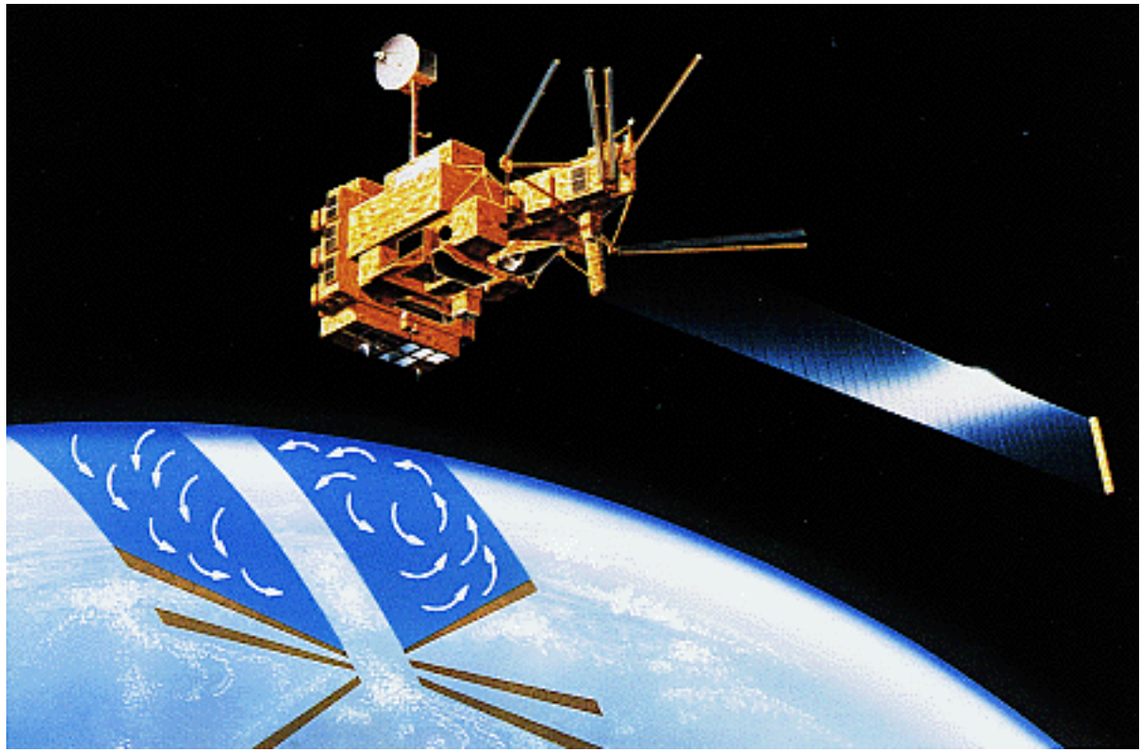
ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

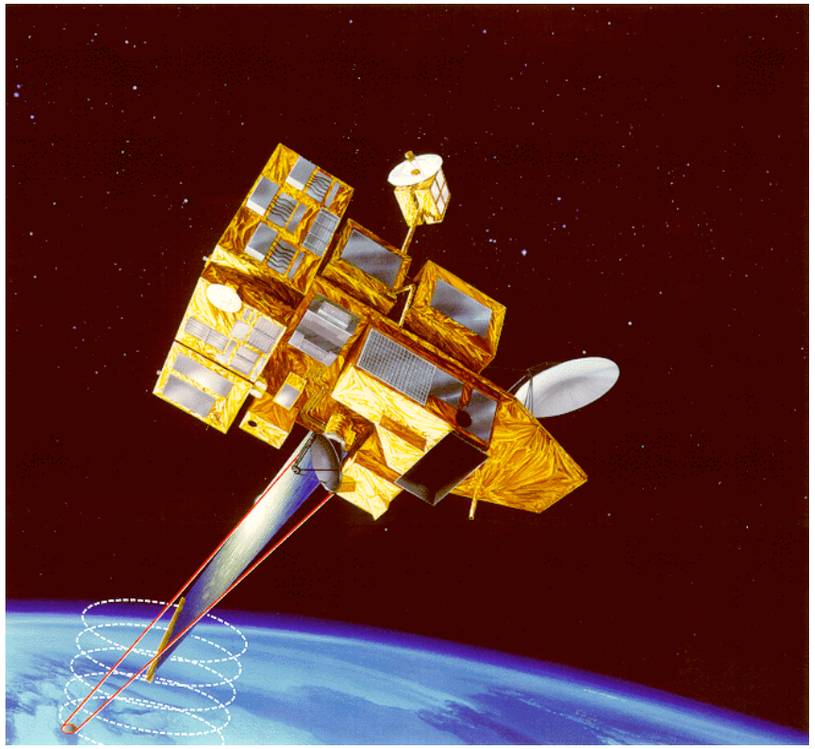
Scientific Applications using Python

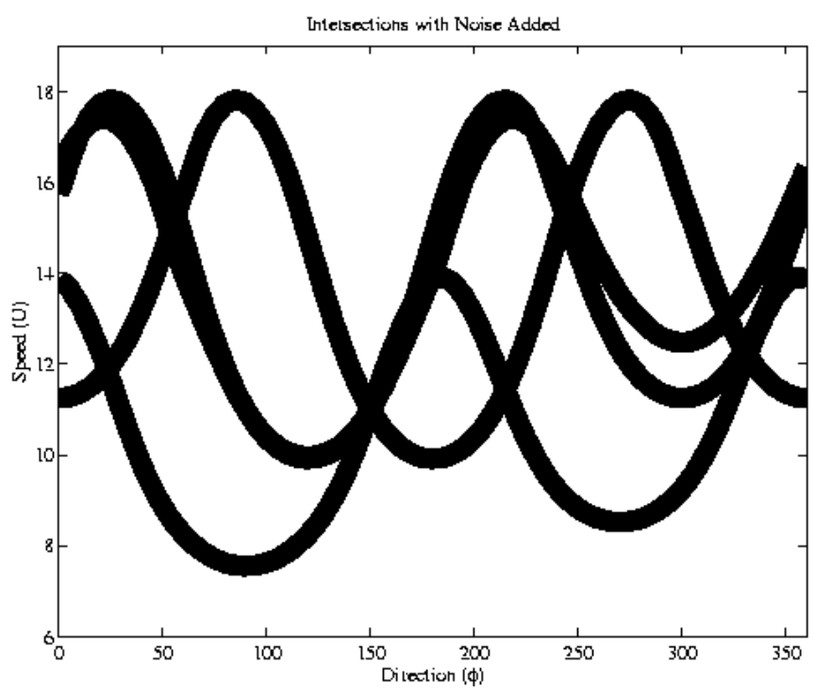
Travis E. Oliphant

Oct 12, 2010

Graduate degree in Electrical Engineering

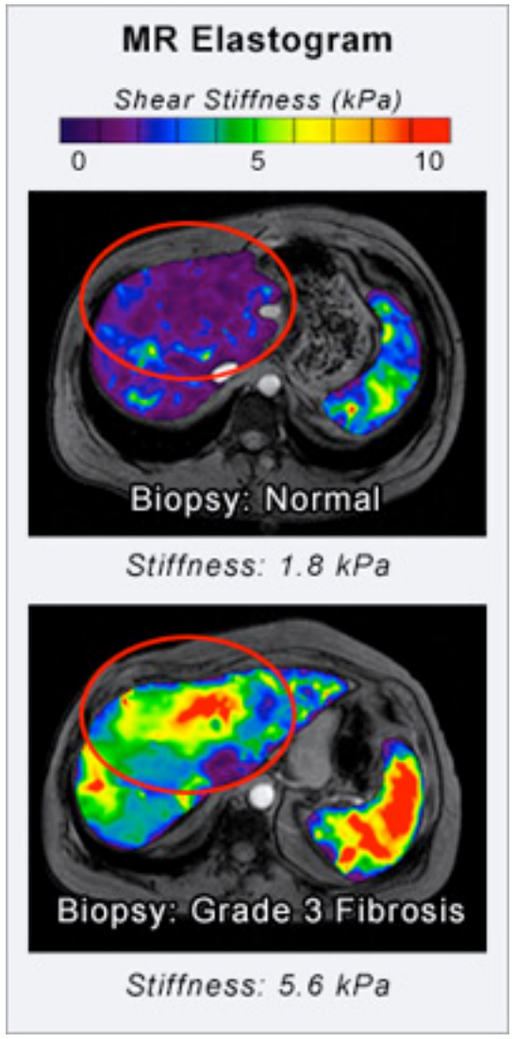






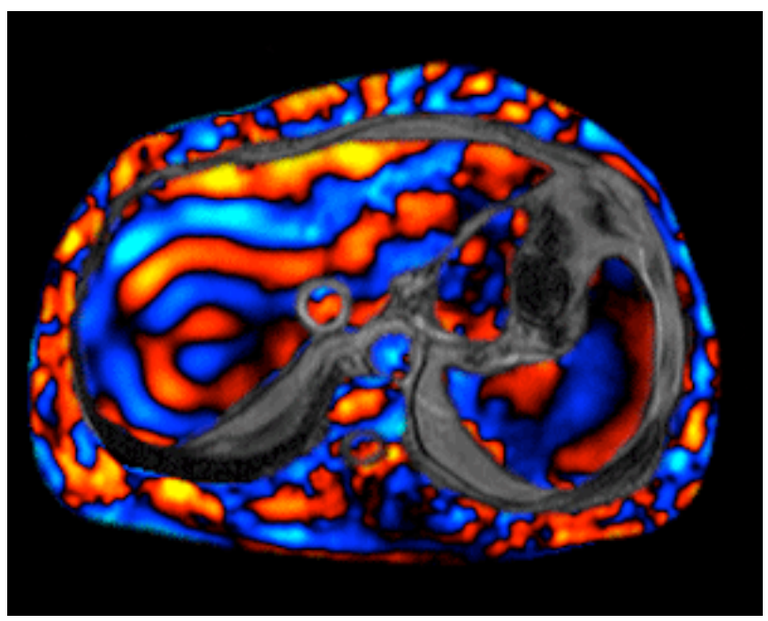
Began an appreciation for Inverse Problems!

Graduate Degree in Biomedical Engineering

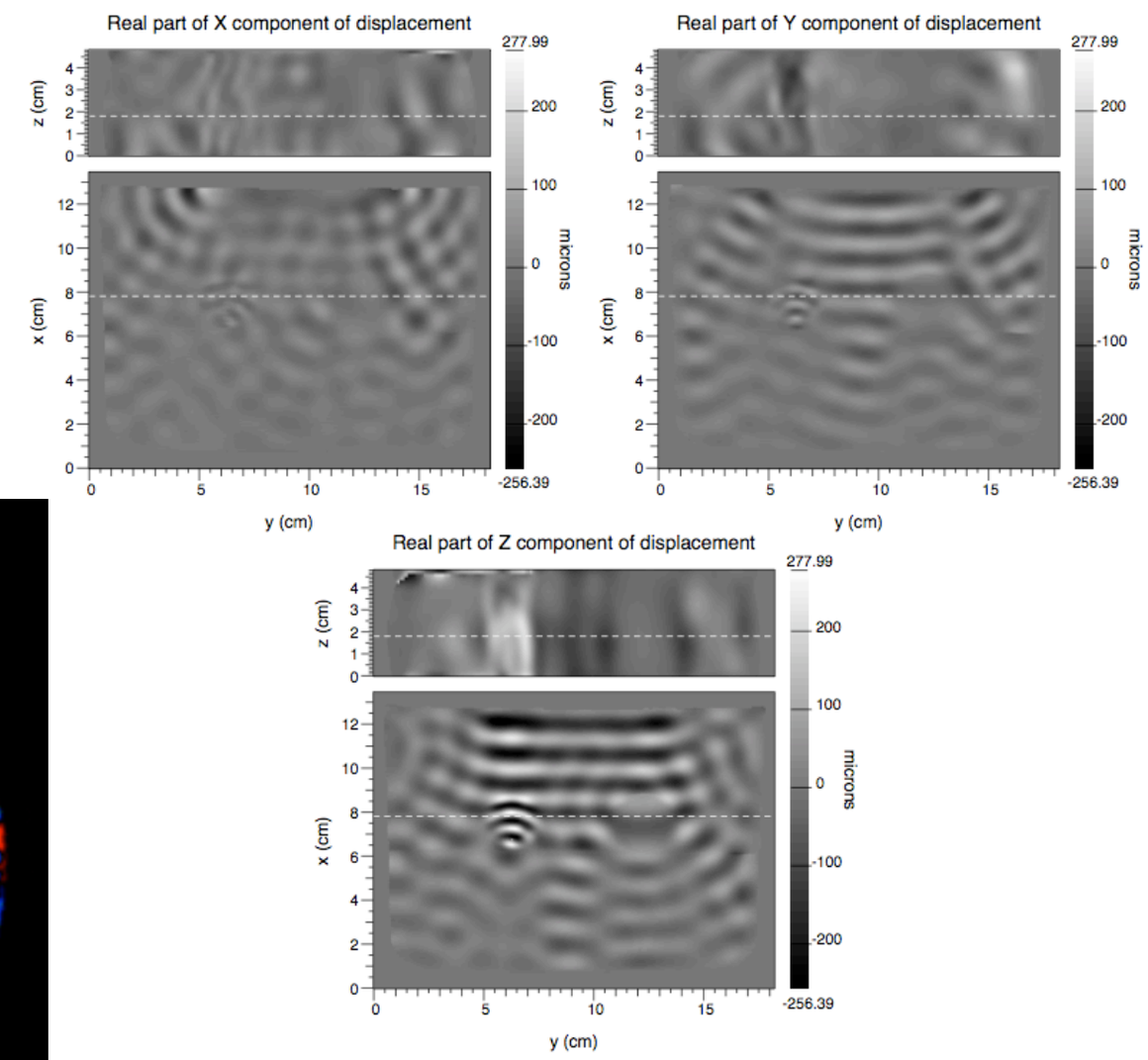


MAYO CLINIC

$U_X(\mathbf{a}, f)$



$U_Y(\mathbf{a}, f)$

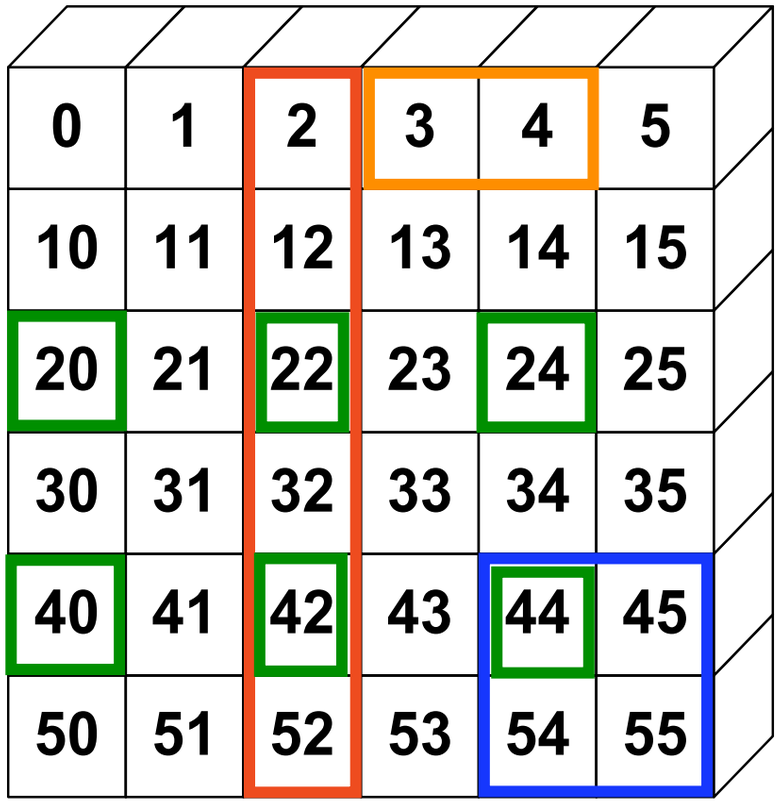


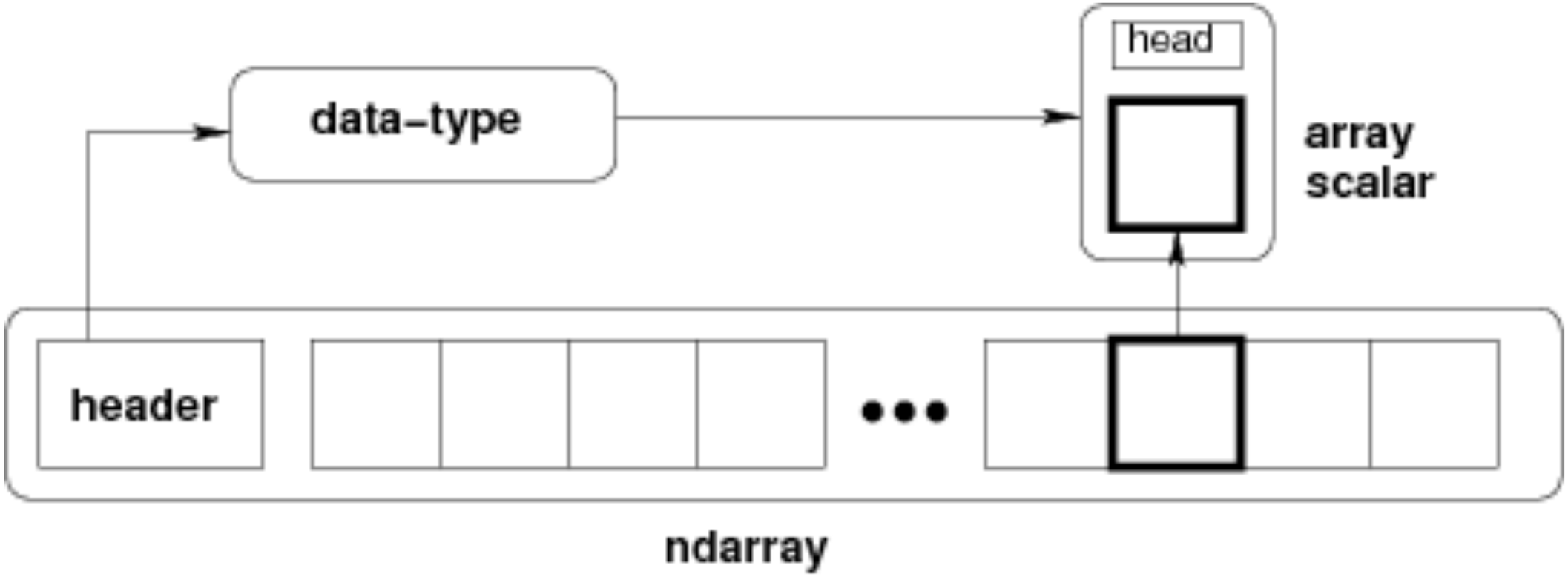
$U_Z(\mathbf{a}, f)$

$$\rho_0 (2\pi f)^2 U_i(\mathbf{a}, f) = [C_{ijkl}(\mathbf{a}, f) U_{k,l}(\mathbf{a}, f)]_{,j}$$

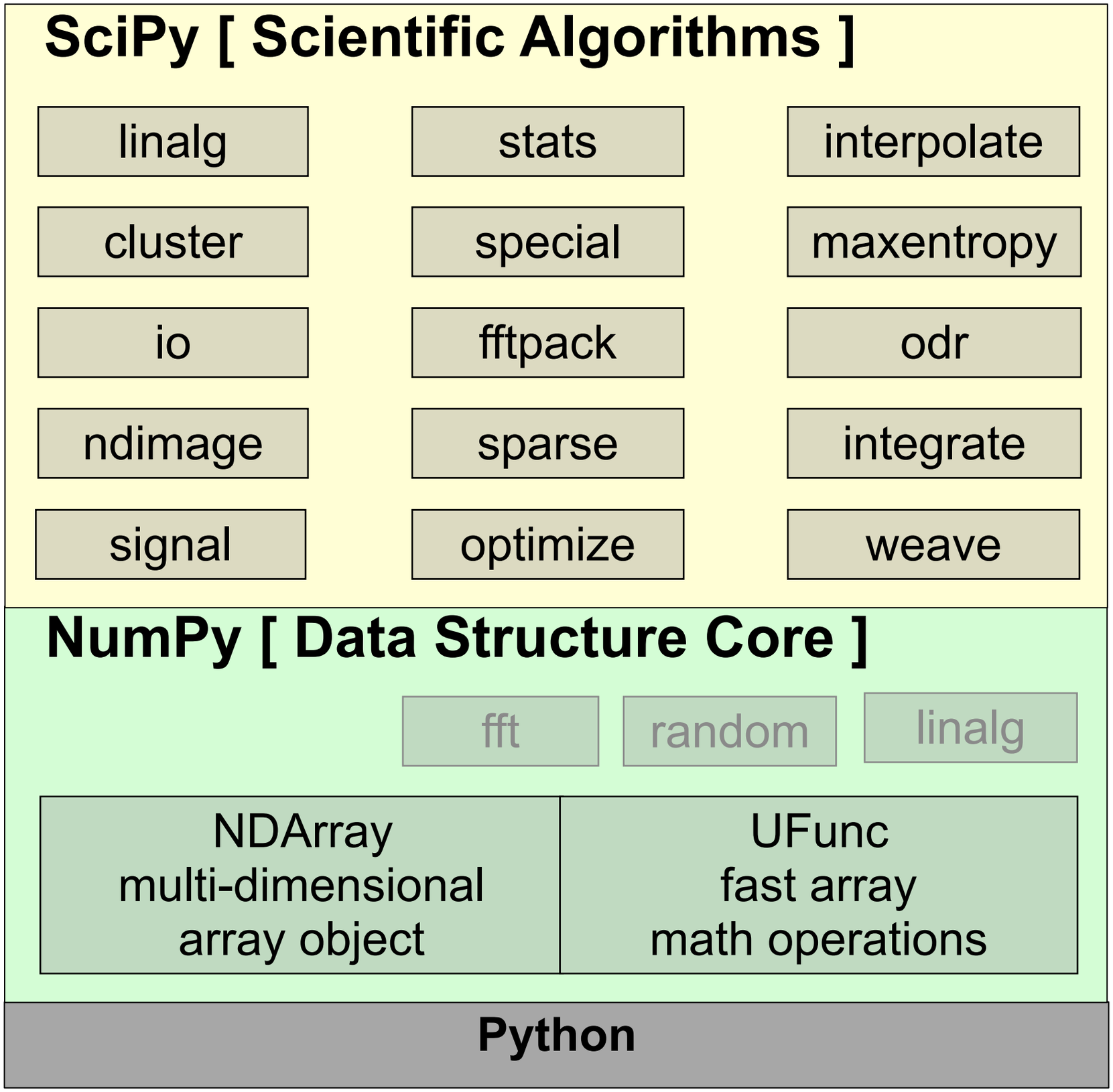
NumPy Array

A NumPy array is an N-dimensional homogeneous collection of “items” of the same “kind”. The kind can be any arbitrary structure and is specified using the data-type.



