot

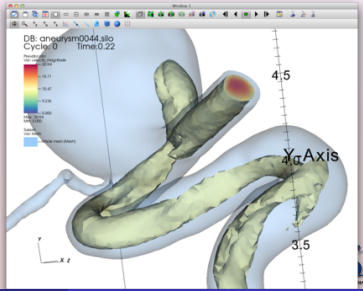


VisIt: Analizing Scalar Fields

Sub-Volumes of High Velocity

As an alternative to *contours*, we can also extract the **sub-volume** between two scalar values using the **IsoVolume Operator**

- Remove / Hide the 'IsoSurface Operator'
- Add an **IsoVolume Operator**:
  [Plot List] → Operators → Selection → **IsoVolume**
- Open (double-click) **IsoVolume Operator Attributes Window**: set the Lower Bound ⇝**10**, and Upper Bound ⇝**20**; click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot

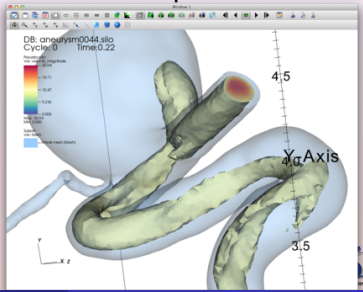


VisIt: Analizing Scalar Fields

Sub-Volumes of High Velocity

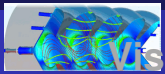
As an alternative to *contours*, we can also extract the **sub-volume** between two scalar values using the **IsoVolume Operator**

- Remove / Hide the 'IsoSurface Operator'
- Add an **IsoVolume Operator**: [Plot List] → Operators → Selection → **IsoVolume**
- Open (double-click) **IsoVolume Operator Attributes Window**: set the Lower Bound ⇝**10**, and Upper Bound ⇝**20**; click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot



Outline

5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

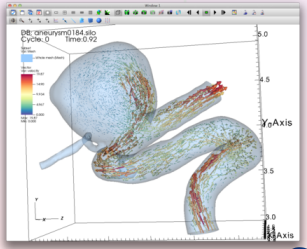


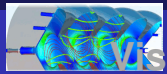
VisIt: Vector Fields Visualization

Plotting the Vector Field Directly with Glyphs

VisIt's vector plot renders a vector field at each time step as a collection of Arrow Glyphs. This allows us to see the direction of the vectors as well as their magnitude.

- Add a **Vector Plot** of *velocity*:
  [Plot List]: Add → **Vector** → velocity
- Open **Vector Plot Attribute Window**:
  [Vector Tab] set Stride ⇝**5**,
  [Data Tab] set Color ⇝**Spectrum+Inv.**,
  [Glyphs Tab] set Scale ⇝**0.5**,
  Arrow Body ⇝**cylinder**,
  Geommetry Quality ⇝**High**
  click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot



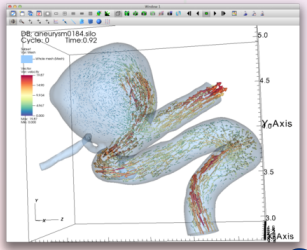


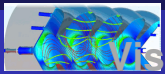
VisIt: Vector Fields Visualization

Plotting the Vector Field Directly with Glyphs

VisIt's vector plot renders a vector field at each time step as a collection of Arrow Glyphs. This allows us to see the direction of the vectors as well as their magnitude.

- Add a **Vector Plot** of *velocity*:
  [Plot List]: Add → **Vector** → velocity
- Open **Vector Plot Attribute Window**:
  [Vector Tab] set Stride ⇝**5**,
  [Data Tab] set Color ⇝**Spectrum+Inv.**,
  [Glyphs Tab] set Scale ⇝**0.5**,
  Arrow Body ⇝**cylinder**,
  Geommetry Quality ⇝**High**
  click Apply and dismiss the window
- [Plot List] → Draw
- use the [Time Slider] to animate the plot





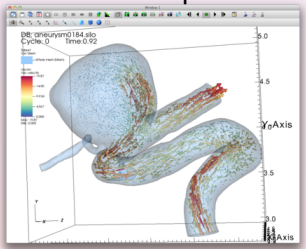
VisIt: Vector Fields Visualization

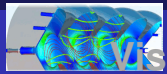
Plotting the Vector Field Directly with Glyphs

VisIt's vector plot renders a vector field at each time step as a collection of Arrow Glyphs. This allows us to see the direction of the vectors as well as their magnitude.