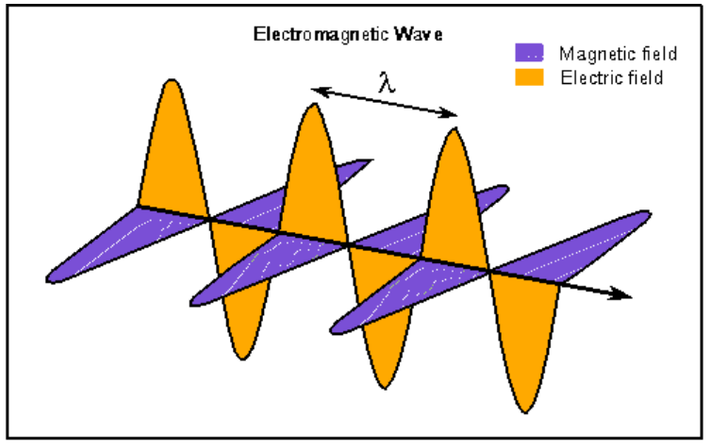


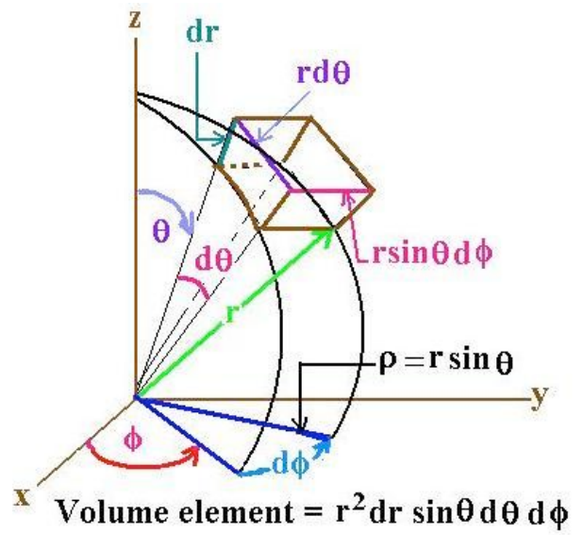
SciPy


SciPy [Scientific Algorithms]
linalg
stats
interpolate
cluster
special
maxentropy
io
fftpack
odr
ndimage
sparse
integrate
signal
optimize
weave
NumPy [Data Structure Core]
fft
random
linalg
NDArray
multi-dimensional
array object
UFunc
fast array
math operations
Python

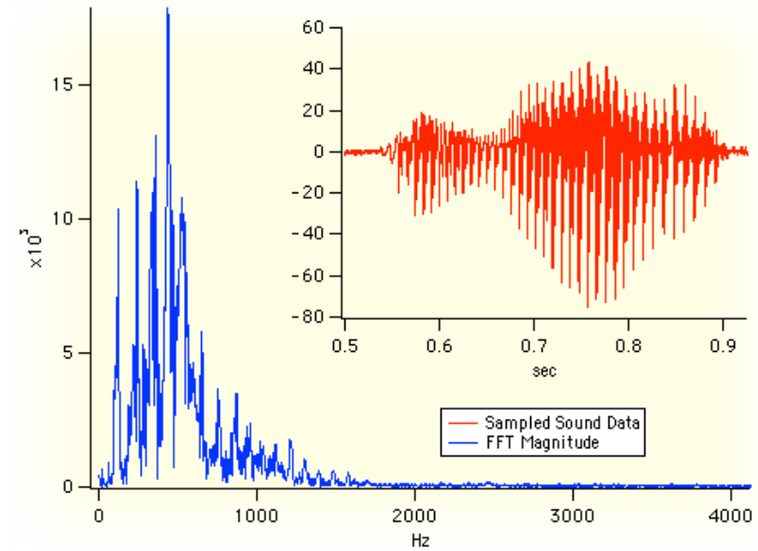
Assistant Professor of EE

Electromagnetics

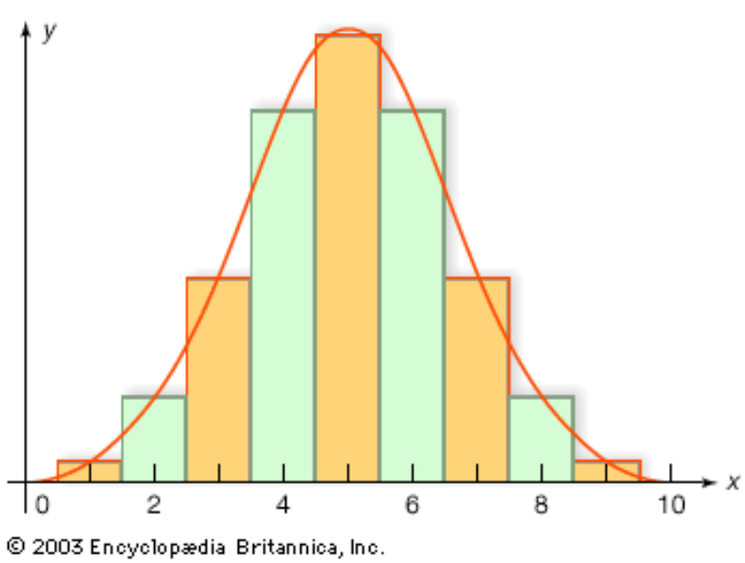




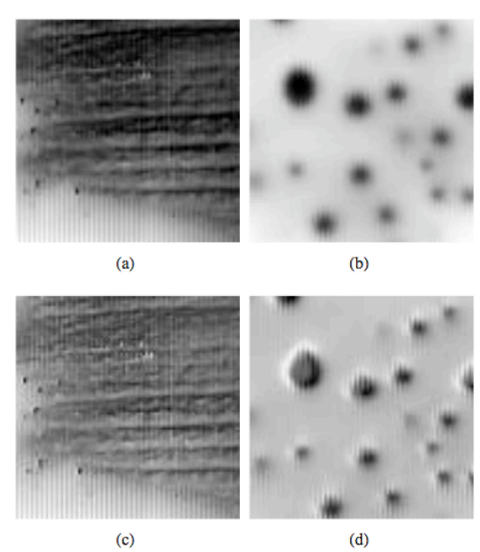
Signal Processing

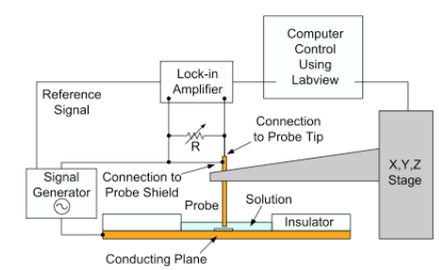


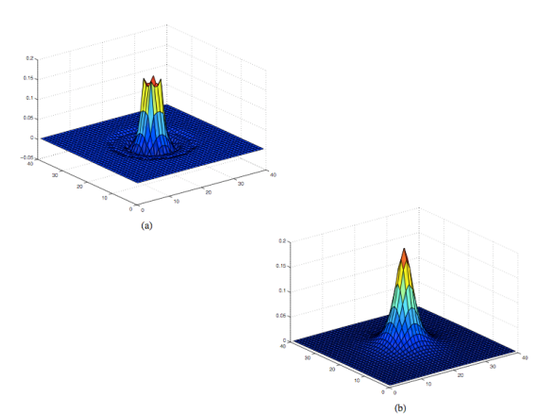
Probability Theory



Inverse Problems







SciPy

Unique Visitors

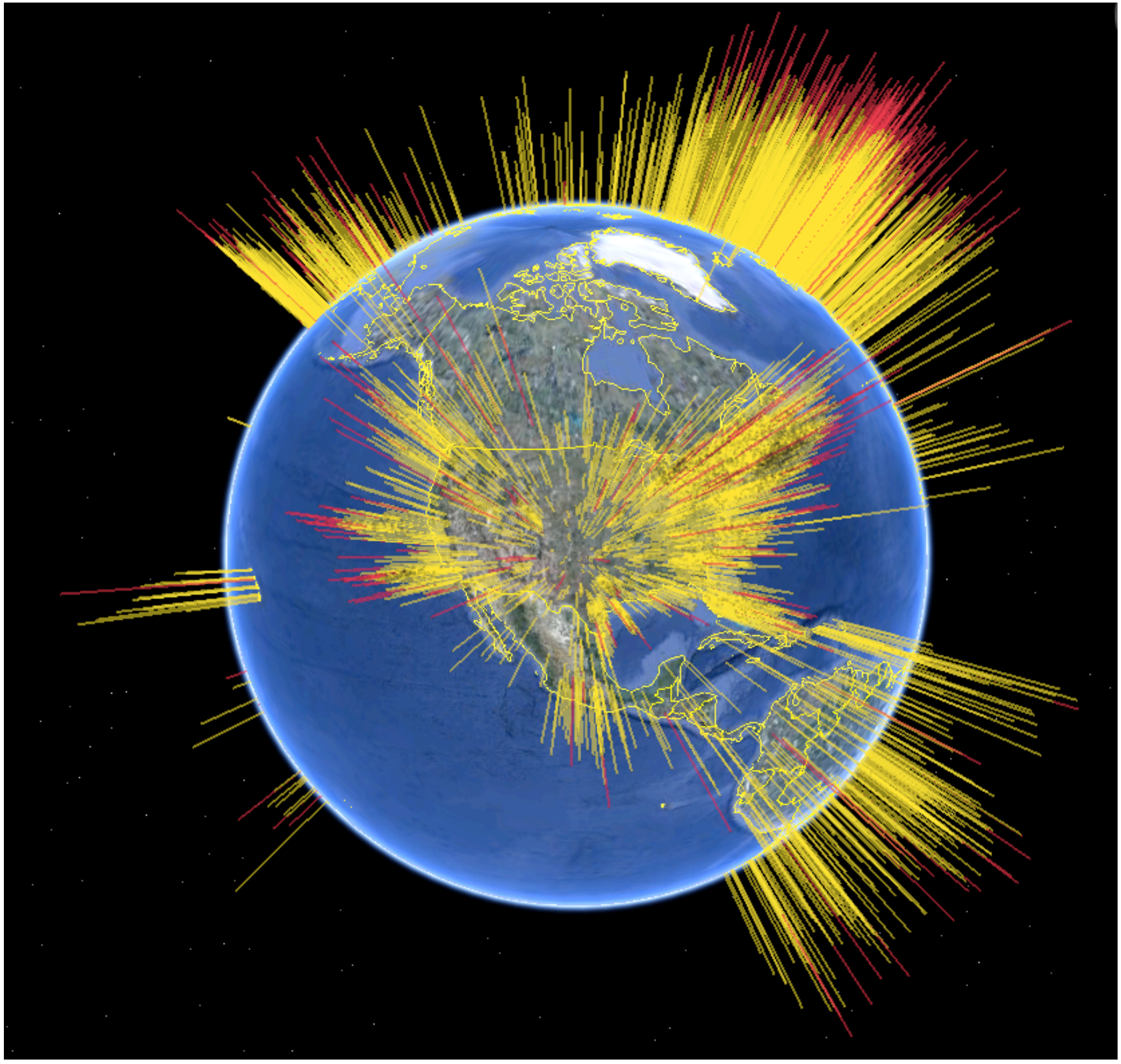
<u>2009</u>

1,073,799

<u>2010</u> (first half)

687,814

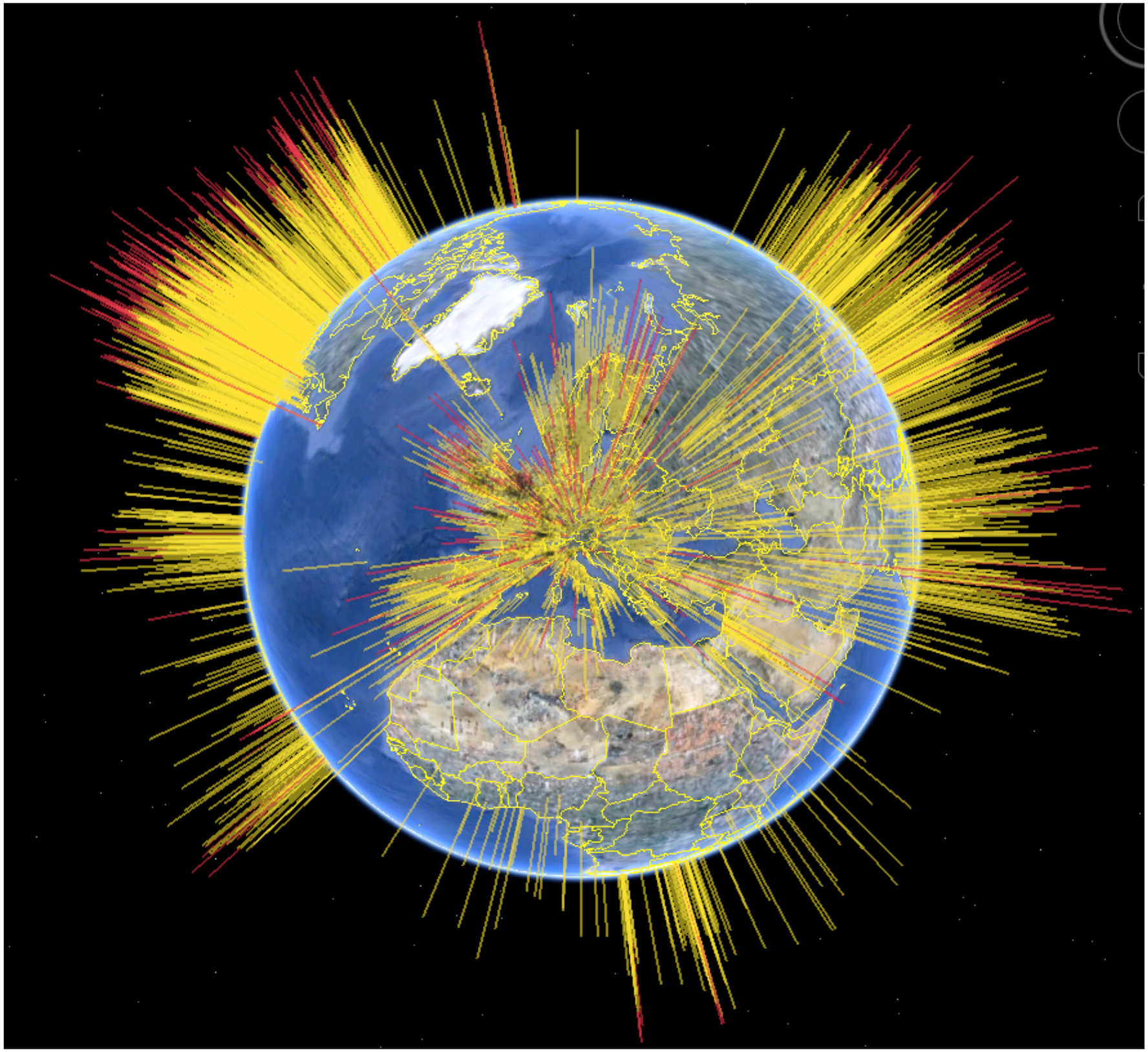
In 2009



North America

In 2009

Europe



SciPy Unique Visits

Universities (.edu)		
Location	2009	2010
MIT	**2631**	**1448**
Stanford	**1374**	**1074**
Berkeley	**1504**	**813**
Harvard	**1493**	**768**
CalTech	**1345**	**711**
U Wash	**954**	**701**
Cornell	**849**	**683**
Cambridge	**968**	**613**
U Mich	**708**	**560**
Oxford	**708**	**443**
...	**...**	**...**
Total	**49,857**	**33,068**

Labs (.gov)		
Location	2009	2010
NASA	**1937**	**1060**
NOAA	**412**	**317**
Sandia	**236**	**236**
LLNL	**246**	**232**
NIH	**338**	**193**
LBL	**295**	**190**
NIST	**288**	**162**
FNL	**179**	**147**
USGS	**190**	**116**
BNL	**179**	**109**
...	**...**	**...**
Total	**7,223**	**4,157**

Enthought's Business

Products & Support	Training	Consulting
Enthought Python Distribution	Scientific Computing with Python	Rigorous process analysis
Open source tools	Software Engineering Best Practices	Custom library & application development
Domain-specific applications & libraries	Custom classes	Visualization design

Enthought's Customers

Oil & Exploration

Shell

ConocoPhillips

Industrial & Consumer

AIRBUS

DOOSAN

Doosan Babcock Energy

P&G

NORTHROP GRUMMAN

Semiconductor

freescale™ semiconductor

XILINX®

Financial

RGM ADVISORS, LLC

EWT LLC

J.P.Morgan

CHASE

Internet

Microsoft®

rackspace

Government & Labs

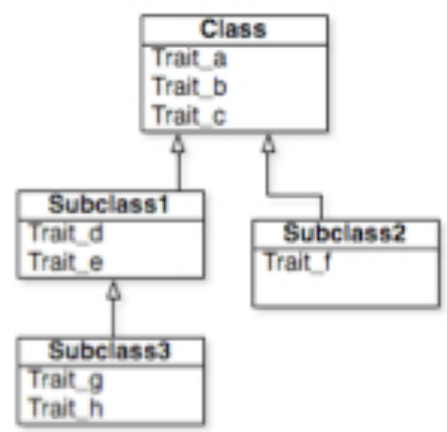
Los Alamos NATIONAL LABORATORY EST. 1943

JPL

ARL

Hardware

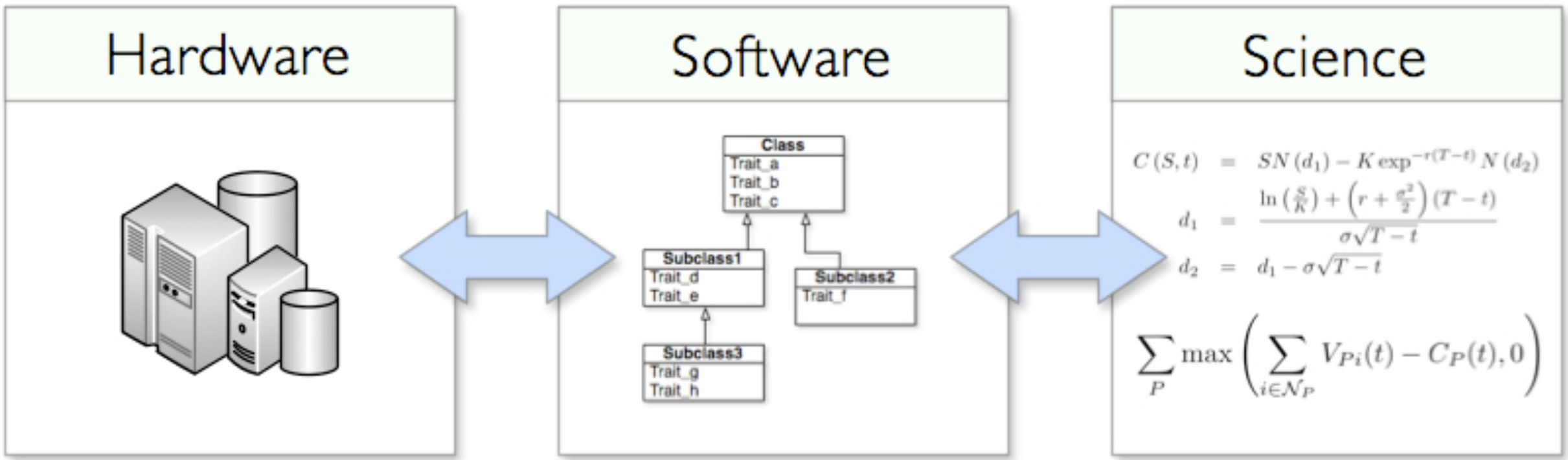
Software

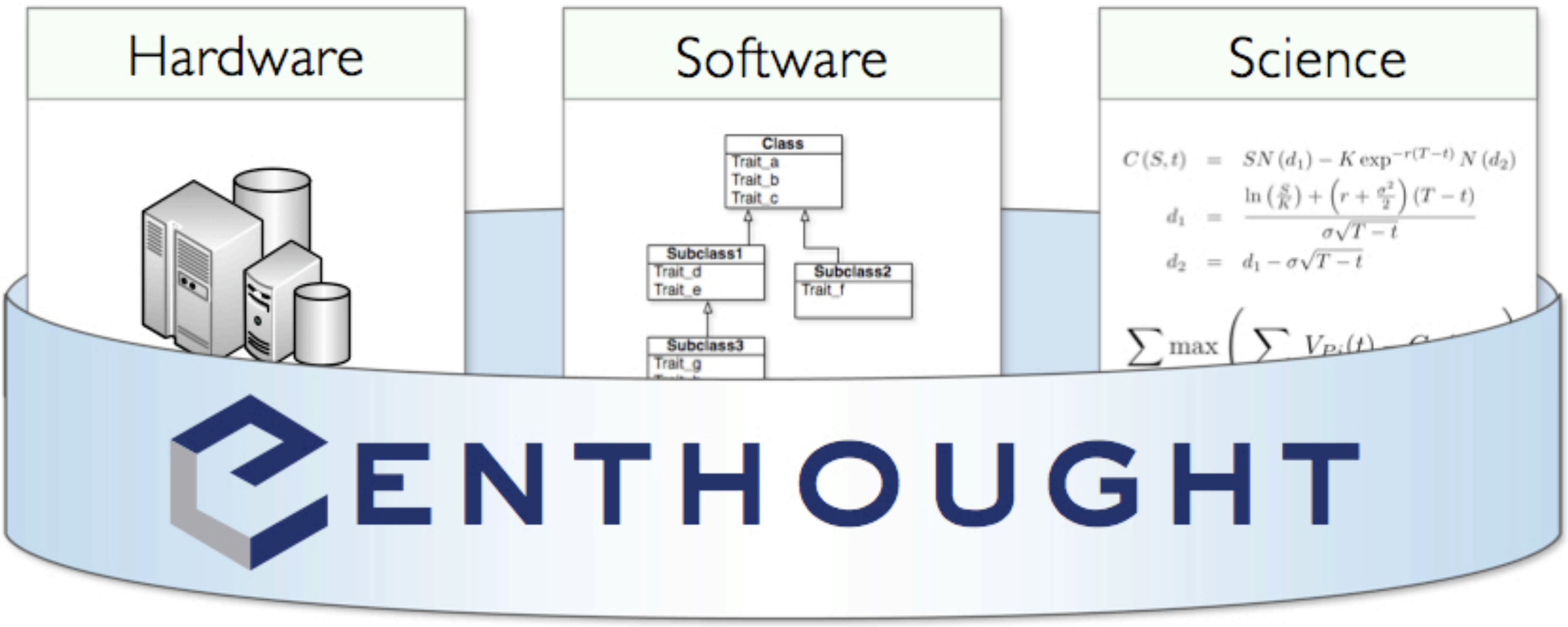


Science

$$
\begin{aligned}
C(S,t) &= SN(d_1) - K\exp^{-r(T-t)}N(d_2) \\
d_1 &= \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\
d_2 &= d_1 - \sigma\sqrt{T-t}
\end{aligned}
$$

$$
\sum_P \max\left(\sum_{i \in \mathcal{N}_P} V_{Pi}(t) - C_P(t), 0\right)
$$


Hardware
Software
Science
Class
Trait_a
Trait_b
Trait_c
Subclass1
Trait_d
Trait_e
Subclass2
Trait_f
Subclass3
Trait_g
Trait_h
$C(S,t) = SN(d_1) - K\exp^{-r(T-t)}N(d_2)$
$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$
$d_2 = d_1 - \sigma\sqrt{T-t}$
$\sum_P \max\left(\sum_{i \in \mathcal{N}_P} V_{Pi}(t) - C_P(t), 0\right)$


Hardware
Software
Science
Class
Trait_a
Trait_b
Trait_c
Subclass1
Trait_d
Trait_e
Subclass2
Trait_f
Subclass3
Trait_g
$C(S,t) = SN(d_1) - K\exp^{-r(T-t)}N(d_2)$
$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$
$d_2 = d_1 - \sigma\sqrt{T-t}$
ENTHOUGHT

Hardware

Software

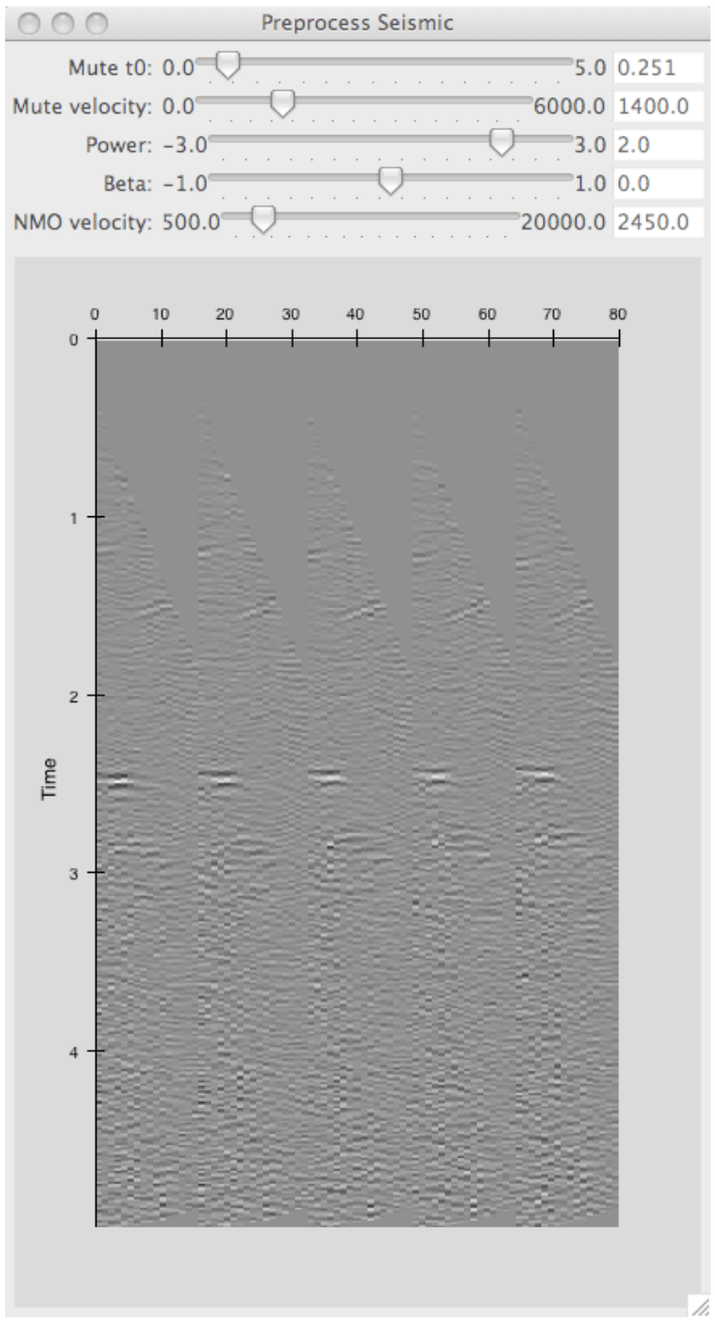
Science

$$
\begin{aligned}
C(S,t) &= SN(d_1) - K\exp^{-r(T-t)}N(d_2) \\
d_1 &= \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\
d_2 &= d_1 - \sigma\sqrt{T-t}
\end{aligned}
$$

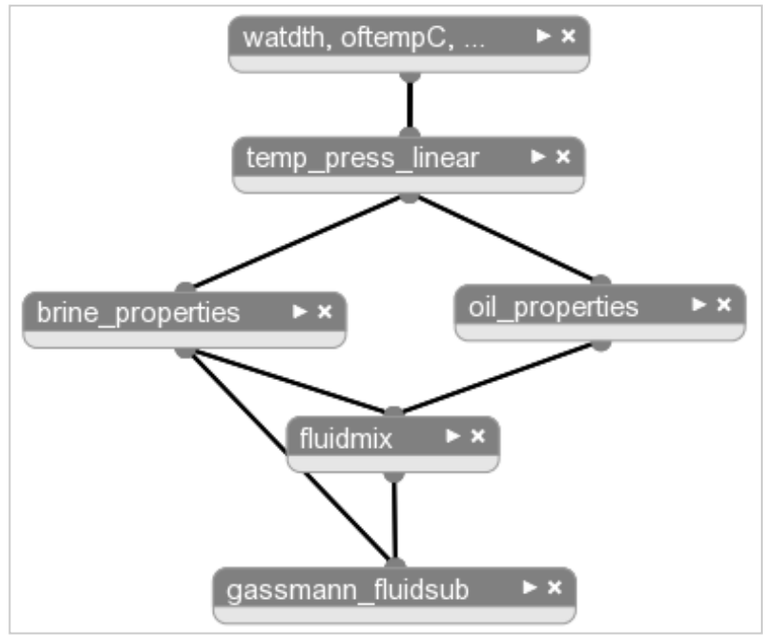
$$
\sum_P \max\left(\sum_{i\in\mathcal{N}_P} V_{Pi}(t) - C_P(t), 0\right)
$$

Three Classes of Users

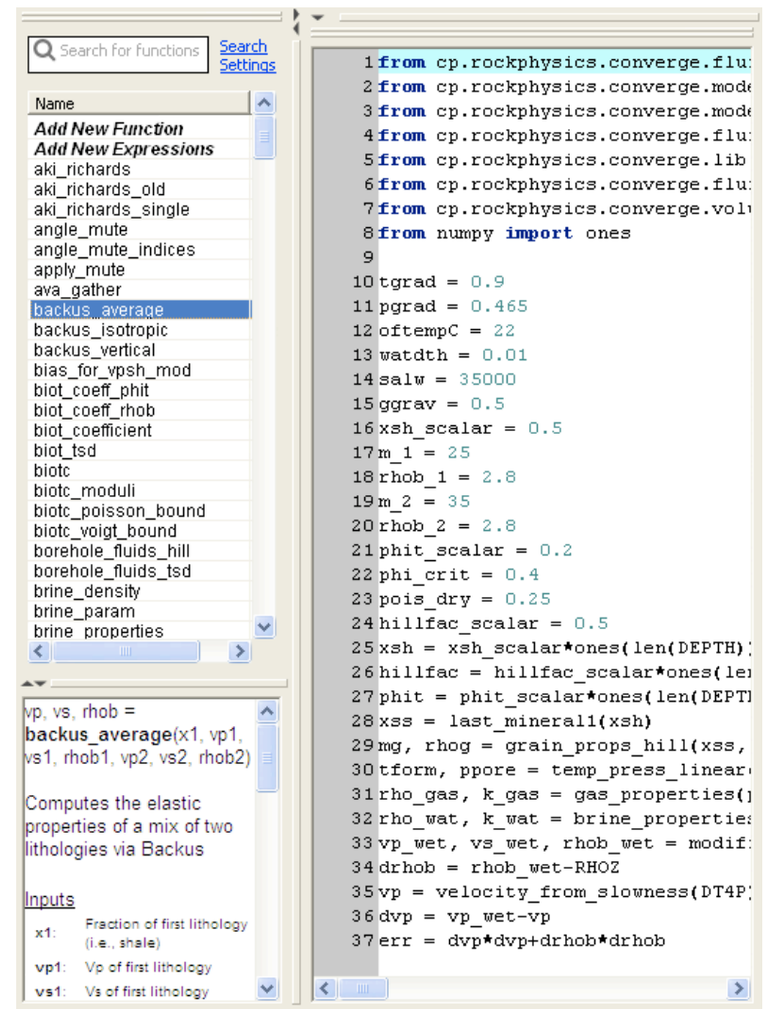
NOVICE



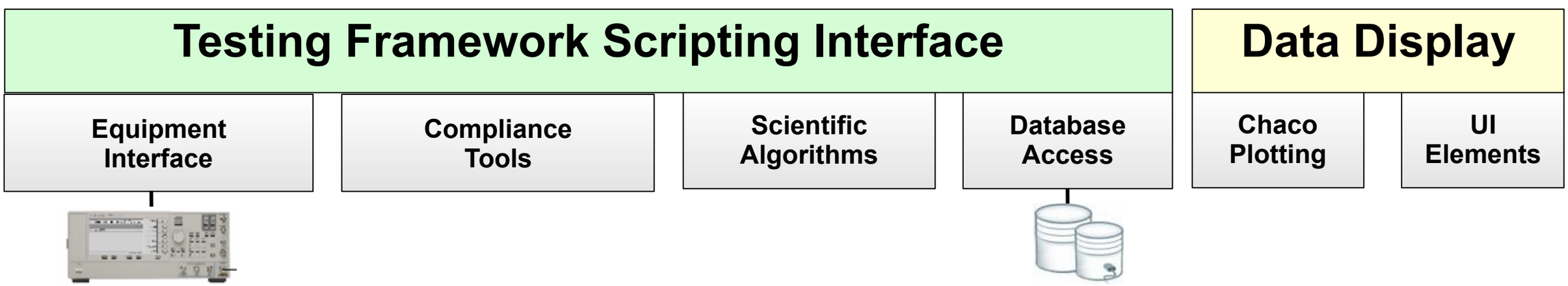
EXPERIENCED



EXPERT



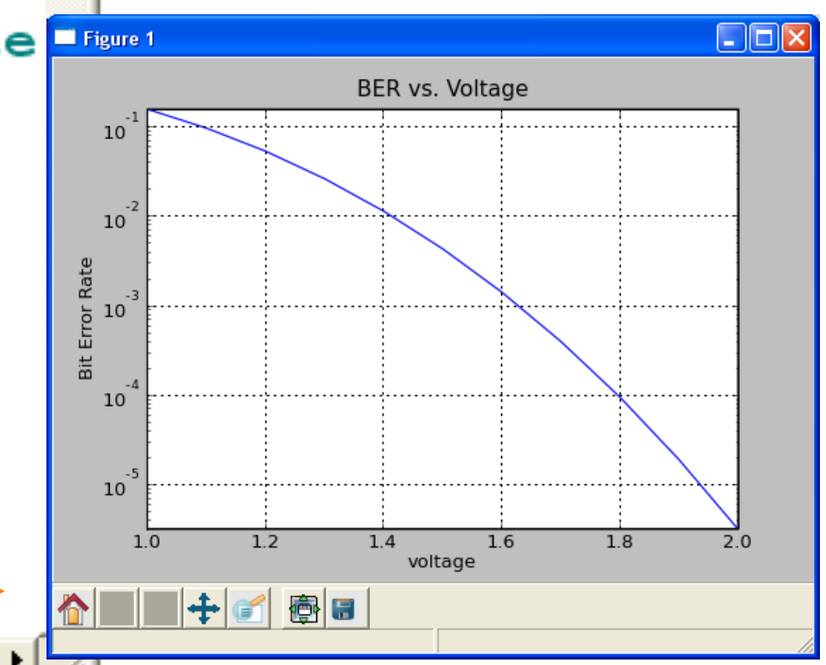
Scripting Layer in Applications

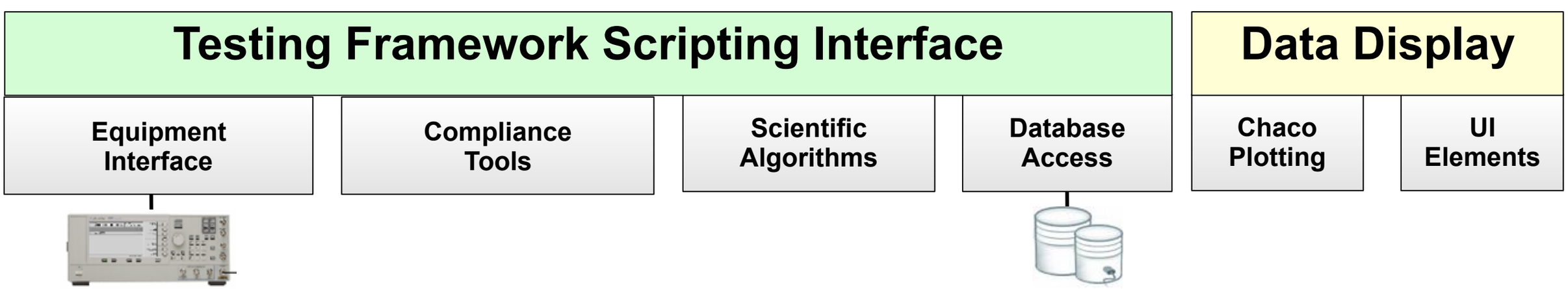


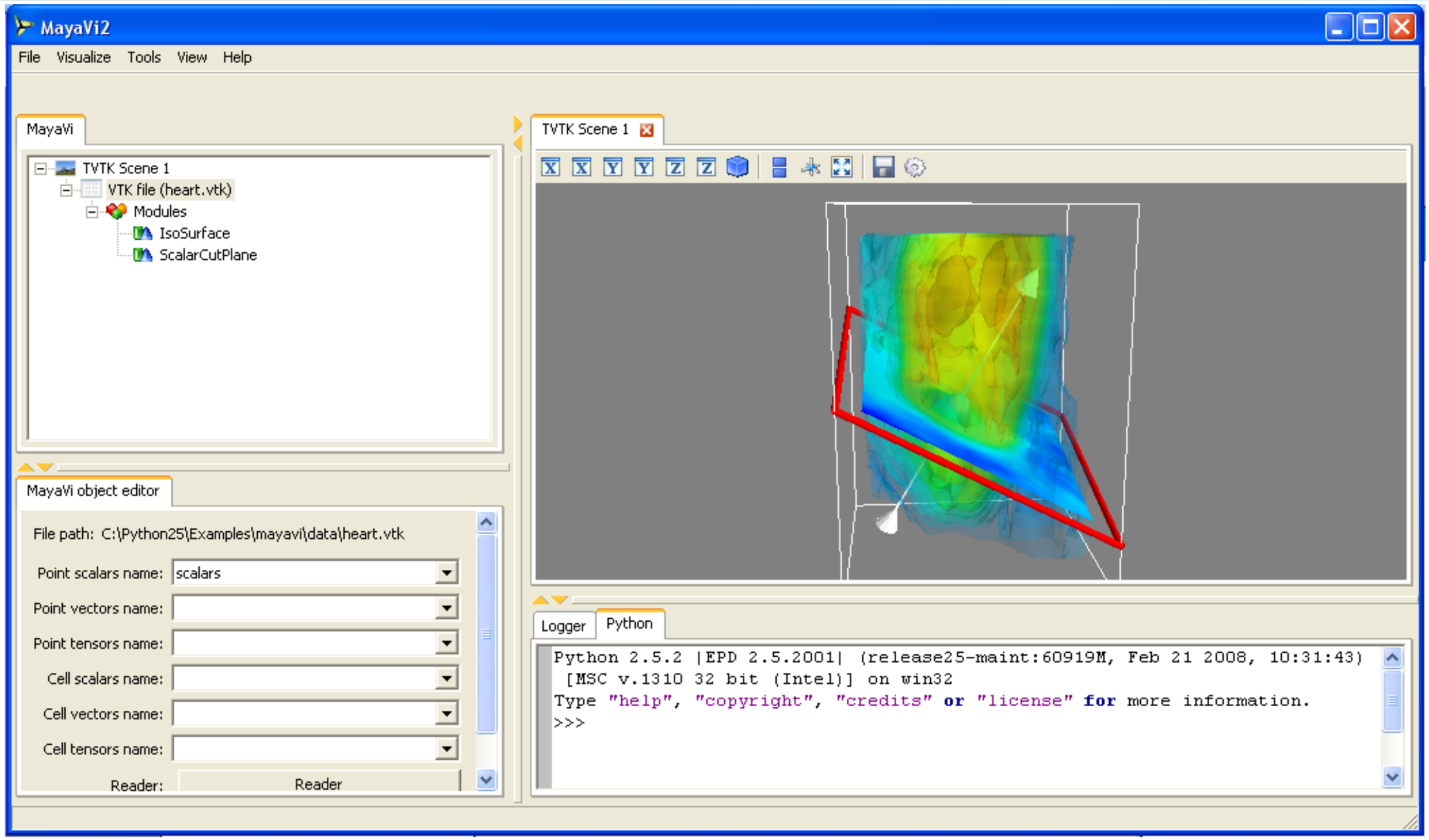
Scripting Layer in Applications

ipython

```
>>> # Create data objects
>>> voltage_source = Agilent66000A()
>>> bert_scope = BERTScope()

>>> # Sweep voltage and look at Bit Error Rate
>>> voltages = linspace(1, 2, 11)
>>> error_rates = []
>>> for voltage in voltages:
...     error_rate = bert_scope.error_rate()
...     error_rates.append(error_rate)

>>> error_rates = array(error_rates)
>>> # Plot the results
>>> plot(voltages, error_rates)
```







Rich Client App (Geophysics, Finance, Etc)

Scripting Interface

Data Display

Equipment Interface

Compliance Tools

Scientific Algorithms

Database Access

Chaco Plotting

UI Elements

VMA – Virtual Mixing Advisor

VMA - New Project *

File Edit Preferences Tools Help

New Project
process 1
process step 1

process step 1

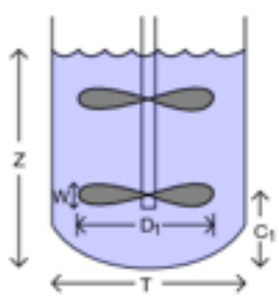
Undo Redo Solve Optimize Plot Debug Profile Edit Docs

Inputs

Baffle angle (deg)	5.00 deg
Baffle configuration	Custom
Number of baffles	4
Baffle offset (mm)	28.0 mm
Baffle width (mm)	85.0 mm
Dish type	flat
Retrofit?	
T (mm)	2000 mm
Variable speed drive?	Yes
Z (mm)	1500 mm
Equipment - Impeller 1	
C_1 (mm)	65.0 mm
D_1 (mm)	900 mm
1st blade angle (deg)	45.0 deg
1st blade count	6.00
Type of first impeller	pitched blade turbine

Outputs

Equipment	
Dish height (m)	0.000 m
1st std. motor percent (percent)	0.813
1st std. motor size (kilowatt)	0.500 kilowatt
2nd std. motor percent (percent)	0.542
2nd std. motor size (kilowatt)	0.750 kilowatt
S (mm)	535 mm
$V_{straight}$ (liters)	137 liters
V (liters)	4710 liters
Z/T	0.750
Equipment - Impeller 1	
C_1/D_1	0.0722
C_1/T	0.0325
D_1/T	0.450
W_1/D_1	0.133
W_1/T	0.0600



Alert Messages

Search alerts

Z/T is below the recommended lower limit.
CFD may be an appropriate tool

Details

Alert

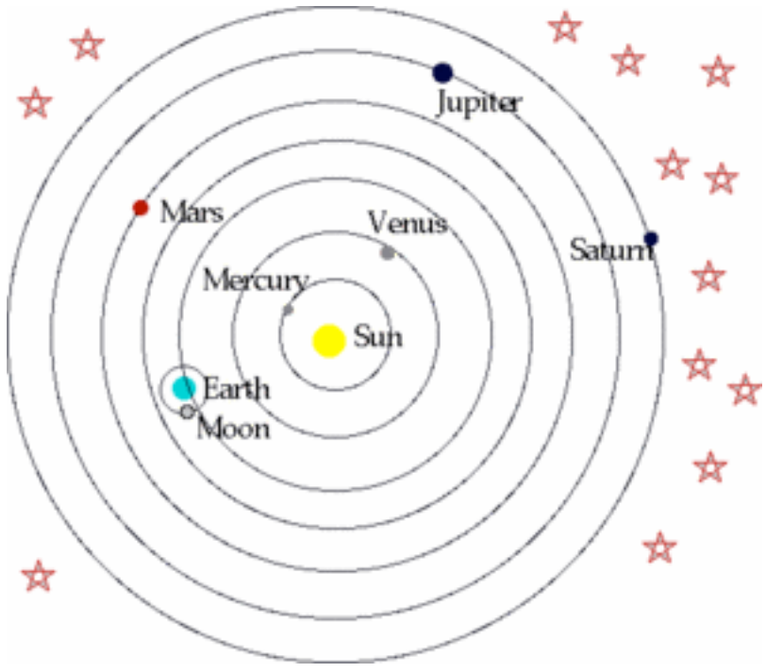
Z/T is below the recommended lower limit.