- Add a **Vector Plot** of *velocity*:
  [Plot List]: Add → **Vector** → velocity
- Open **Vector Plot Attribute Window**:
  [Vector Tab] set Stride ⇝**5**,
  [Data Tab] set Color ⇝**Spectrum+Inv.**,
  [Glyphs Tab] set Scale ⇝**0.5**,
  Arrow Body ⇝**cylinder**,
  Geommetry Quality ⇝**High**
  click Apply and dismiss the window

- [Plot List] → Draw
- use the [Time Slider] to animate the plot





VisIt: Vector Fields Viz – streamlines

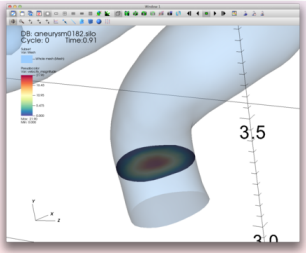
Examining features of the Flow Field with Streamlines

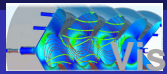
To explore the flow field further we will seed and advect a set of streamlines near the inflow of the artery.

Streamlines show the path massless tracer particles would take if advected by a static vector field.

To construct Streamlines, the first step is selecting a set of spatial locations that can serve as the initial seed points.

We want to center our seed points around the peak velocity value on a slice near the inflow of the artery. To find this location, we query a sliced pseudocolor plot of the velocity_magnitude

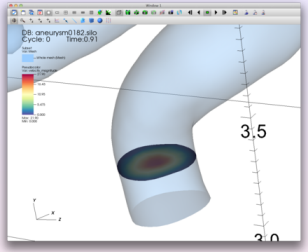


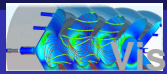


VisIt: Vector Fields Viz – streamlines

Slicing the blood flow:

- Add a **Pseudocolor Plot** of *velocity_magnitude*:
  [Plot List]:
  Add → **Pseudocolor** → velocity_magnitude
- Open **Pseudocolor Plot Attributes Window**:
  [Data Tab] set Color ↝ **Spectral**
- Add a **Slice Operator**:
  [Plot List]: Operators → **Slicing** → slice
- Open **Slice Operator Attributes Window**:
  [Normal] set Orthogonal ↝ **Y axis**.
  [Origin] set Point ↝ 3 3 3.
  [Up Axis] uncheck Project to 2D click Apply and dismiss the window

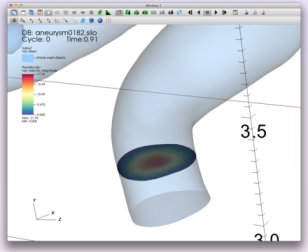


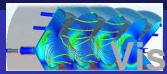


VisIt: Vector Fields Viz – streamlines

Slicing the blood flow:

- Add a **Pseudocolor Plot** of *velocity_magnitude*:
  [Plot List]:
  Add → **Pseudocolor** → velocity_magnitude
- Open **Pseudocolor Plot Attributes Window**:
  [Data Tab] set Color ⇝ **Spectral**
- Add a **Slice Operator**:
  [Plot List]: Operators → **Slicing** → slice
- Open **Slice Operator Attributes Window**:
  [Normal] set Orthogonal ⇝ **Y axis**,
  [Origin] set Point ⇝ **3 3 3**,
  [Up Axis] uncheck Project to 2D click Apply and dismiss the window

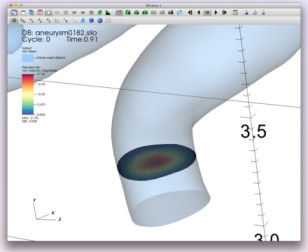


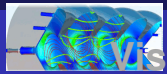


VisIt: Vector Fields Viz – streamlines

Slicing the blood flow:

- Add a **Pseudocolor Plot** of *velocity_magnitude*:
  [Plot List]:
  Add → **Pseudocolor** → velocity_magnitude
- Open **Pseudocolor Plot Attributes Window**:
  [Data Tab] set Color ⇝**Spectral**
- Add a **Slice Operator**:
  [Plot List]: Operators → **Slicing** → slice
- Open **Slice Operator Attributes Window**:
  [Normal] set Orthogonal ⇝**Y axis**,
  [Origin] set Point ⇝**3 3 3**,
  [Up Axis] uncheck Project to 2D click Apply and dismiss the window

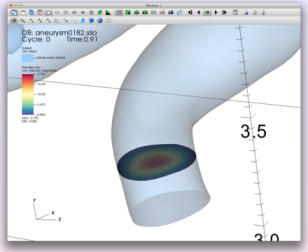


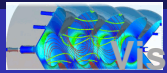


VisIt: Vector Fields Viz – streamlines

Slicing the blood flow:

- Add a **Pseudocolor Plot** of *velocity_magnitude*:
  [Plot List]:
  Add → **Pseudocolor** → velocity_magnitude
- Open **Pseudocolor Plot Attributes Window**:
  [Data Tab] set Color ⇝**Spectral**
- Add a **Slice Operator**:
  [Plot List]: Operators → **Slicing** → slice
- Open **Slice Operator Attributes Window**:
  [Normal] set Orthogonal ⇝**Y axis**,
  [Origin] set Point ⇝**3 3 3**,
  [Up Axis] uncheck Project to 2D click Apply and dismiss the window

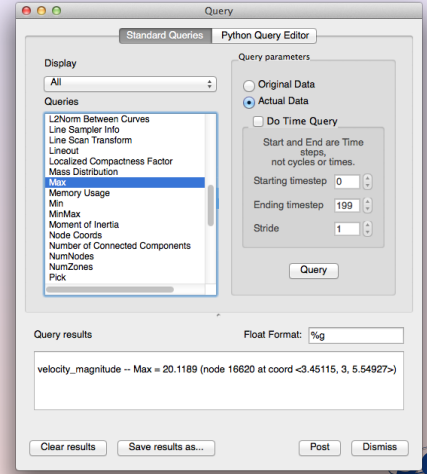


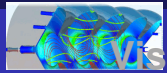


VisIt: Vector Fields Viz – streamlines

Query to find the Maximum Velocity on the Slice

- [Plot List]: click on the pseudocolor [velocity_magnitude] to make is **active**
- Go to the **Controls Menu**: [Max] select [Actual Data] ⇝ **Query**
- This will return the *maximum scalar value on the slice* and the x,y,z coordinates of the node associated with this value.
- We will use the x,y,z coordinates of this node to seed a set of streamlines.

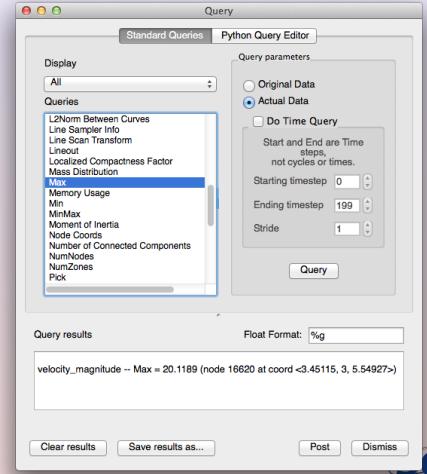


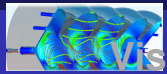


VisIt: Vector Fields Viz – streamlines

Query to find the Maximum Velocity on the Slice

- [Plot List]: click on the pseudocolor velocity_magnitude to make is **active**
- Go to the **Controls Menu**: [Max] select Actual Data ⇝ **Query**
- This will return the *maximum scalar value on the slice* and the x, y, z coordinates of the node associated with this value.
- We will use the x,y,z coordinates of this node to seed a set of streamlines.