Short, fat tanks require a larger torque machine, and hence increase capital cost. Adjust Z or T so that Z/T ≥ 0.90.

blndlv.pdf

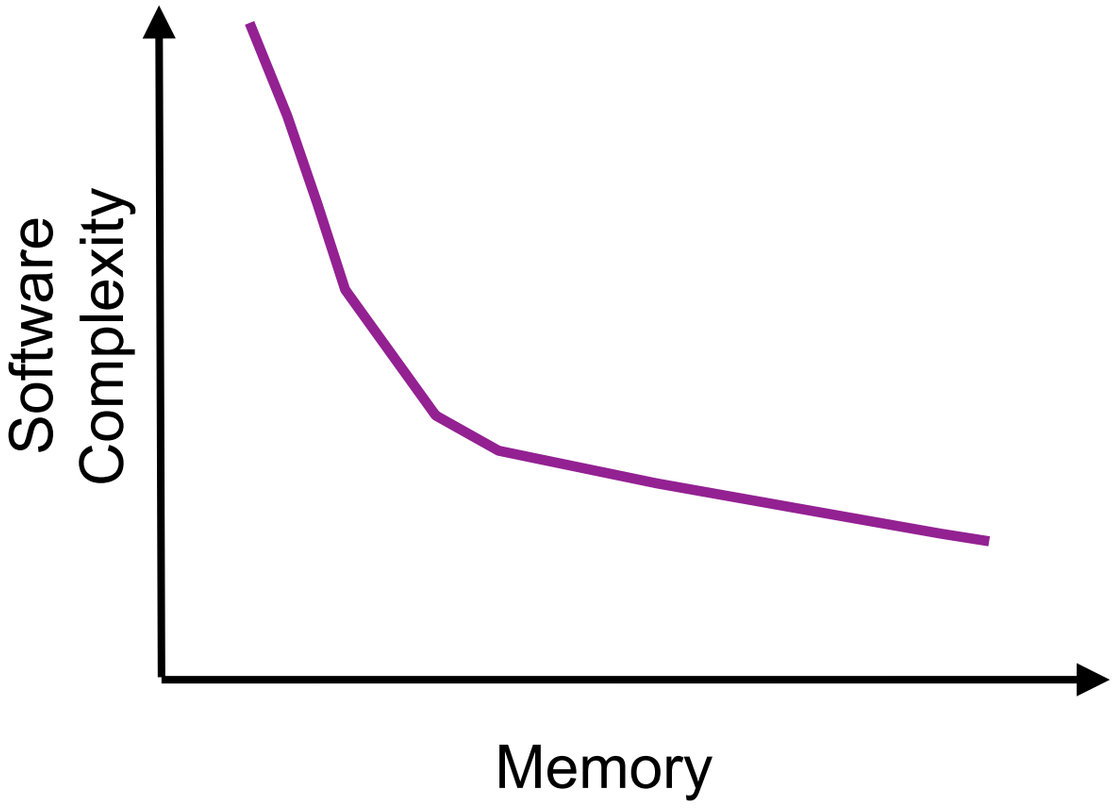
Open in a new window

7 Heresies of Technical Computing



Heresy #1

Computer memory will always be limited

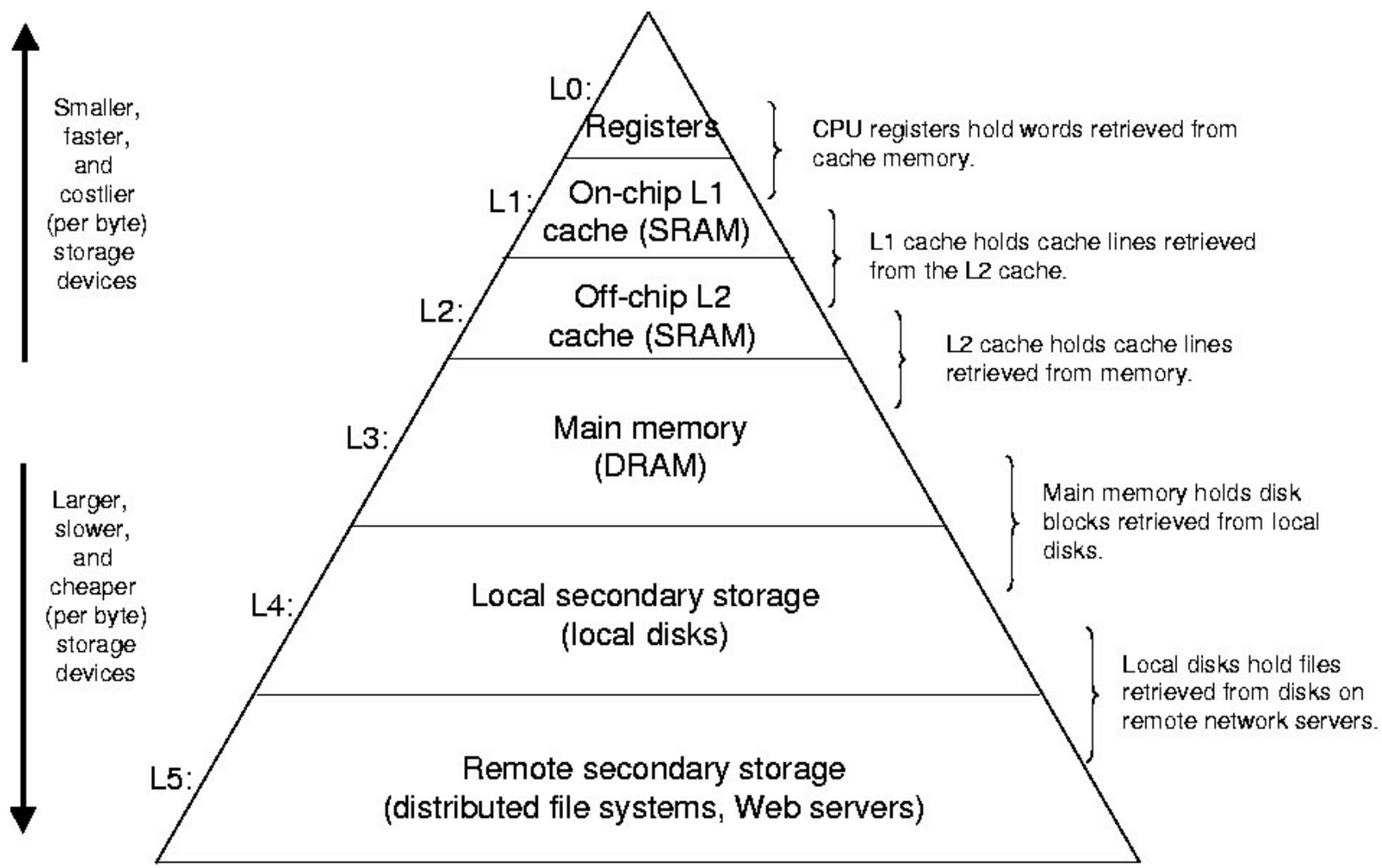


Heresy #1

- There will always be a larger problem than can fit in available memory.
- While automatic memory management can be useful --- programmer should be able to get access to lower-level if needed.

Heresy #2

There will always be a hierarchy of computer memory: fast -> slow

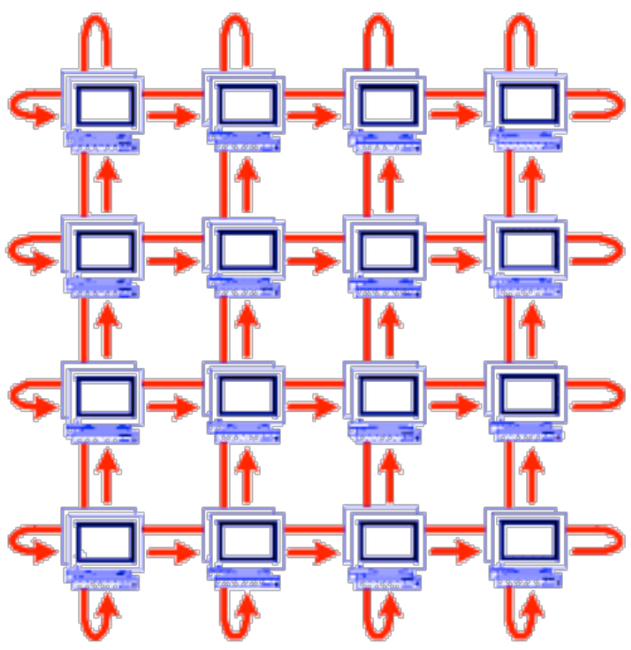


Heresy #2

- Single-memory model while powerful and useful, interferes with the ability to write programs that are optimal on modern hardware.
- Hiding the network layer can be disastrous to usability
- Different memory speeds require different approaches to manage

Heresy #3

There will be no magic concurrency solution

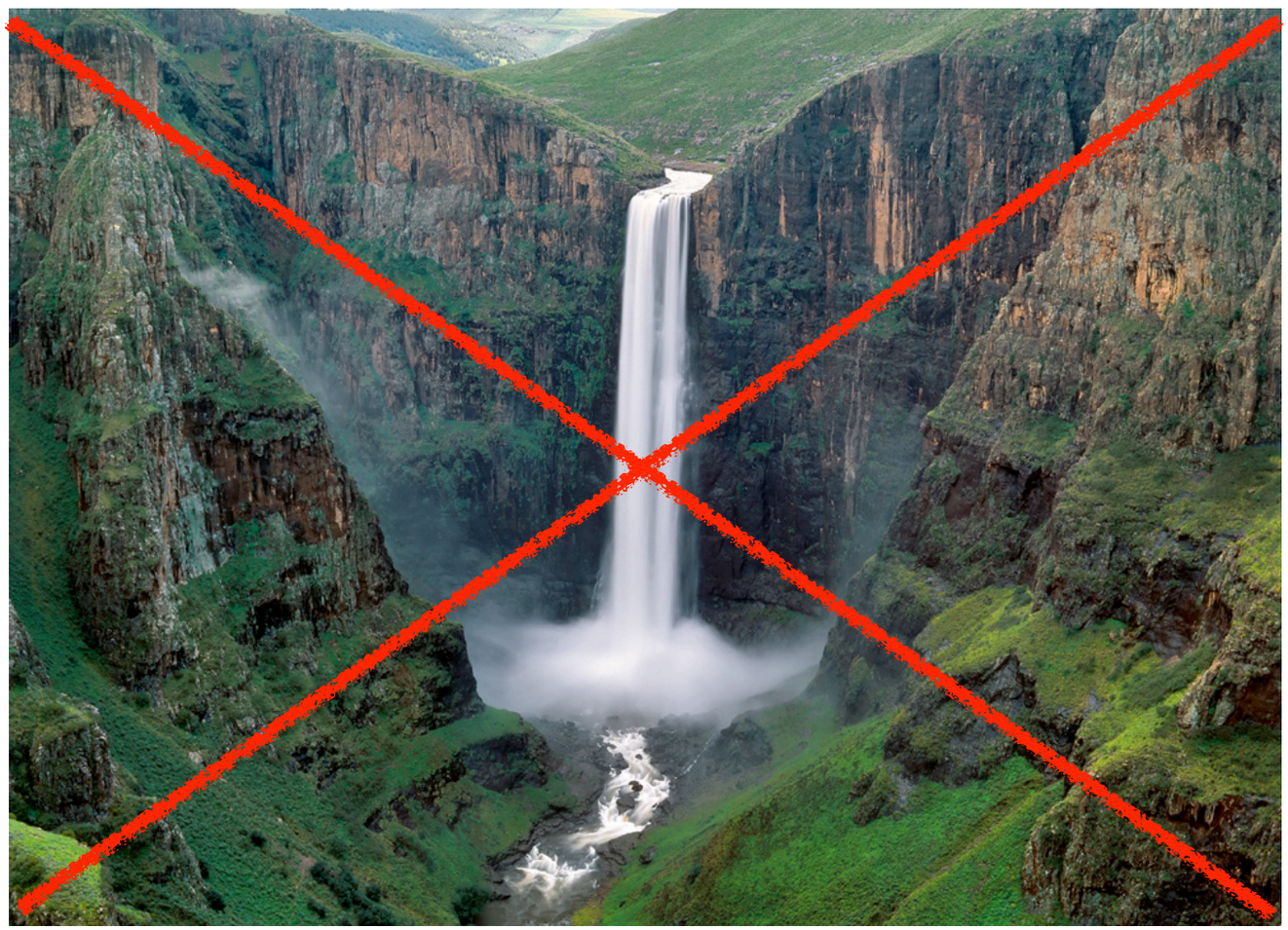


Heresy #3

- Python's GIL can be a blessing
- Use appropriate tools for the job
  - zeromq
  - multiprocessing
  - subprocess
  - threaded libraries (LAPACK, BLAS, FFT, etc.)
  - OpenMP in extensions

Heresy #4

Rapid prototyping via interactive programming leads to better results than extensive up-front planning

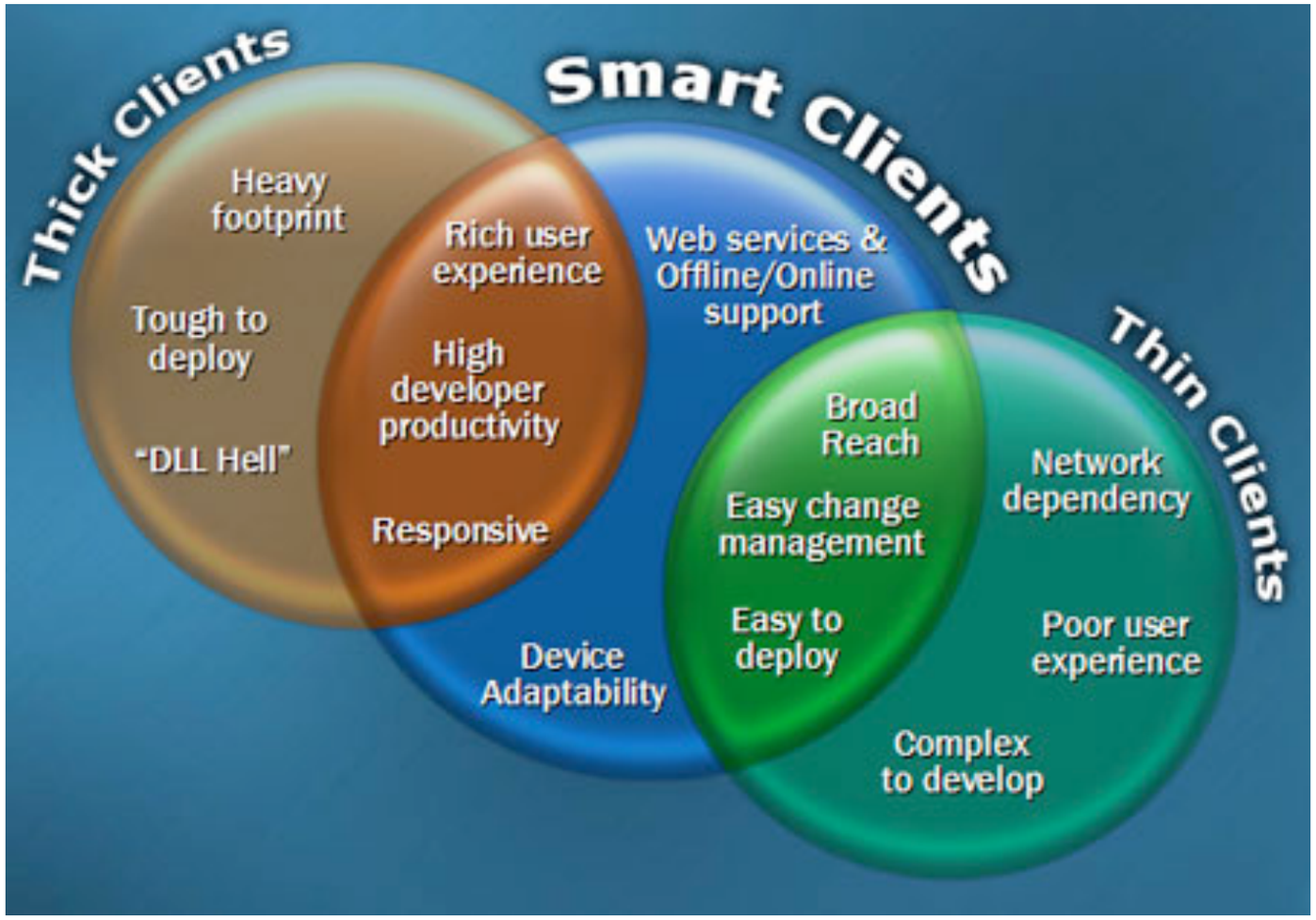


Heresy #4

- You rarely know exactly what you want the finished software to do
- Initial ideas lead to further refinement
- Interactively developing pieces of code is a form of test-driven development
- No hard data, but experience indicates it results in fewer bugs and easier to maintain software

Heresy #5

Multi-platform rich client software will always be relevant

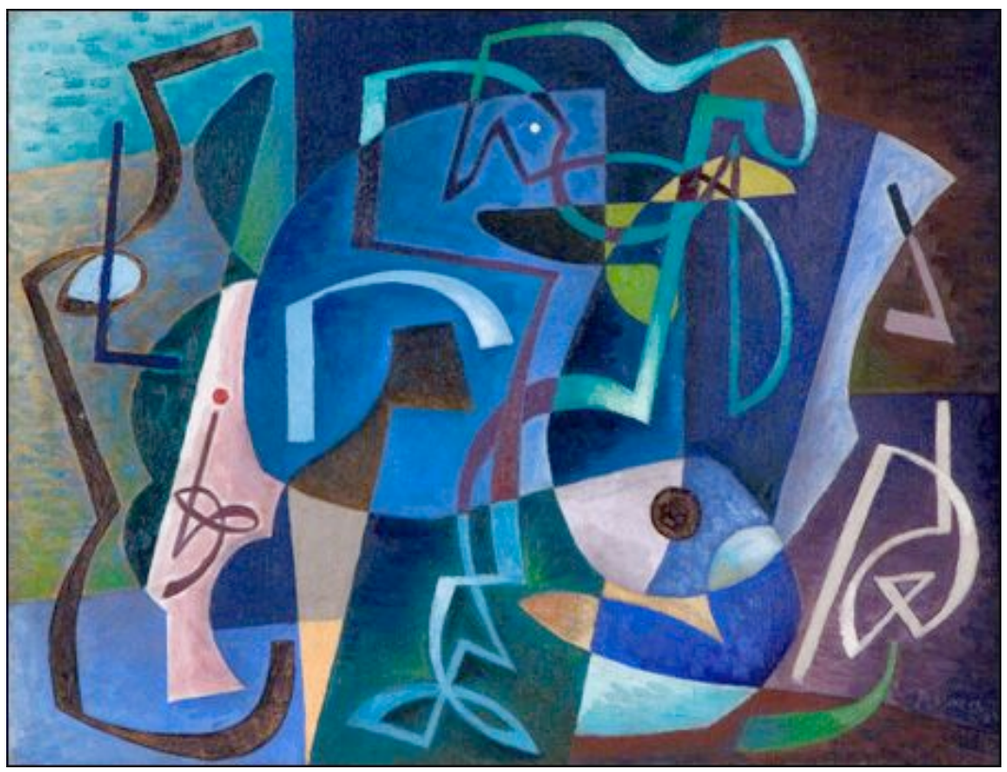


Heresy #5

- In-browser applications are still complicated and expensive to write well
- Data and I/O intensive applications still need a local client
- Browser will not be the only rich client running internet applications

Heresy #6

Object-oriented programming is over-used



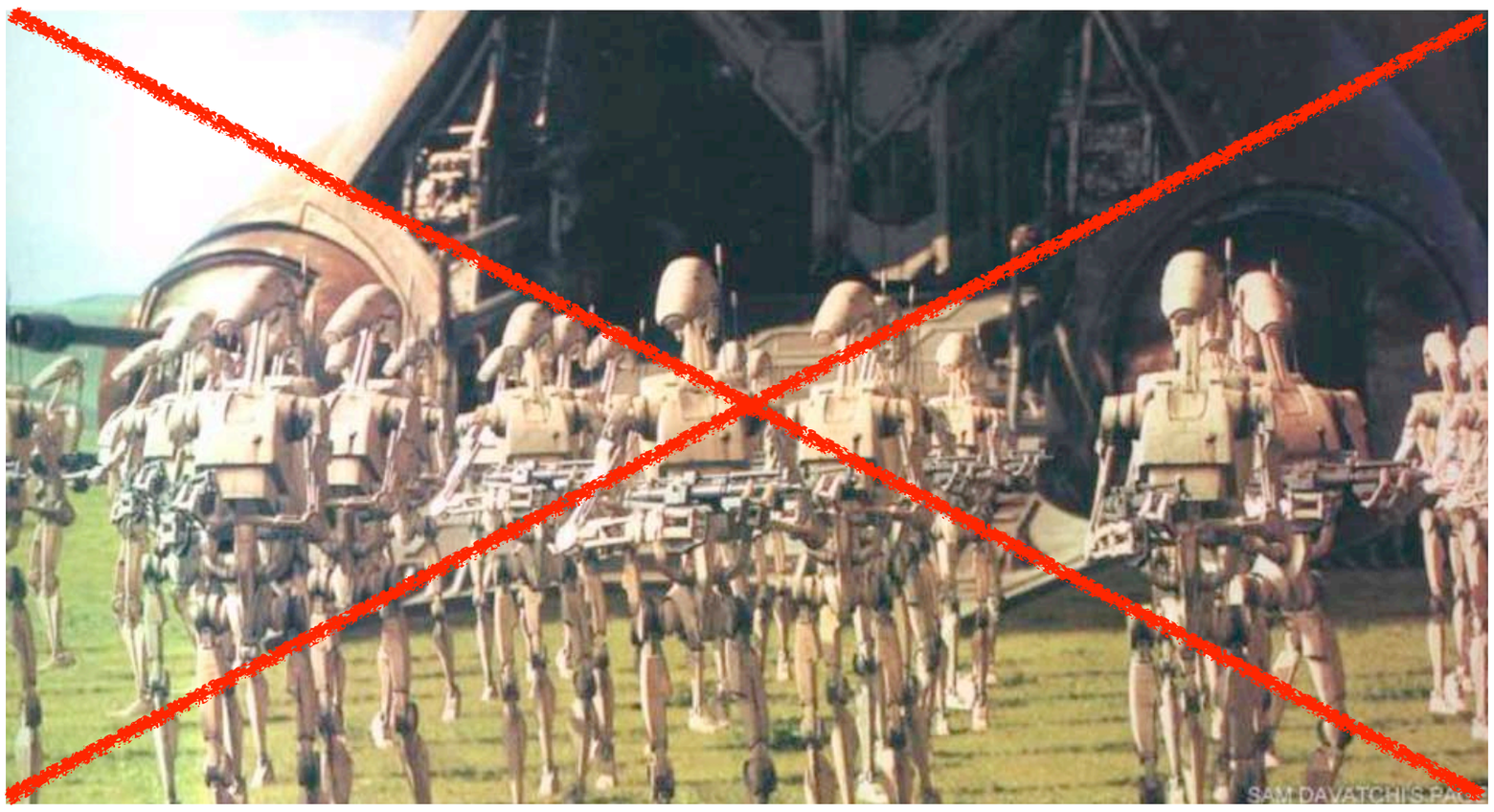
Heresy #6

Paul Graham (a partner in Y combinator, and author of Viaweb, the first web-based application):

> the purpose of OOP is to act as a “herding mechanism” which keeps mediocre programmers in mediocre organizations from “doing too much damage”. This is at the expense of slowing down productive programmers who know how to use more powerful and more compact techniques.