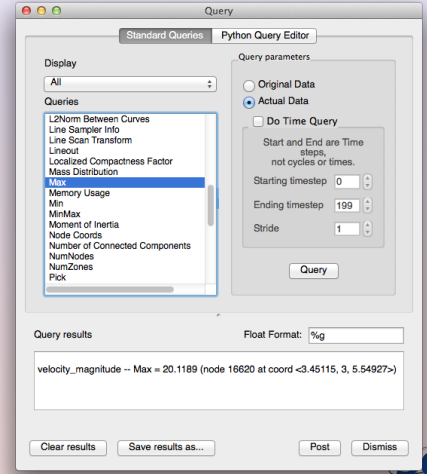


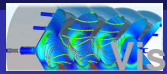


VisIt: Vector Fields Viz – streamlines

Query to find the Maximum Velocity on the Slice

- [Plot List]: click on the pseudocolor velocity_magnitude to make is **active**
- Go to the **Controls Menu**: [Max] select Actual Data ⇝**Query**
- ⇝ This will return the *maximum scalar value on the slice* and the x, y, z coordinates of the node associated with this value.
- ⇝ We will use the x,y,z coordinates of this node to seed a set of streamlines.





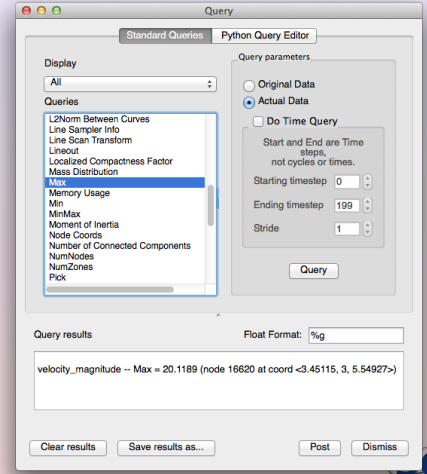
VisIt: Vector Fields Viz – streamlines

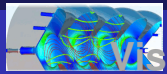
Query to find the Maximum Velocity on the Slice

- [Plot List]: click on the pseudocolor velocity_magnitude to make is **active**
- Go to the **Controls Menu**: [Max] select Actual Data ⇝**Query**

➠ This will return the *maximum scalar value on the slice* and the x, y, z coordinates of the node associated with this value.

➠ We will use the x,y,z coordinates of this node to seed a set of streamlines.

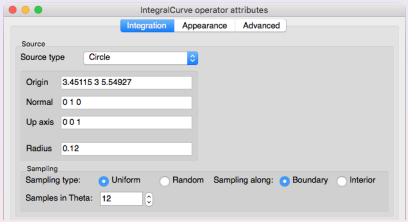


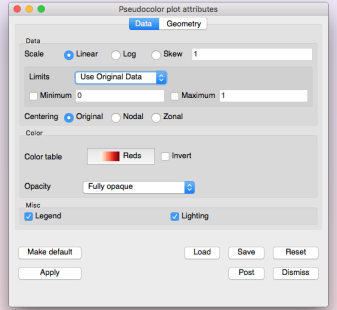


VisIt: Vector Fields Viz – streamlines

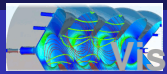
Plotting Streamlines of Velocity

[Plot List]: Add → Pseudocolor → operators/IntergralCurve/velocity





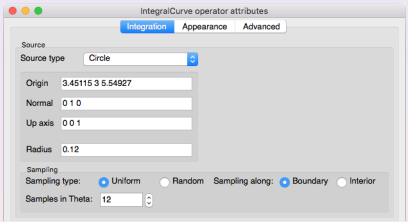
- [Geometry tab]
  - Line type → Tubes, Tail → Sphere,
  - Head → Cone,
  - Head/Tail Radius → 0.02
- Apply / Draw → [Time Slider] click Play / Stop

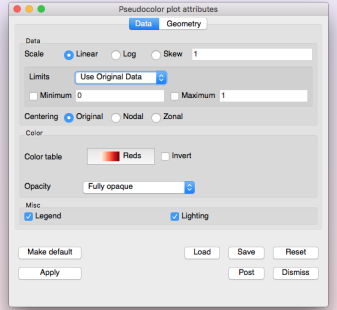


VisIt: Vector Fields Viz – streamlines

Plotting Streamlines of Velocity

[Plot List]: Add → **Pseudocolor** → operators/IntergralCurve/velocity





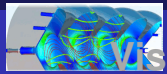
- [Geometry tab]

Line type ⇝ **Tubes**, Tail ⇝ **Sphere**,

Head ⇝ **Cone**,

Head/Tail Radius ⇝ **0.02**

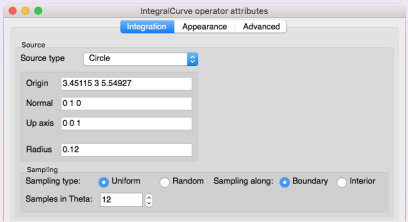
- Apply / Draw ➡ [Time Slider] click Play / Stop