Heresy #7

The most important aspect of software development will always be people



Heresy #7

- Syntax matters
- Comments matter
- Unit testing matter
- Shared abstractions matter
- Communication matters
- DVCS matters
- Limitations of the human brain
- There will always be a level at which people are putting together the final level.

Heresy #7

Programs should be written for people to read, and only incidentally for machines to execute.

Structure and Interpretation of Computer Programs
Harold Abelson and Gerald Sussman

You need to have empathy not just for your users, but for your readers. It's in your interest, because you'll be one of them. Many a hacker has written a program only to find on returning to it six months later that he has no idea how it works.

Hackers and Painters
Paul Graham

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Brian Kernighan
co-author of "The C Programming Language"

Why Python?

"A programmer can be 5-10 times more productive in Python than in Java."

http://pythonconquerstheuniverse.wordpress.com/category/java-and-python/

Java	Python

`public class HelloWorld`
`{`
`  public static void main (String[] args)`
`  {`
`  System.out.println("Hello, world!");`
`  }`
`}`
	`print "Hello, world!"`
	———
	`print("Hello, world!") # Python version 3`

What is Python?

ONE LINER

Python is an interpreted programming language that allows you to do almost anything possible with a compiled language (C/C++/Fortran) without requiring all the complexity.

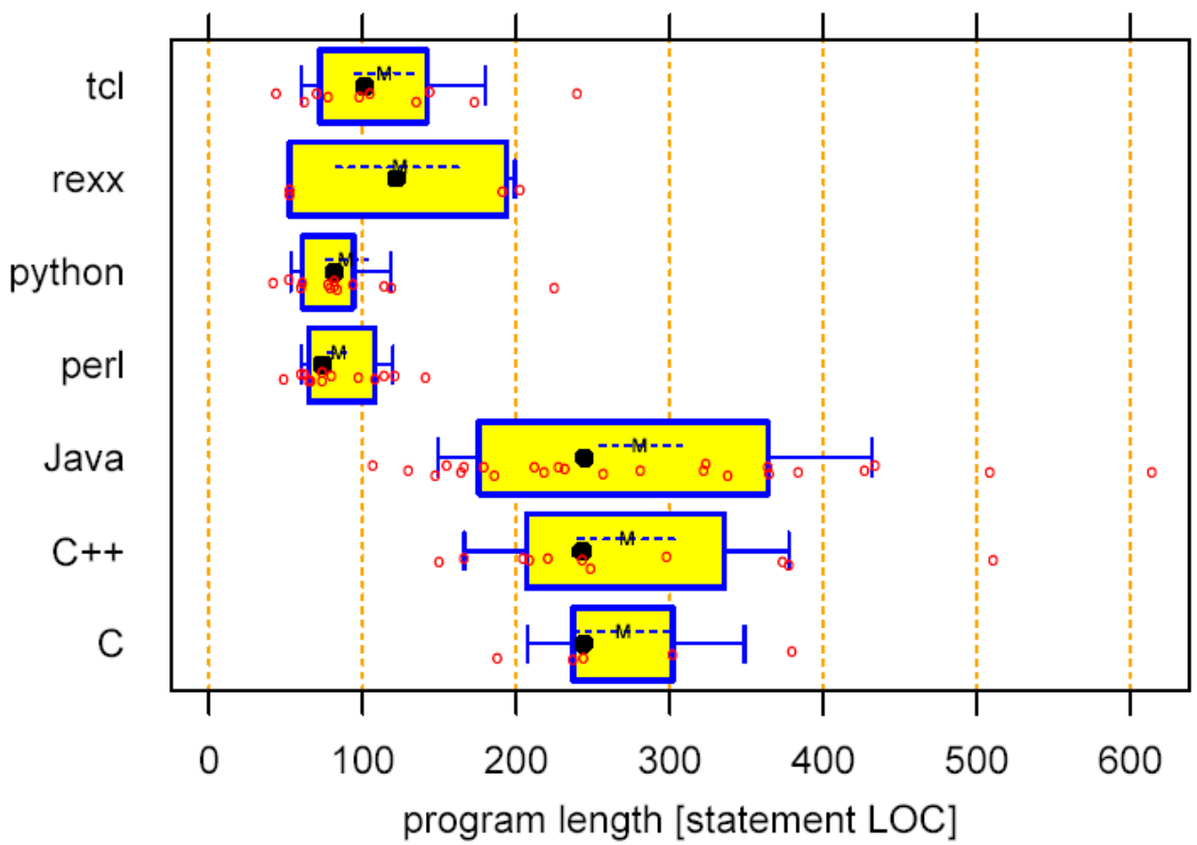
PYTHON HIGHLIGHTS

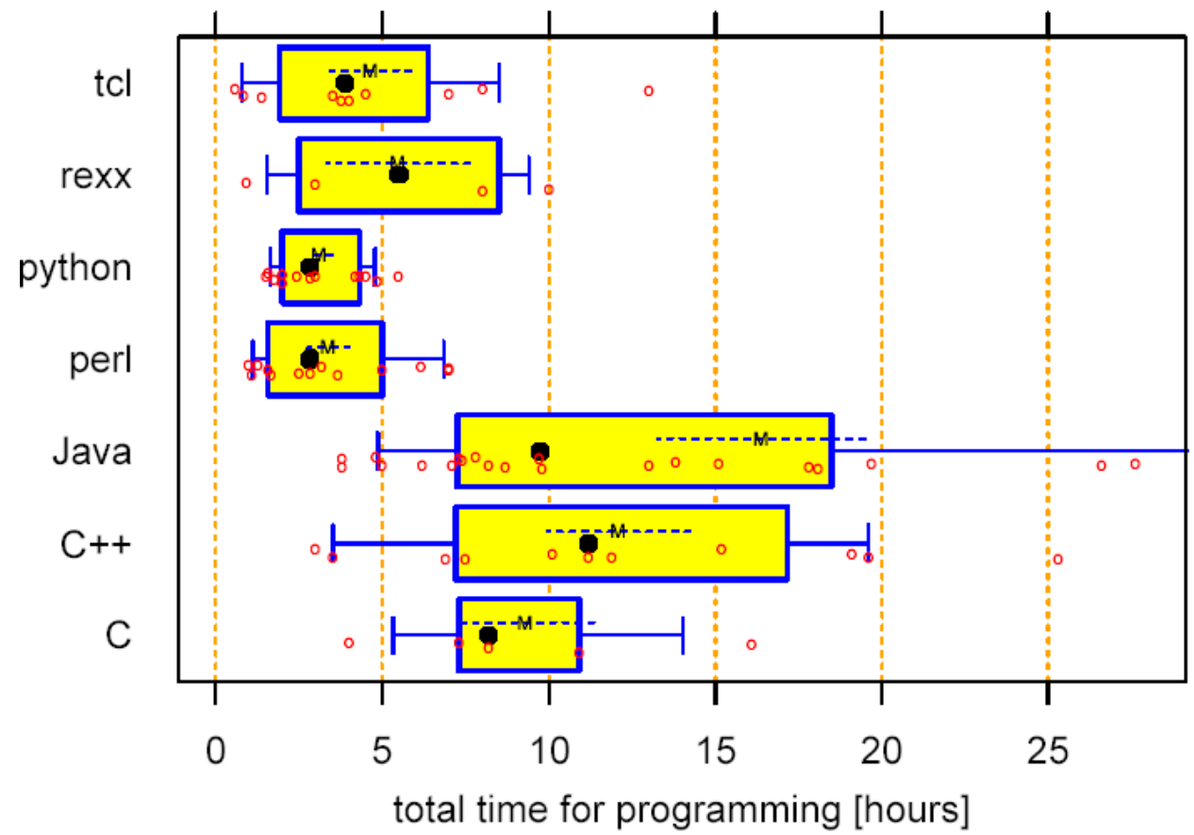
- **Automatic garbage collection**
- **Dynamic typing**
- **Interpreted and interactive**
- **Object-oriented**
- **“Batteries Included”**
- **Free**
- **Portable**
- **Easy to Learn and Use**
- **Truly Modular**

Program Size and Programmer Efficiency

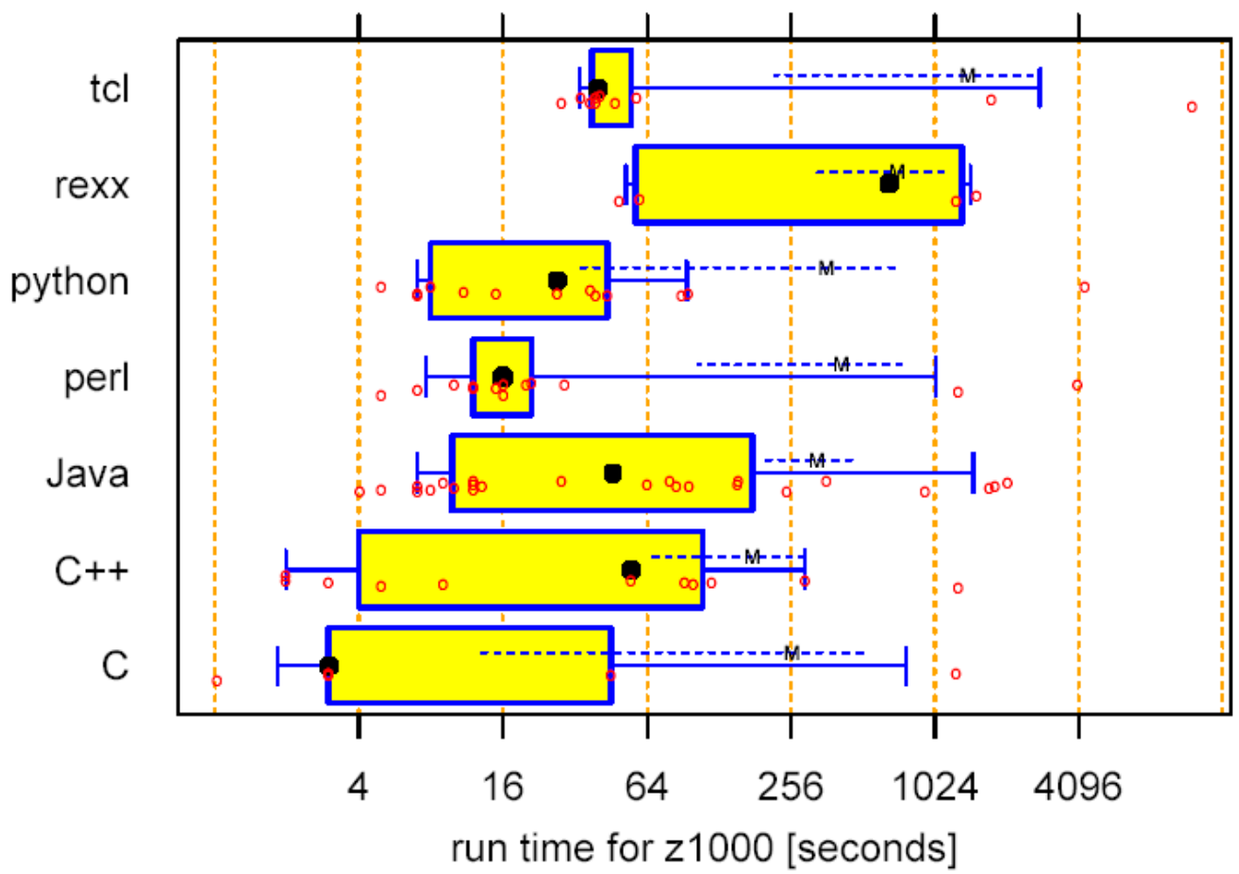
Quantitative language study based on 80 programmers implementing the same program. The program maps telephone numbers into strings of words according to a given dictionary and a fixed digit-to-character encoding.

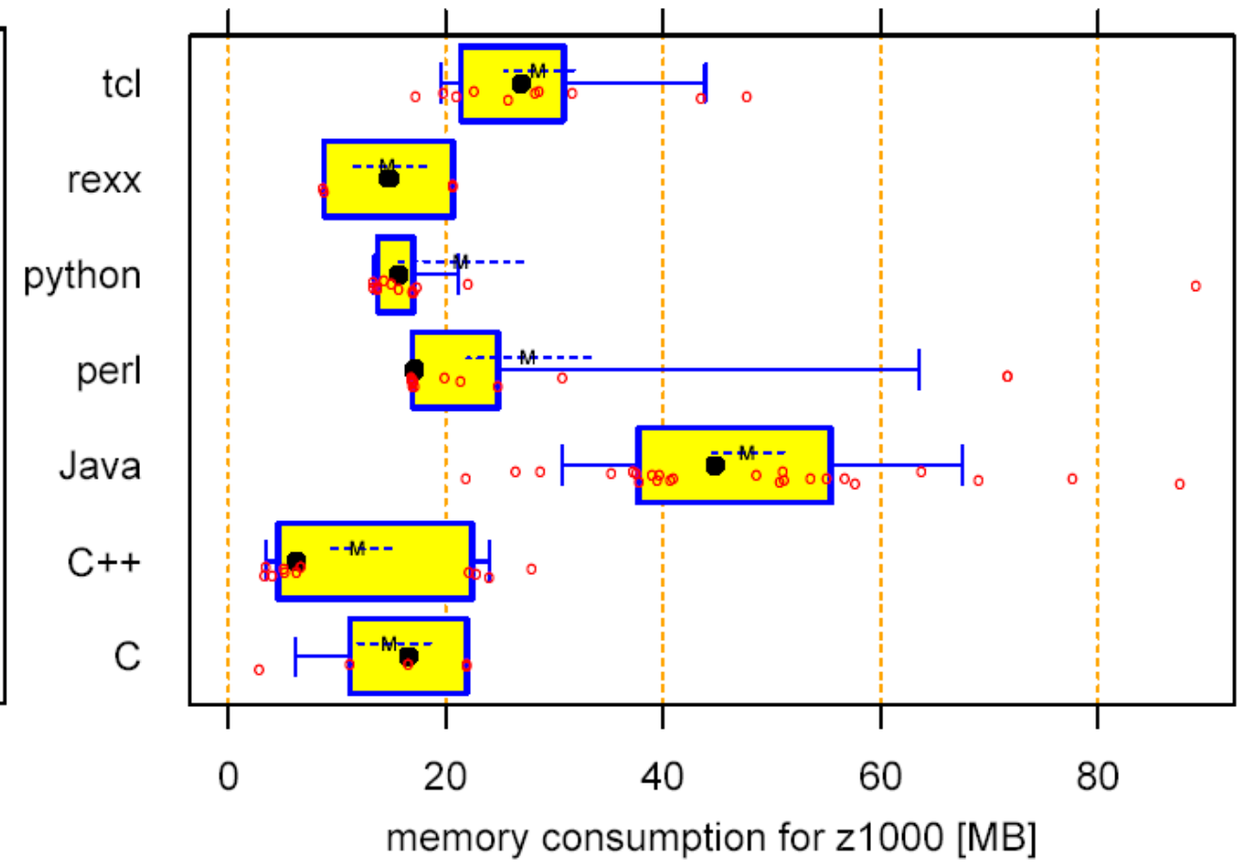
http://www.ipd.uka.de/~prechelt/Biblio/jccpprtTR.pdf





Language CPU/Memory





(Very) Short comparison

PYTHON

- Permissive philosophy
- Simple syntax.
- Excels as “glue” between other languages such as C/C++/Fortran
- Scriptible – allows expert users to extend the application natively.
- Supports complex numbers natively
- Supports multi-dimensional arrays
- Supports operator overloading

```
a + b * c
```

JAVA

- Strict protective rules intended to prevent errors
- Complex syntax.
- Designed as implementation language akin to C/C++
- Not scriptable, although secondary options exists (Jython, Bean Shell, Rhino)
- No support for complex numbers
- No native support for multi-dimensional arrays
- No support for operator overloading

```
a.add(b.multiply(c))
```

Output to a File Comparison

PYTHON

```
f = open('demo.txt','w')
print >> f, "hello", 42
f.close()
```

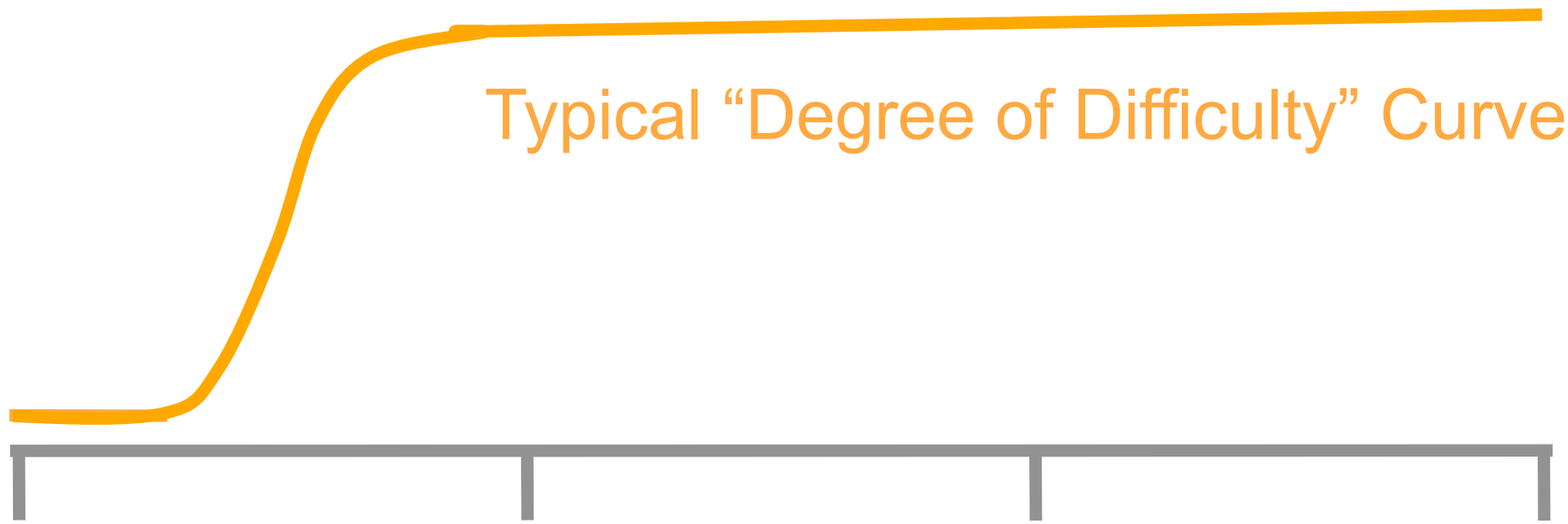
JAVA

```
import java.io.*;
public class HelloClient {
    public static void main(String args[]) {
            String hi = "Hello World!";
            int i = 42;
            FileOutputStream f = new FileOutputStream("demo.txt");
            PrintWriter pw = new PrintWriter(f);
            pw.println(hi);
            pw.println(Integer.toString(i));
            pw.close();
    }
}
```

Universe of Software Users

End User Scientist/Technician	Research Scientist	Research Scientist/ Programmer	Computer Architect
Only Uses Features provided by UI	Occasionally wants to modify an application to work “their way”	Research Scientist by day, programmer by night.	Cares (and knows) more about object hierarchies than science

Universe of Software Users



End User Scientist/Technician

Only Uses Features provided by UI

Research Scientist

Occasionally wants to modify an application to work “their way”

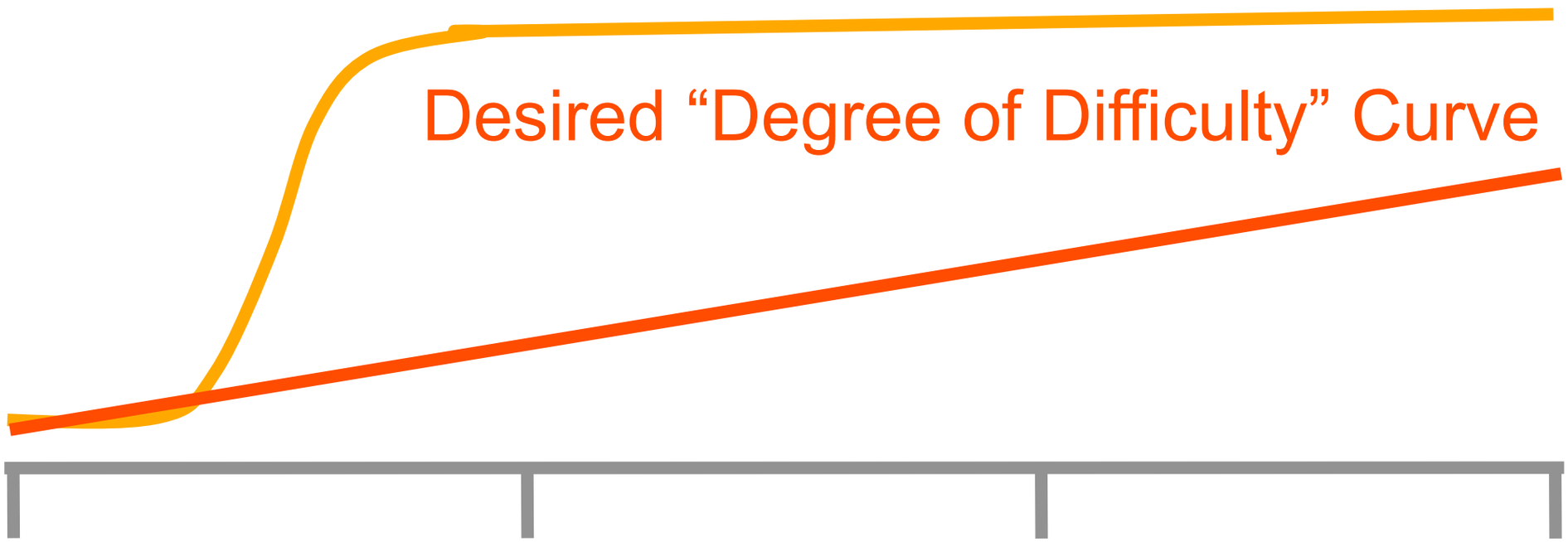
Research Scientist/ Programmer

Research Scientist by day, programmer by night.

Computer Architect

Cares (and knows) more about object hierarchies than science

Universe of Software Users



End User Scientist/Technician	Research Scientist	Research Scientist/ Programmer	Computer Architect
Only Uses Features provided by UI	Occasionally wants to modify an application to work “their way”	Research Scientist by day, programmer by night.	Cares (and knows) more about object hierarchies than science

August 16, 2007

The Forrester Wave™: Dynamic Programming Languages, Q3 2007

Python Is An All-Purpose Leader; PHP And Perl Are Popular Options

by Michael Goulde and Jeffrey Hammond
with Mike Gilpin, John R. Rymer, and Katie Smillie

EXECUTIVE SUMMARY

Forrester evaluated leading dynamic programming languages across 90 criteria and found that Python, PHP, and Perl have established leadership, thanks to their dynamic programming language capabilities and breadth of focus. **Python represents the best combination of features, breadth of applicability, and strong community.** PHP also had strong functionality and community but tends to be used for fewer types of applications. Perl has somewhat less functionality than Python or PHP but is also widely used and has a strong community. ECMAScript is a standard that has been implemented in several vendors' products, including JavaScript in Mozilla and JScript in Internet Explorer. It is a Leader, but it is typically used for building Web pages, and although it has good features for accomplishing that purpose, it is seldom used for other purposes. Ruby is also a Strong Performer with solid functionality and wide applicability, but because of its relative newness, it has a smaller, though enthusiastic, following.

Who is Using Python?

WALL STREET

Banks and hedge funds rely on Python for their high speed trading systems, data analysis, and visualization.

HOLLYWOOD

Digital animation and special effects:

- Industrial Light and Magic
- Imageworks
- Tippett Studios
- Disney
- Dreamworks

PETROLEUM INDUSTRY

Geophysics and exploration tools:

- ConocoPhillips
- Shell

GOOGLE

One of top three languages used at Google along with C++ and Java. Guido works there.

YOUTUBE

Highly scalable web application for delivering video content.

REDHAT

Anaconda, the Redhat Linux installer program, is written in Python.

PROCTER & GAMBLE

Fluid dynamics simulation tools.

How we use Python?

IPython

```
PyLab
Enthought Python Distribution (2.5.2001) -- http://code.enthought.com

Python 2.5.2 |EPD 2.5.2001| (release25-maint:60919M, Feb 21 2008, 10:31:43) [MSC
 v.1310 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 0.8.1 -- An enhanced Interactive Python.
?         -> Introduction to IPython's features.
%magic    -> Information about IPython's 'magic' % functions.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

  Welcome to pylab, a matplotlib-based Python environment.
  For more information, type 'help(pylab)'.

In [1]: a = array([1,2,3,4])

In [2]: a + 10
Out[2]: array([11, 12, 13, 14])

In [3]: _
```

Structured Arrays

Elements of array can be any fixed-size data structure.

```
Name char[10]
Age int
Weight double
```

Brad	Jane	John	Fred
33	25	47	54
135.0	105.0	225.0	140.0
Henry	George	Brian	Amy
29	61	32	27
154.0	202.0	137.0	187.0
Ron	Susan	Jennifer	Jill
19	33	18	54
188.0	135.0	88.0	145.0

Example

```
>>> from numpy import dtype, empty
# structured data format
>>> fmt =dtype ([('name', 'S10'),
                ('age', int),
                ('weight', float)])

>>> a = empty ((3,4), dtype=fmt)
>>> a.itemsize
22
>>> a['name'] = [['Brad', ... , 'Jill']]
>>> a['age'] = [[33, ... ,  54]]
>>> a['weight'] = [[135, ... , 145]]
>>> print a
[[('Brad', 33, 135.0)
   ...
  ('Jill', 54, 145.0)]]
```

Nested Datatype

Time	Size	Position				Gain	Samples (2048) ...			
		Az	El	Type	ID					
1172581077060	4108	0.715594	–0.148407	1	4	40	561	1467	997	–30
1172581077091	4108	0.706876	–0.148407	1	4	40	7	591	423	
1172581077123	4108	0.698157	–0.148407	1	4	40	49	–367	–565	–35
1172581077153	4108	0.689423	–0.148407	1	4	40	–55	–953	–1151	–30
1172581077184	4108	0.680683	–0.148407	1	4	40	–719	–1149	–491	38
1172581077215	4108	0.671956	–0.148407	1	4	40	–1503	–683	661	149
1172581077245	4108	0.663232	–0.148407	1	4	40	–2731	–281	2327	291
1172581077276	4108	0.654511	–0.148407	1	4	40	–3493	–159	3277	380
1172581077306	4108	0.645787	–0.148407	1	4	40	–3255	–247	3145	385
1172581077339	4108	0.637058	–0.148407	1	4	40	–2303	–101	2079	247
1172581077370	4108	0.628321	–0.148407	1	4	40	–1495	–553	571	107
1172581077402	4108	0.619599	–0.148407	1	4	40	–955	–1491	–1207	–25
1172581077432	4108	0.61087	–0.148407	1	4	40	–875	–3009	–2987	–93
1172581077463	4108	0.602148	–0.148407	1	4	40	–491	–3681	–4193	–175
1172581077497	4108	0.593438	–0.148407	1	4	40	167	–3501	–4573	–250
1172581077547	4108	0.584696	–0.148407	1	4	40	1007	–2613	–4463	–303
1172581077599	4108	0.575972	–0.148407	1	4	40	1261	–2155	–4299	–339
1172581077650	4108	0.567244	–0.148407	1	4	40	1537	–2633	–4945	–367
1172581077702	4108	0.558511	–0.148407	1	4	40	1105	3701	6139	423

Nested Datatype

```
dt = dtype([('time', uint64),
            ('size', uint32),
            ('position', [('az', float32),
                          ('el', float32),
                          ('region_type', uint8),
                          ('region_ID', uint16)]),
            ('gain', np.uint8),
            ('samples', np.int16, 2048)])

data = np.fromfile(f, dtype=dt)
```

Structured Arrays

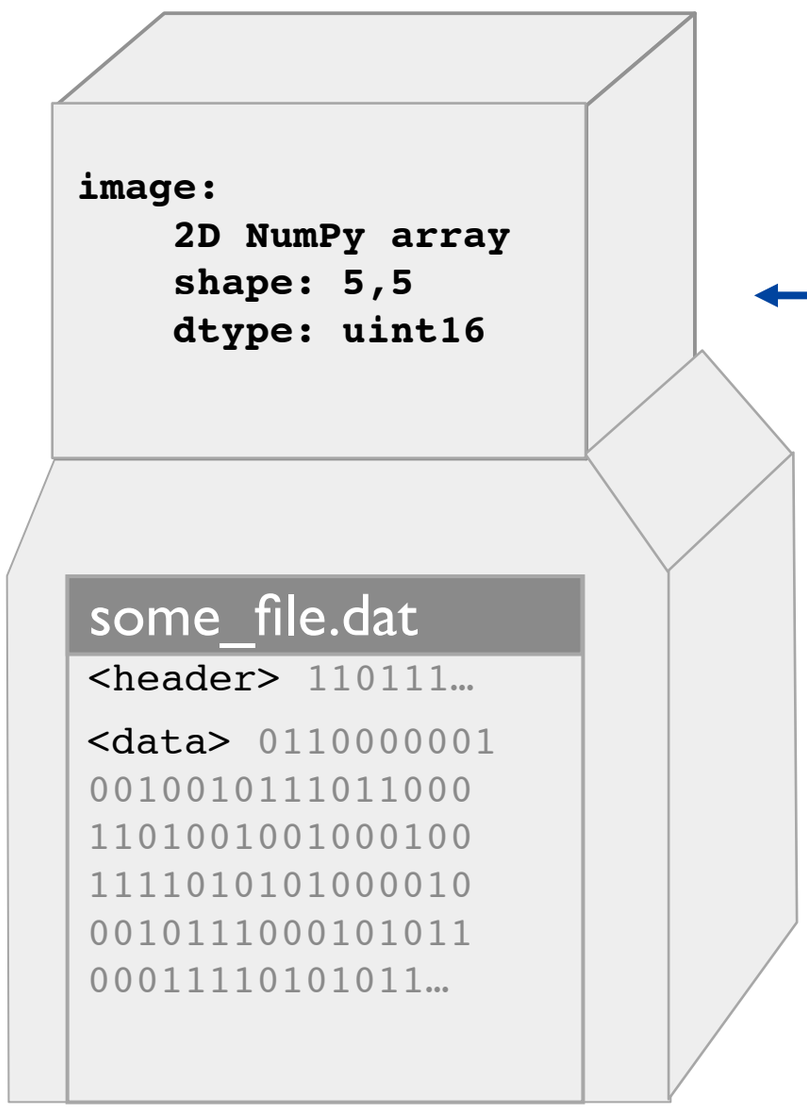
Name **char[12]**	Time **int64**	Value **float32**
MSFT_profit	10	6.20
GOOG_profit	12	-1.08
MSFT_profit	18	8.40
INTC_profit	25	-0.20
⋮	⋮	⋮
GOOG_profit	1000325	3.20
GOOG_profit	1000350	4.50
INTC_profit	1000385	-1.05
MSFT_profit	1000390	5.60

Example

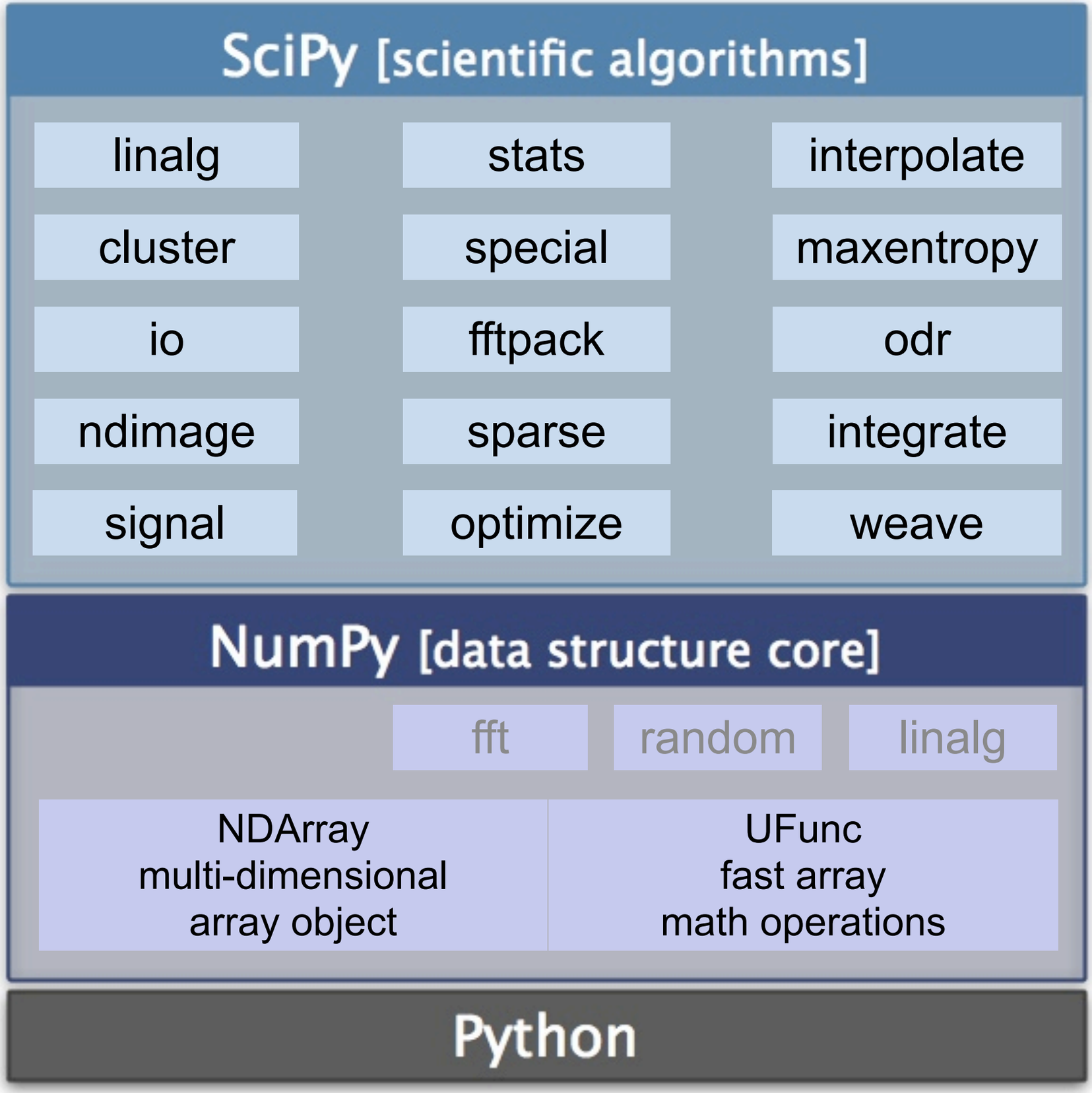
```
>>> import numpy as np
>>> fmt =np.dtype ([('name', 'S12'),
                   ('time', np.int64),
                   ('value', np.float32)])

>>> vals = [('MSFT_profit', 10, 6.20),
            ('GOOG_profit', 12, -1.08),


            ('GOOG_profit', 1000385, -1.05)
            ('MSFT_profit', 1000390, 5.60)]

>>> arr = np.array(vals, dtype=fmt)
# or
>>> arr = np.fromfile('db.dat', dtype=fmt)
# or
>>> arr = np.memmap('db.dat', dtype=fmt,
                    mode='c')
```

MSFT_profit	10	6.20	GOOG_profit	12	-1.08	• • •	INTC_profit	1000385	-1.05	MSFT_profit	1000390	5.60

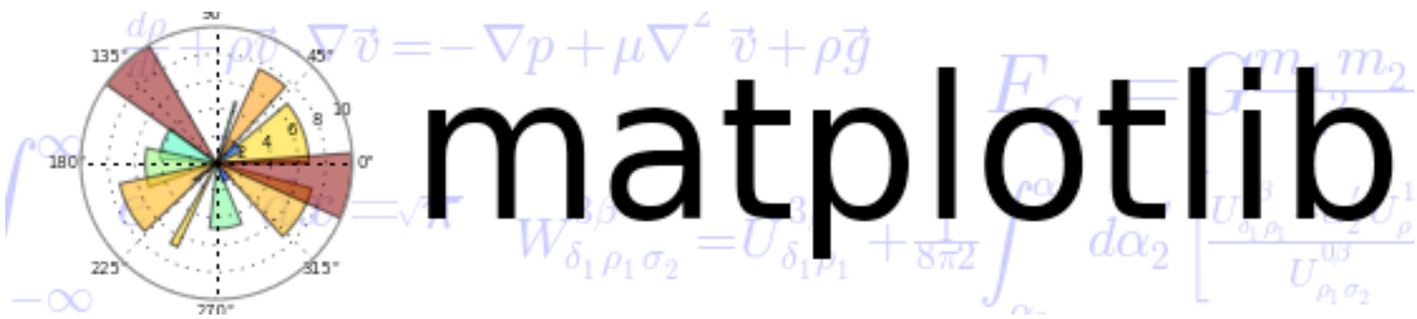
Memory Mapped Example




```
# Create a "memory mapped array where
# the array data is stored in a file on
# disk instead of in main memory.
>>> from numpy import memmap
>>> image =('some_file.dat',
            dtype = uint16,
            mode= 'r+',
            mode= (5,5),
            offset = header_size)
# Standard array methods work.
>>> mean_value = image.mean()

# Standard math operations work.
# The resulting scaled_image *is*
# stored in main memory. It is a
# standard numpy array.
>>> scaled_image = image * .5
```

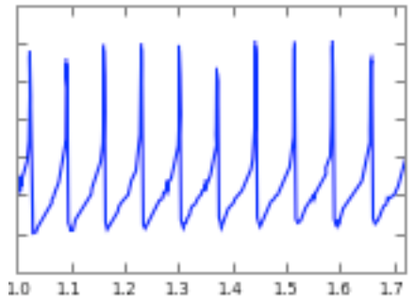

SciPy [scientific algorithms]
linalg
stats
interpolate
cluster
special
maxentropy
io
fftpack
odr
ndimage
sparse
integrate
signal
optimize
weave
NumPy [data structure core]
fft
random
linalg
NDArray
multi-dimensional
array object
UFunc
fast array
math operations
Python