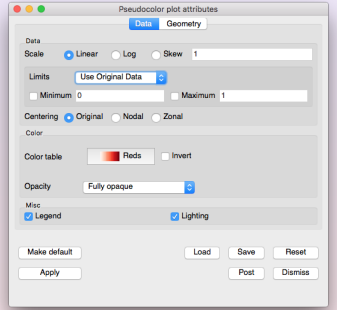


VisIt: Vector Fields Viz – streamlines

Plotting Streamlines of Velocity

[Plot List]: Add → **Pseudocolor** → operators/IntergralCurve/velocity





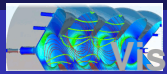
- [Geometry tab]

Line type ⇝ **Tubes**, Tail ⇝ **Sphere**,

Head ⇝ **Cone**,

Head/Tail Radius ⇝ **0.02**

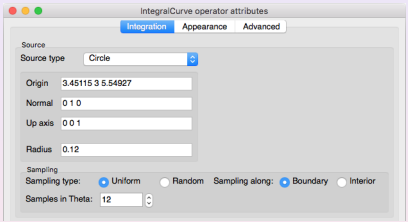
- Apply / Draw → [Time Slider] click Play / Stop

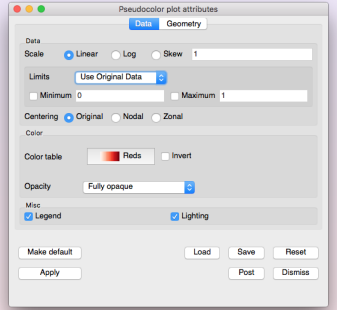


VisIt: Vector Fields Viz – streamlines

Plotting Streamlines of Velocity

[Plot List]: Add → **Pseudocolor** → operators/IntergralCurve/velocity





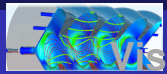
- [Geometry tab]

Line type ⇝ **Tubes**, Tail ⇝ **Sphere**,

Head ⇝ **Cone**,

Head/Tail Radius ⇝ **0.02**

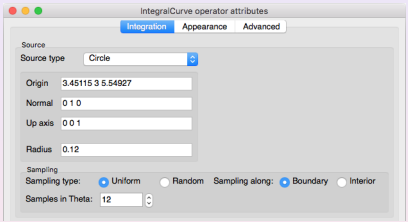
- Apply / Draw → [Time Slider] click Play / Stop

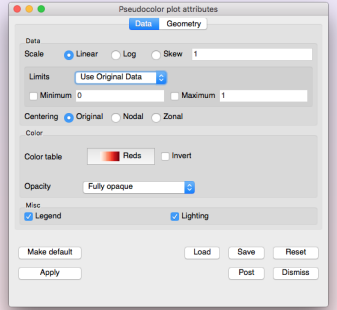


VisIt: Vector Fields Viz – streamlines

Plotting Streamlines of Velocity

[Plot List]: Add → **Pseudocolor** → operators/IntergralCurve/velocity





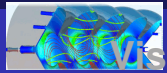
- [Geometry tab]

  Line type ⇝ **Tubes**, Tail ⇝ **Sphere**,

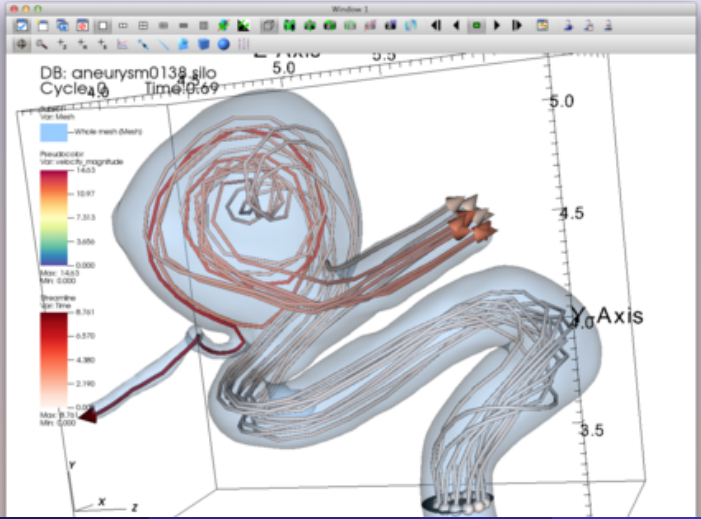
  Head ⇝ **Cone**,

  Head/Tail Radius ⇝ **0.02**

- Apply / Draw ➟ [Time Slider] click Play / Stop



VisIt: Vector Fields Viz – streamlines



Outline

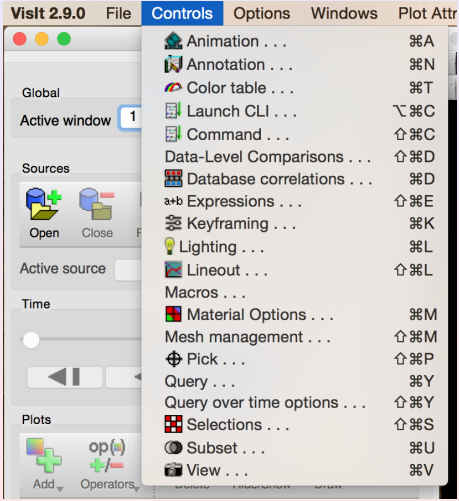
5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

Outline

5 Additional Material
- VTK Format
- Blood Aneurism – tutorial
- Dataset Exploration
- Vector Fields Visualizations
- High Quality Plots
- Professional Quality Plots

Professional Quality Plots

- Annotations
- Colors
- Lighting
- Views



Annotations

Annotations

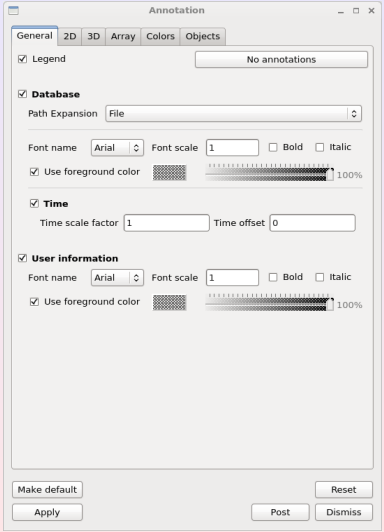
- objects in the viz-window that convey information about the plots
- make clear what is being visualized and make the visualization appear more polished

Types of Annotations

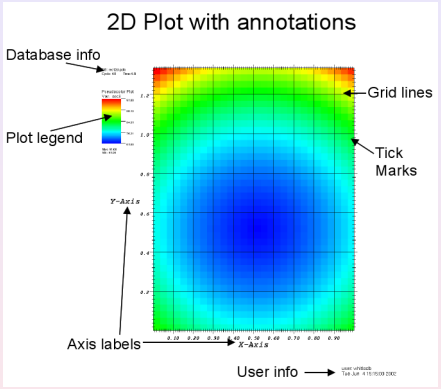
- Database name
- User name
- plot legends
- plot axes and labels (2d & 3d)
- 3d triad
- 2d, 3d text
- time slider
- images
- line and arrows

Annotation Window

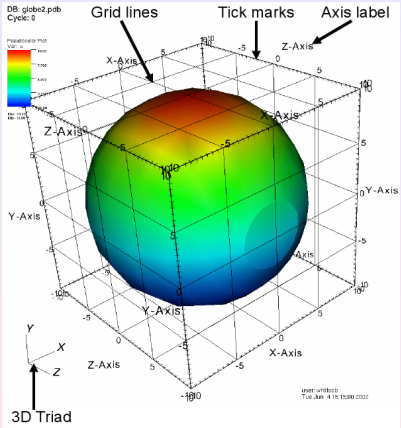
- General annotations
- 2D axes settings
- 3D axes settings
- Array axis settings
- Color and Backgrounds
- Objects (legend, time slider, ...)