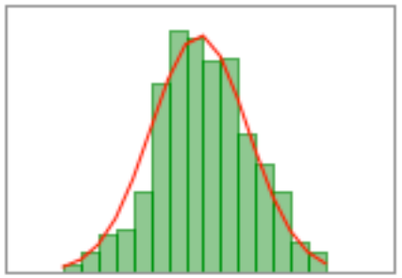
Static Plotting

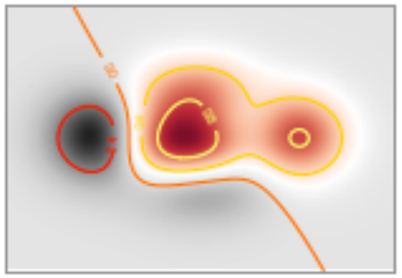


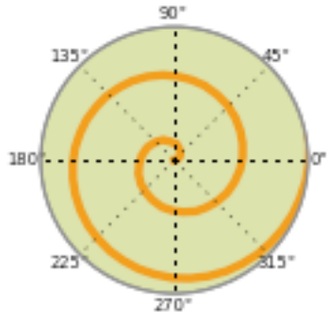
John Hunter

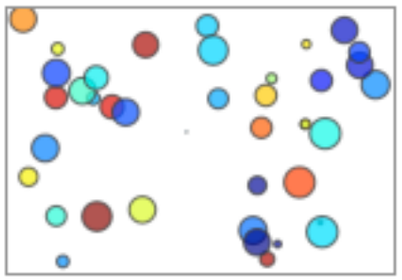
2001



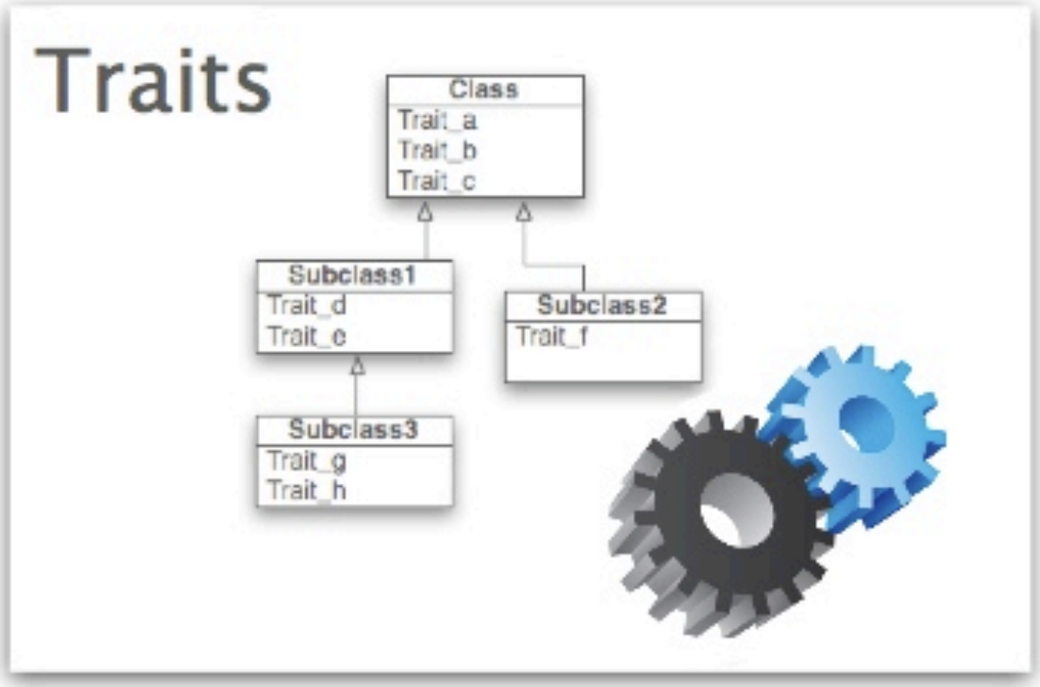


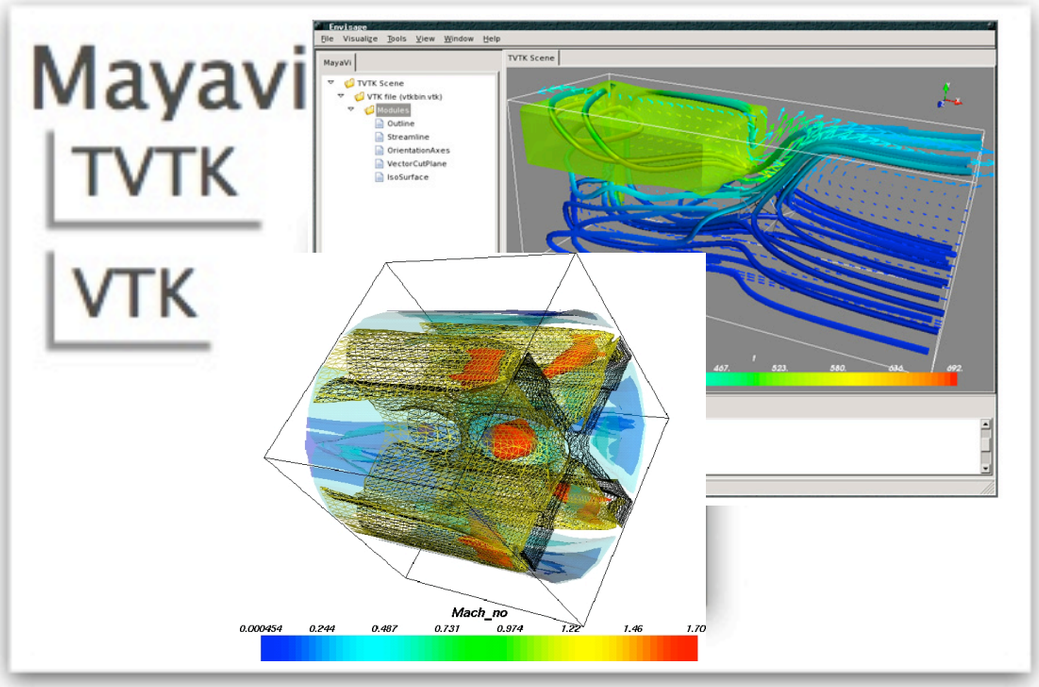


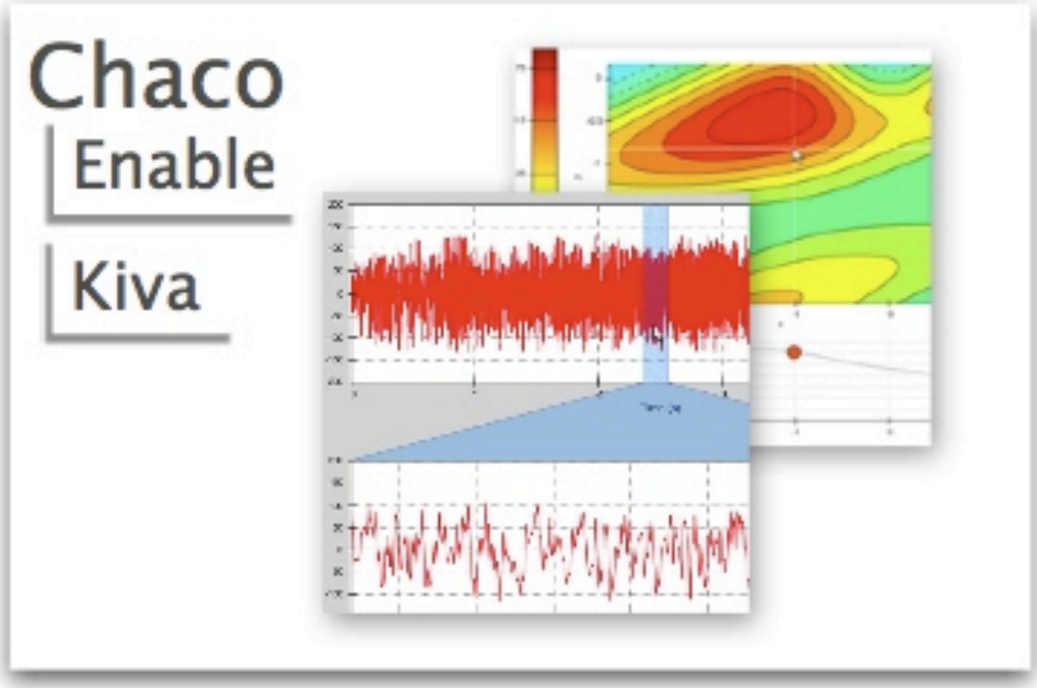


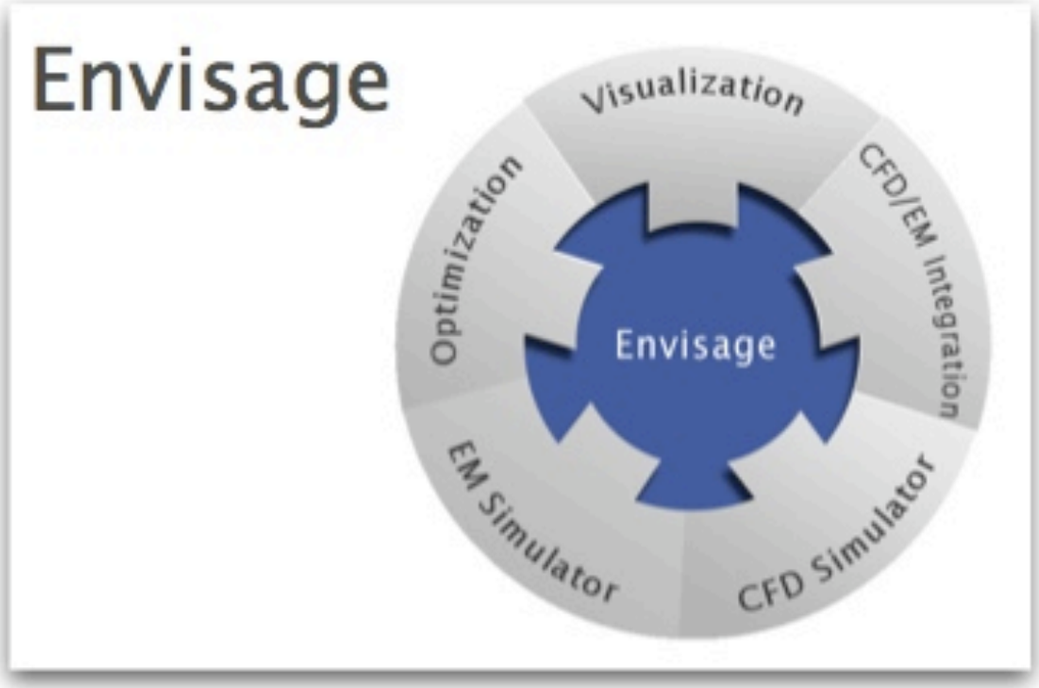


Enthought Tool Suite

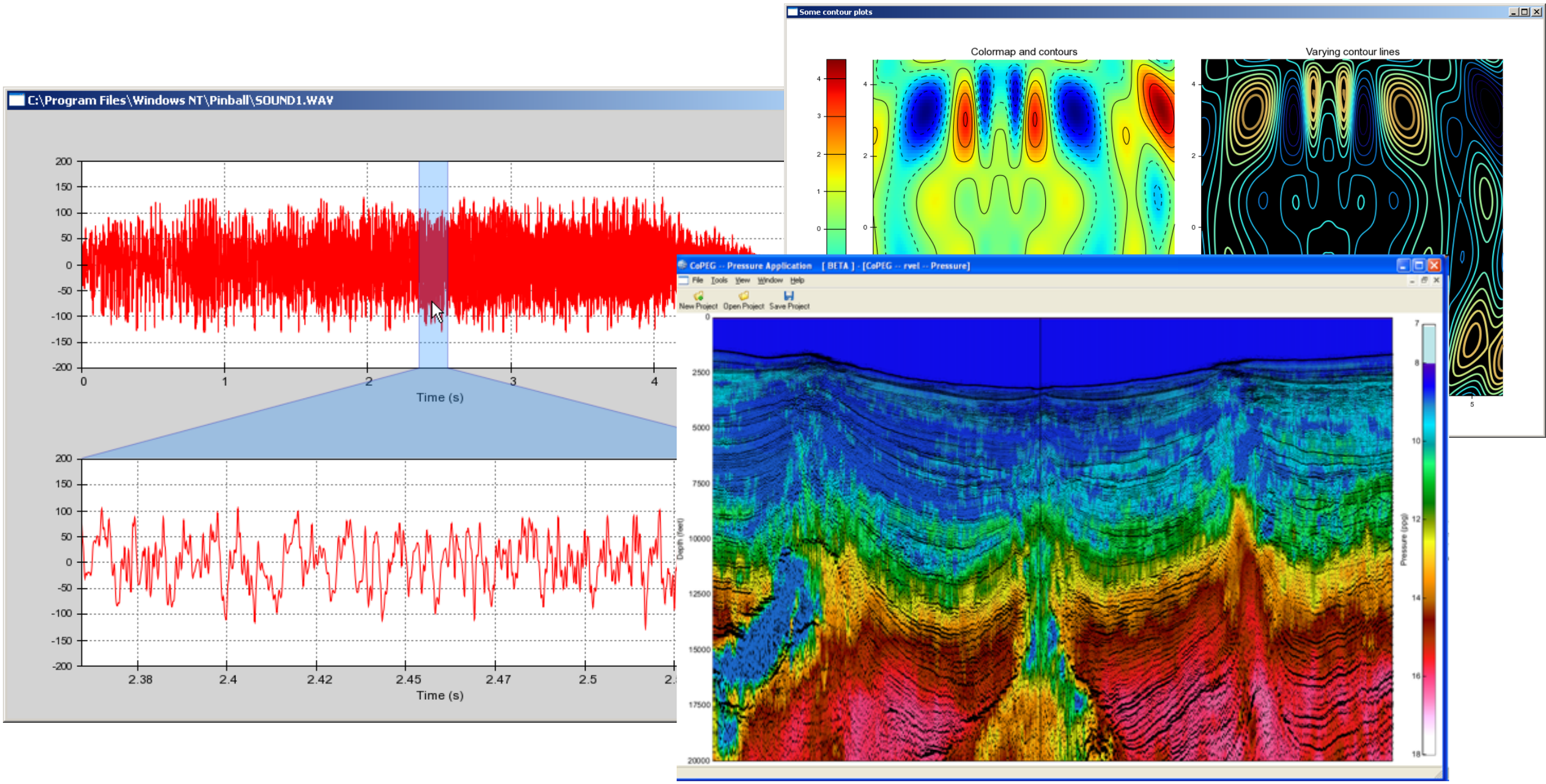






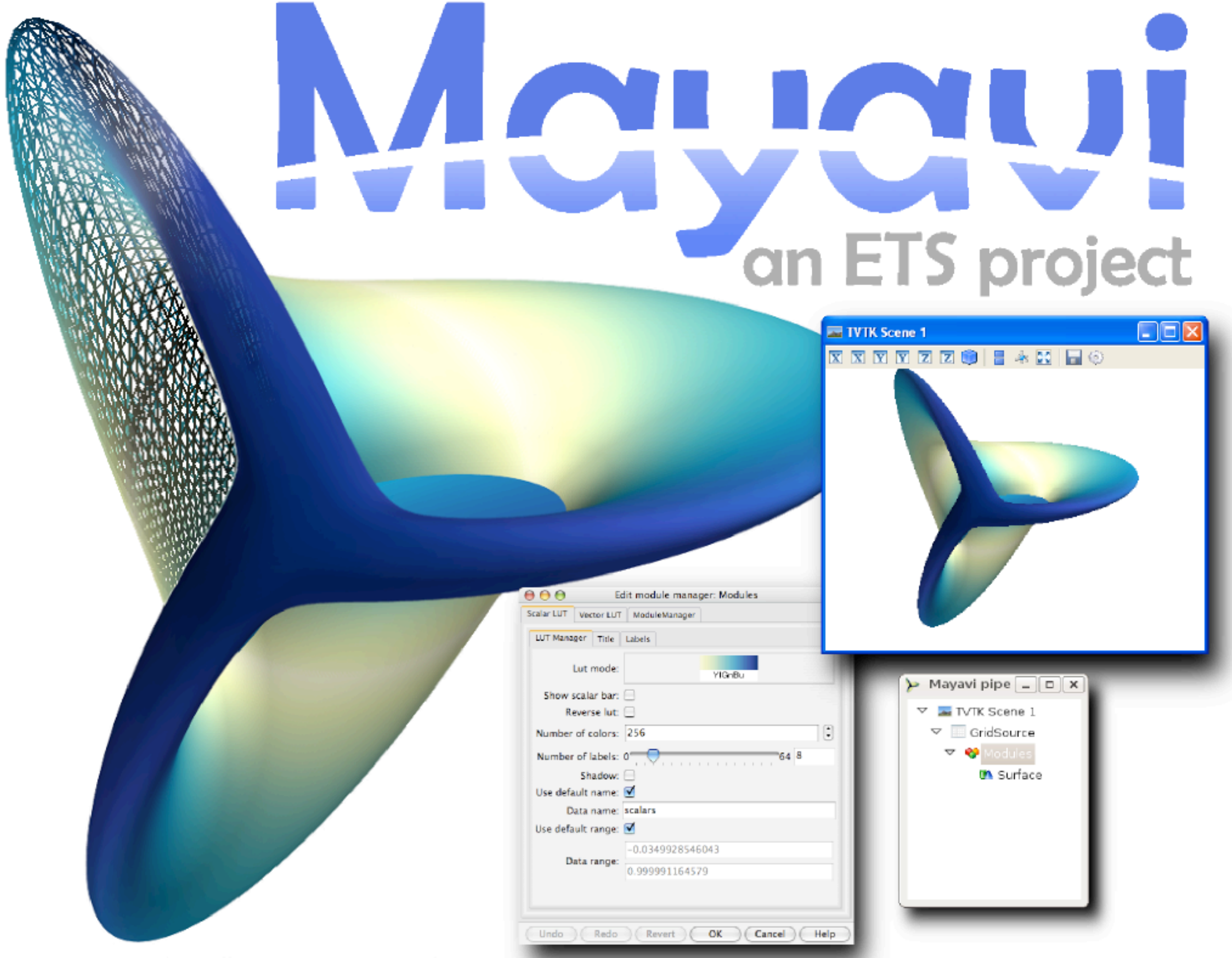


Chaco: Interactive Graphics

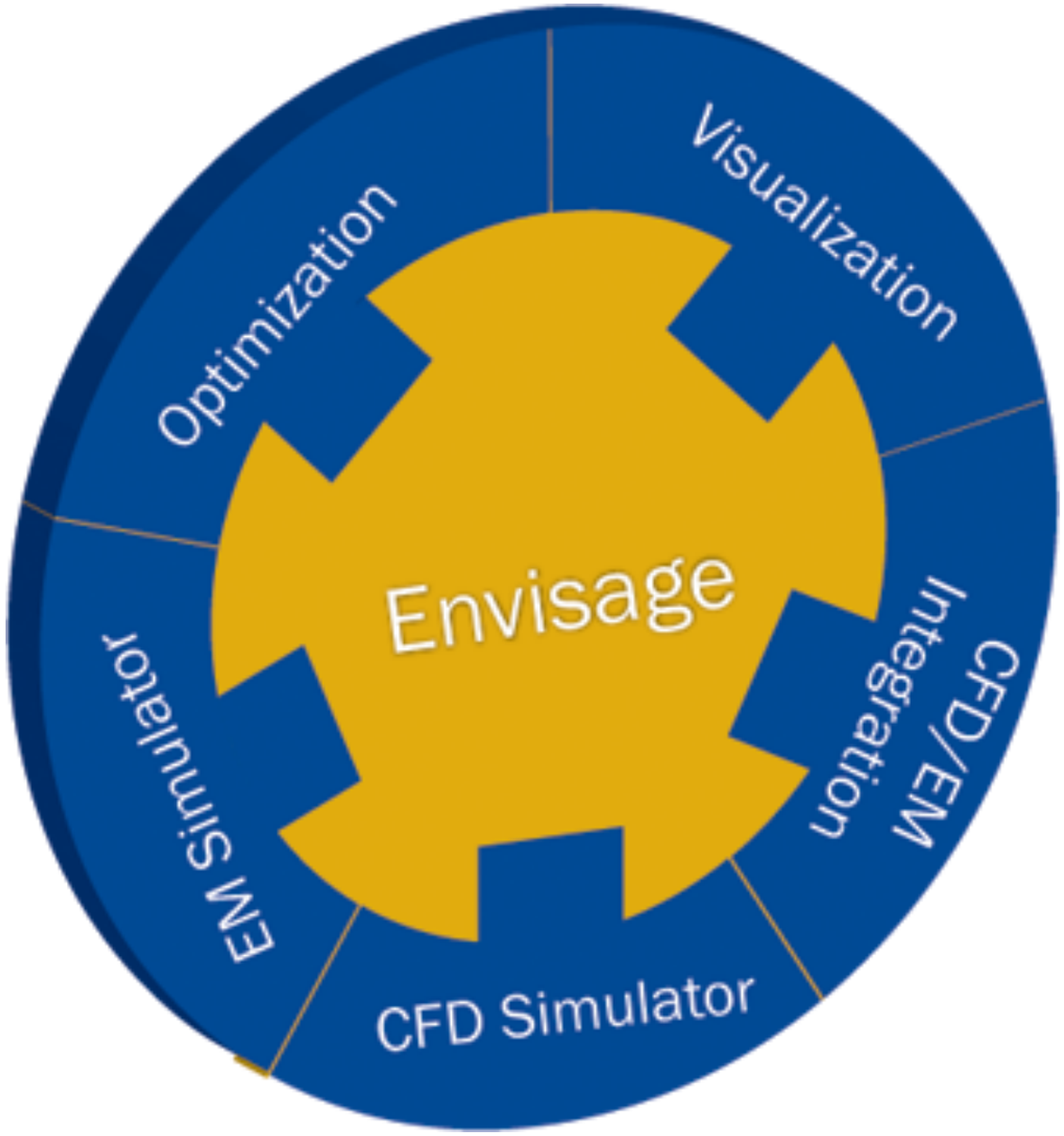


Mayavi

an ETS project

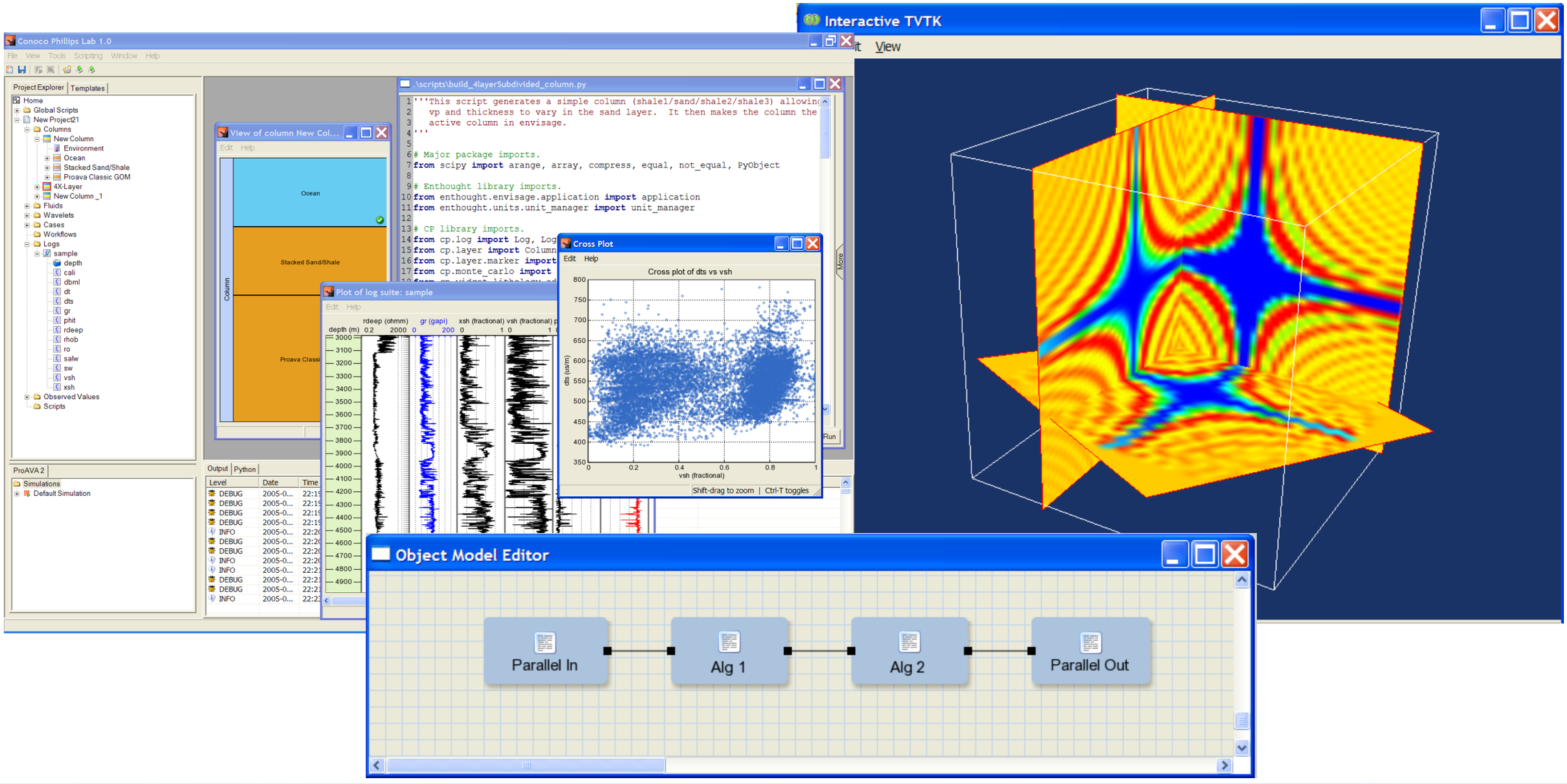


Multiple Plug-ins. One Application

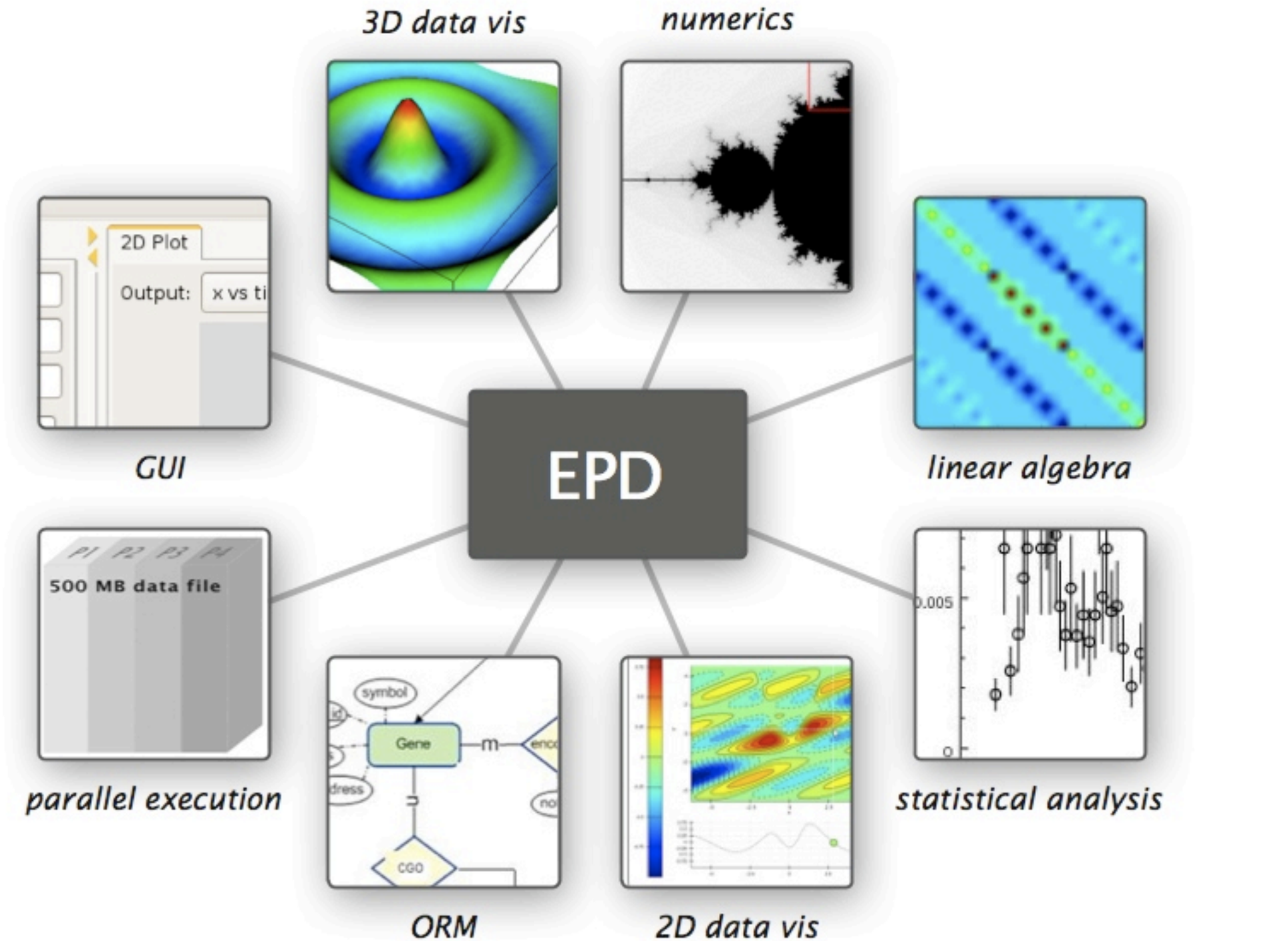


Envisage Application Framework Goals

- Extensible
- Scriptable
- “Humane” Interface Tools built-in
- Easy Deployment



Enthought Python Distribution (EPD)



Software Application Layers

Domain Specific GUI Applications
Semiconductor, Fluid Dynamics, Seismic Modeling, Financial, etc.

ETS (App construction)
Traits, Chaco, Envisage, Mayavi, etc.

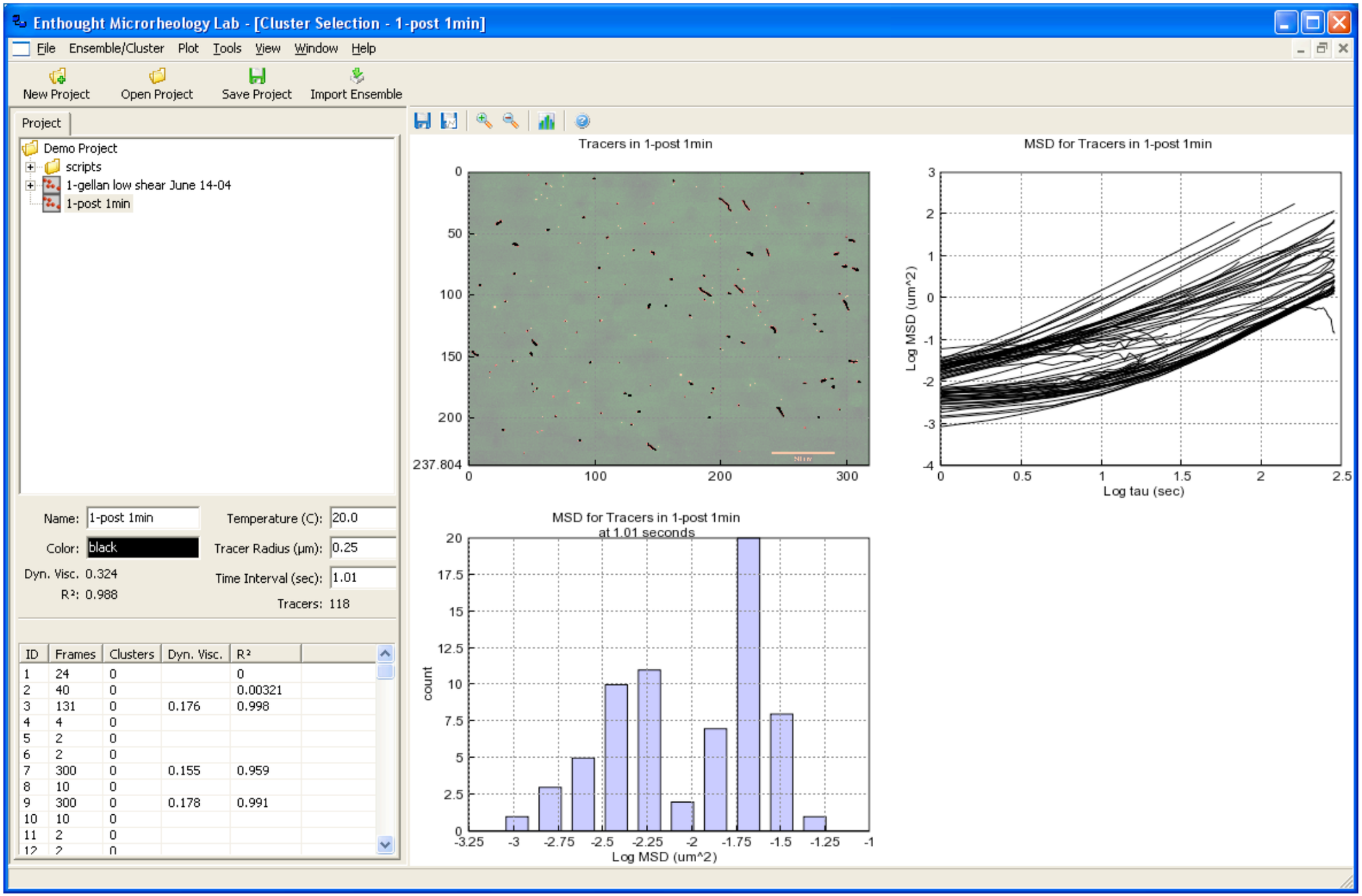
SciPy (Scientific Algorithms)

NumPy (Array Mathematics)

3rd Party Libraries
wxPython
VTK, etc.

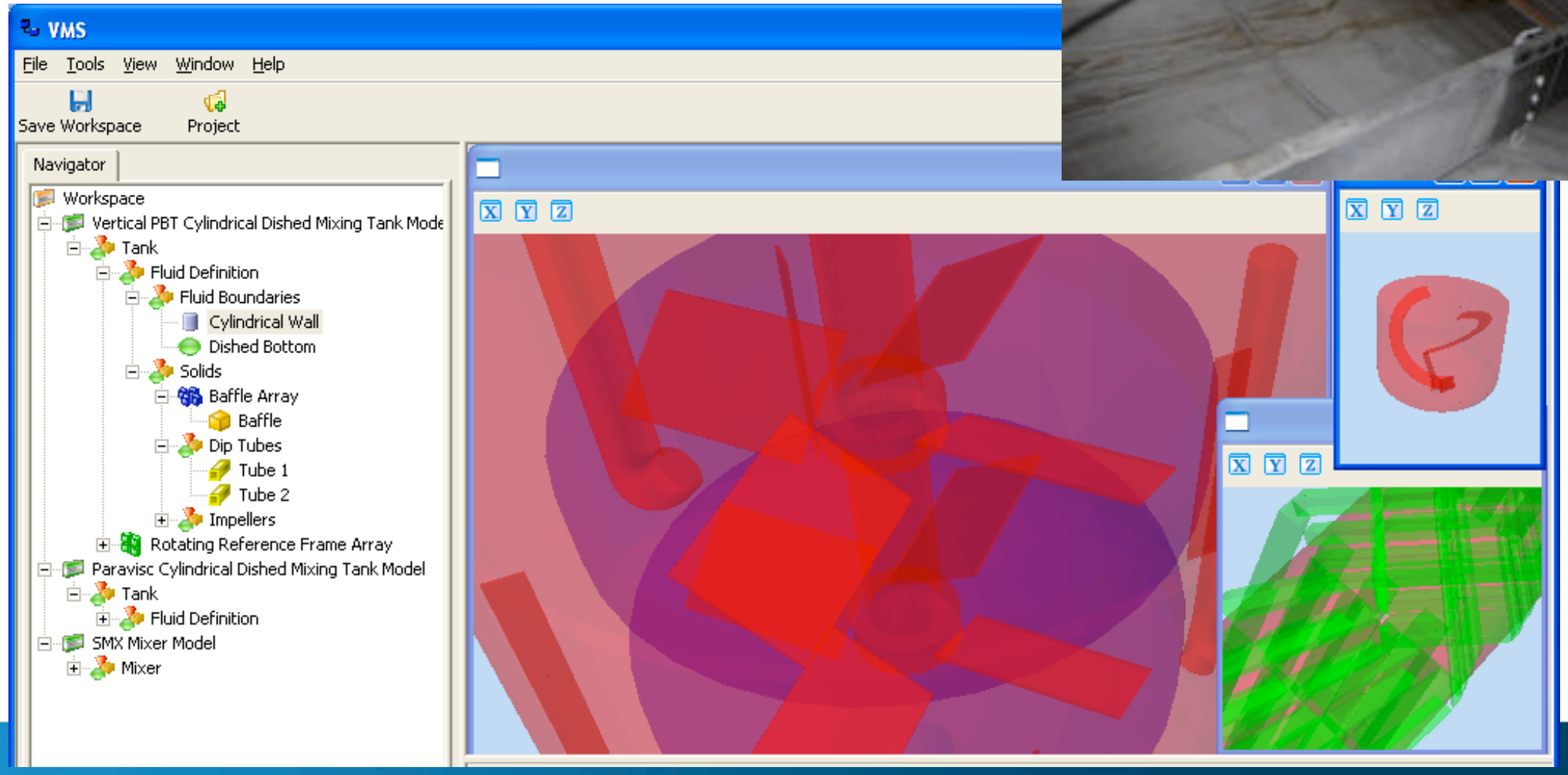
Python

Fluid Characterization -- Micro-rheology



VMS – Virtual Mixing System

- ***Design Drawings***
- ***Computational Fluid Dynamics***
- ***Parallel Simulation***
- ***Data Visualization***



VMA – Virtual Mixing Advisor

VMA - New Project *

File Edit Preferences Tools Help