2D & 3D Annotations



➡ Controls → **Annotation...**
➡ **["2D" tab]**



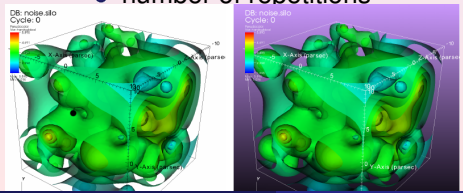
➡ Controls →
Annotation... ➡ **["3D" tab]**

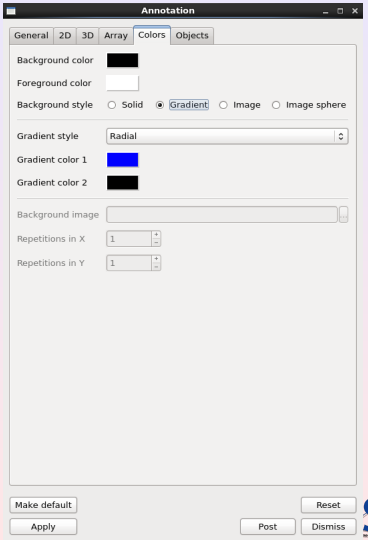
Colors and Backgrounds

Controls → **Annotation...**

➡["**Colors" tab**]

- set background/foreground
- Backgroun styles:
  - solid
  - gradient
  - image (flat image)
  - image sphere (warped image, that rotates with the view)
  - number of repetitions



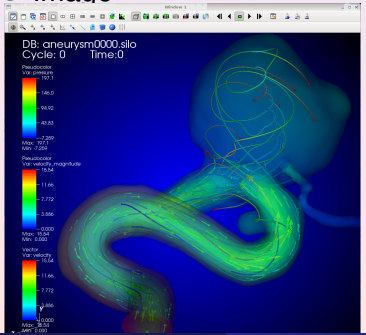


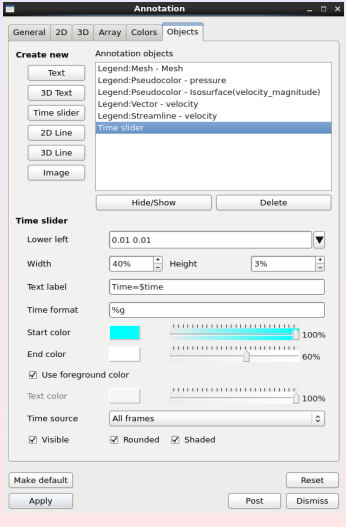
Annotation Objects

Controls → **Annotation...**

➡["Objects" tab]

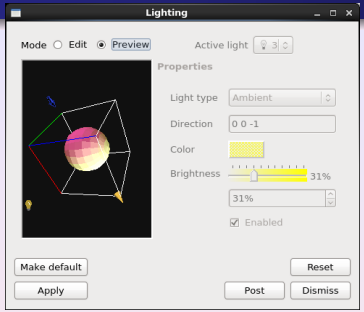
- 2D/3D Text
- 2D/3D Lines
- Time Slider
- Image





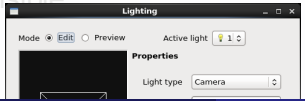
Lighting

- Lighting affects the brightness of plots
- 3D visualizations may require multiple *light sources*
- VisIt allows up to 8 light sources
- Each light source can be positioned and colored



➡ Controls → Lighting...

- [Edit]: configure light sources
- [Preview]: all sources are visible

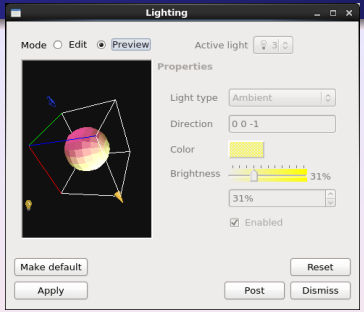


- only the active light can be modified
- once a light has been selected, you can change its props.
- Types of lights: Ambient, Camera, Object Light

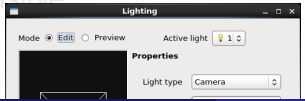
Lighting

- Lighting affects the brightness of plots
- 3D visualizations may require multiple *light sources*
- VisIt allows up to 8 light sources
- Each light source can be positioned and colored

➠ Controls → **Lighting...**



- [Edit]: configure light sources
- [Preview]: all sources are visible

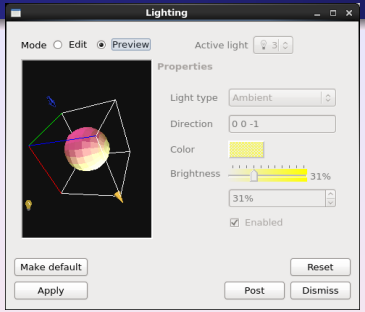


- only the active light can be modified
- once a light has been selected, you can change its props.
- Types of lights: Ambient, Camera, Object Light

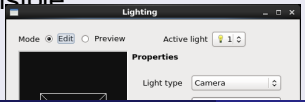
Lighting

- Lighting affects the brightness of plots
- 3D visualizations may require multiple *light sources*
- VisIt allows up to 8 light sources
- Each light source can be positioned and colored

➠ Controls → **Lighting...**



- [Edit]: configure light sources
- [Preview]: all sources are visible

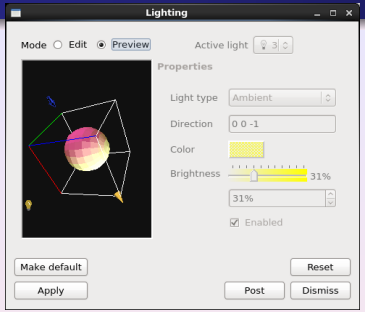


- only the active light can be modified
- once a light has been selected, you can change its props.
- Types of lights: Ambient, Camera, Object Light

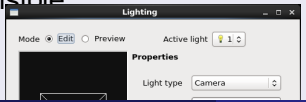
Lighting

- Lighting affects the brightness of plots
- 3D visualizations may require multiple *light sources*
- VisIt allows up to 8 light sources
- Each light source can be positioned and colored

➠ Controls → **Lighting...**



- [Edit]: configure light sources
- [Preview]: all sources are visible

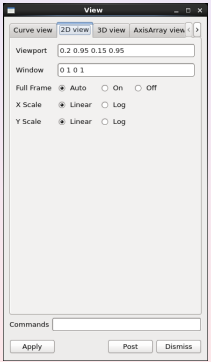


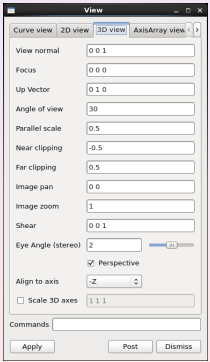
- only the active light can be modified
- once a light has been selected, you can change its props.
- Types of lights: Ambient, Camera, Object Light

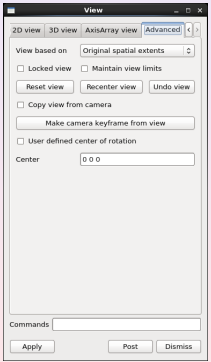
View

- the “view” can be set *interactively* in the viz-window (click and drag, ...)
- or using a “View Window”, to specify exactly the configuration view

➡ Controls → View...





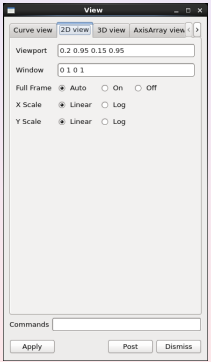


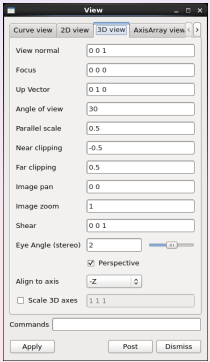
- “**Locked view**”: when the view changes in any locked window, all other locked windows readjust to it

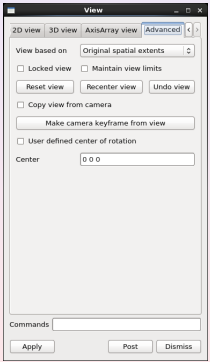
View

- the “view” can be set *interactively* in the viz-window (click and drag, ...)
- or using a “View Window”, to specify exactly the configuration view

➠ Controls → **View...**







- “**Locked view**”: when the view changes in any locked window, all other locked window readjust to it

Generating “hardcopies”, aka plotting images and figures

Generating Hardcopies: Figures

1. File → **Set Save option...**
   Allows you to control the properties of the image: file type, resolution, naming convention, etc...
2. File → **Save window**
   Generates the image/file of the currently displayed window.