New Project
process 1
process step 1

process step 1

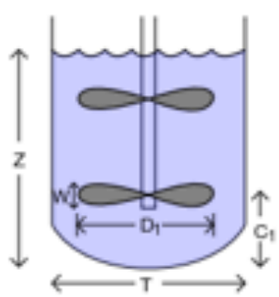
Undo Redo Solve Optimize Plot Debug Profile Edit Docs

Inputs

Baffle angle (deg)	5.00 deg
Baffle configuration	Custom
Number of baffles	4
Baffle offset (mm)	28.0 mm
Baffle width (mm)	85.0 mm
Dish type	flat
Retrofit?	
T (mm)	2000 mm
Variable speed drive?	Yes
Z (mm)	1500 mm
Equipment - Impeller 1	
C_1 (mm)	65.0 mm
D_1 (mm)	900 mm
1st blade angle (deg)	45.0 deg
1st blade count	6.00
Type of first impeller	pitched blade turbine

Outputs

Equipment	
Dish height (m)	0.000 m
1st std. motor percent (percent)	0.813
1st std. motor size (kilowatt)	0.500 kilowatt
2nd std. motor percent (percent)	0.542
2nd std. motor size (kilowatt)	0.750 kilowatt
S (mm)	535 mm
$V_{straight}$ (liters)	137 liters
V (liters)	4710 liters
Z/T	0.750
Equipment - Impeller 1	
C_1/D_1	0.0722
C_1/T	0.0325
D_1/T	0.450
W_1/D_1	0.133
W_1/T	0.0600



Alert Messages

Search alerts

Z/T is below the recommended lower limit.
CFD may be an appropriate tool

Details

Alert

Z/T is below the recommended lower limit.

Short, fat tanks require a larger torque machine, and hence increase capital cost. Adjust Z or T so that Z/T ≥ 0.90.

blndlv.pdf

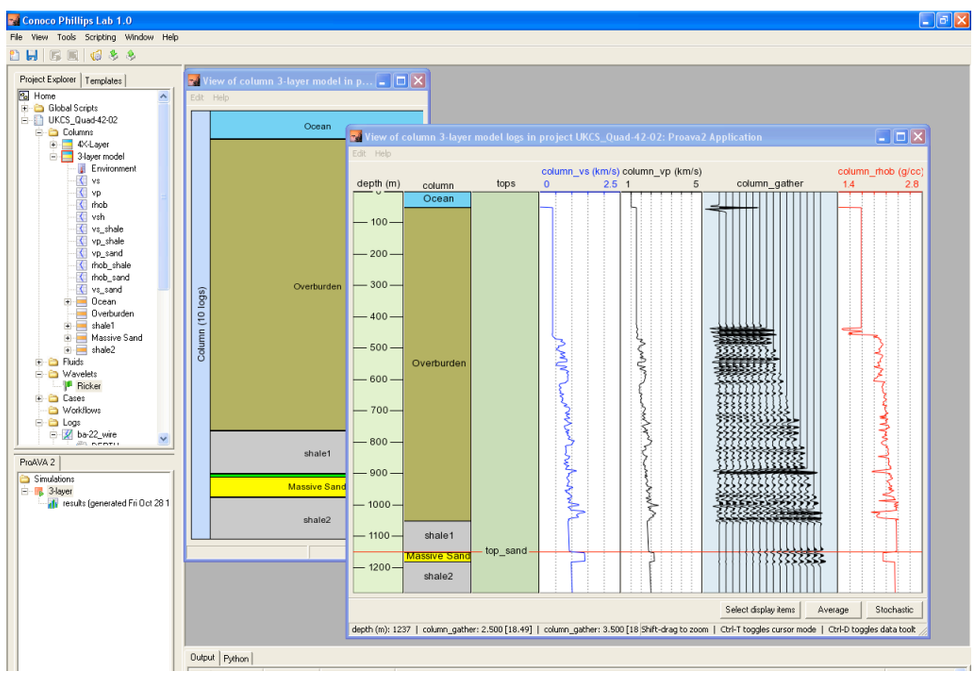
Open in a new window

Oil & Gas

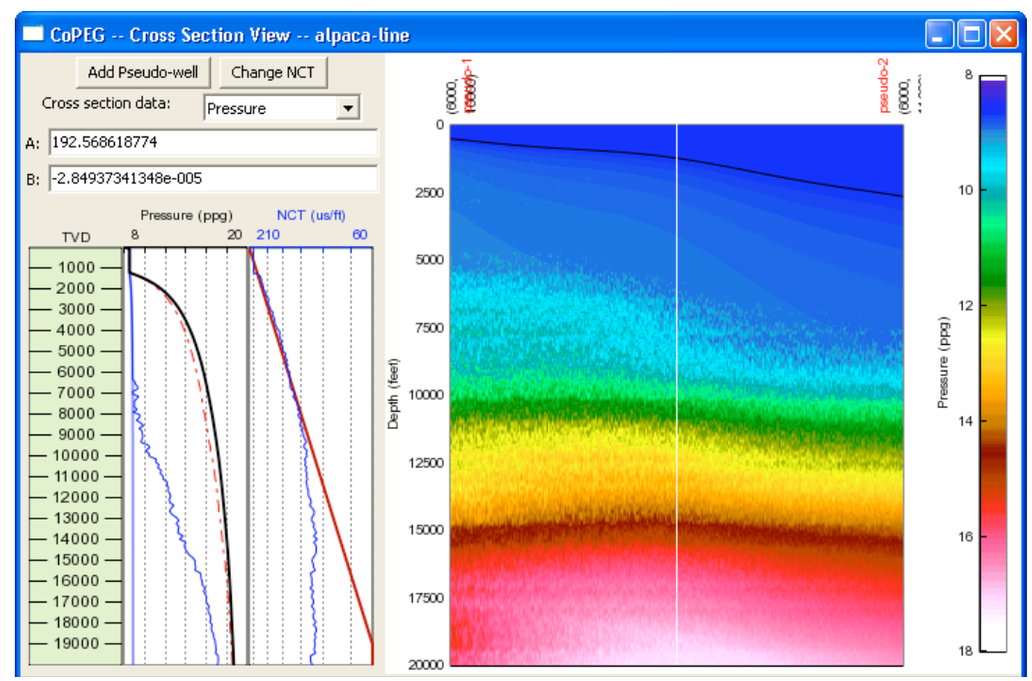
ConocoPhillips

Shell

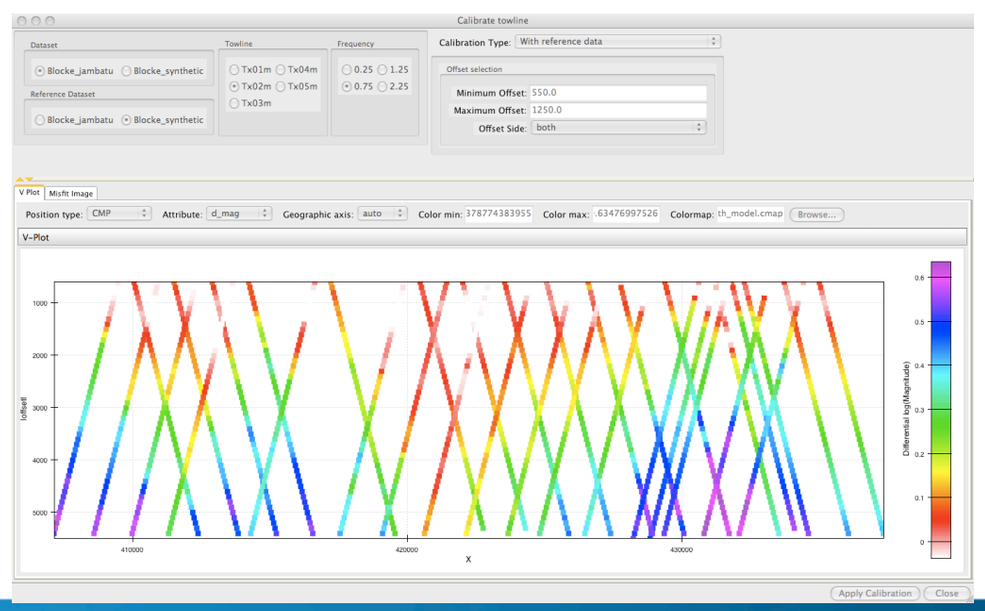
Stochastic Seismic Analysis



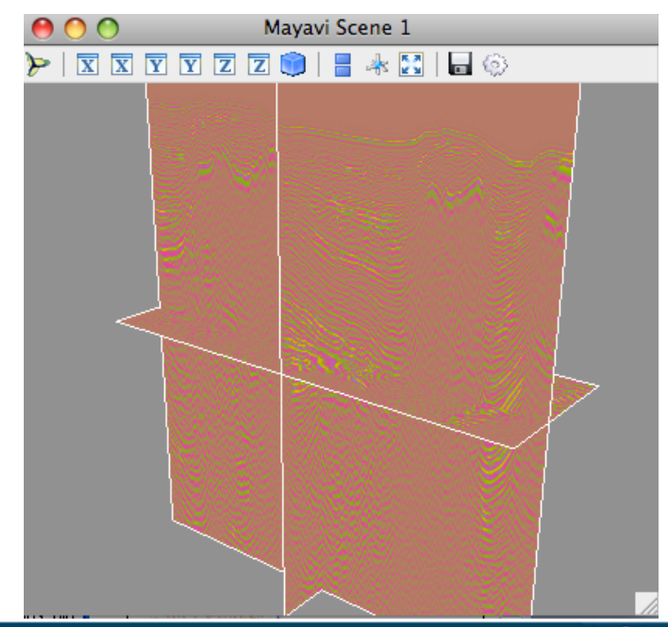
Pore Pressure Analysis



Electromagnetic Analysis

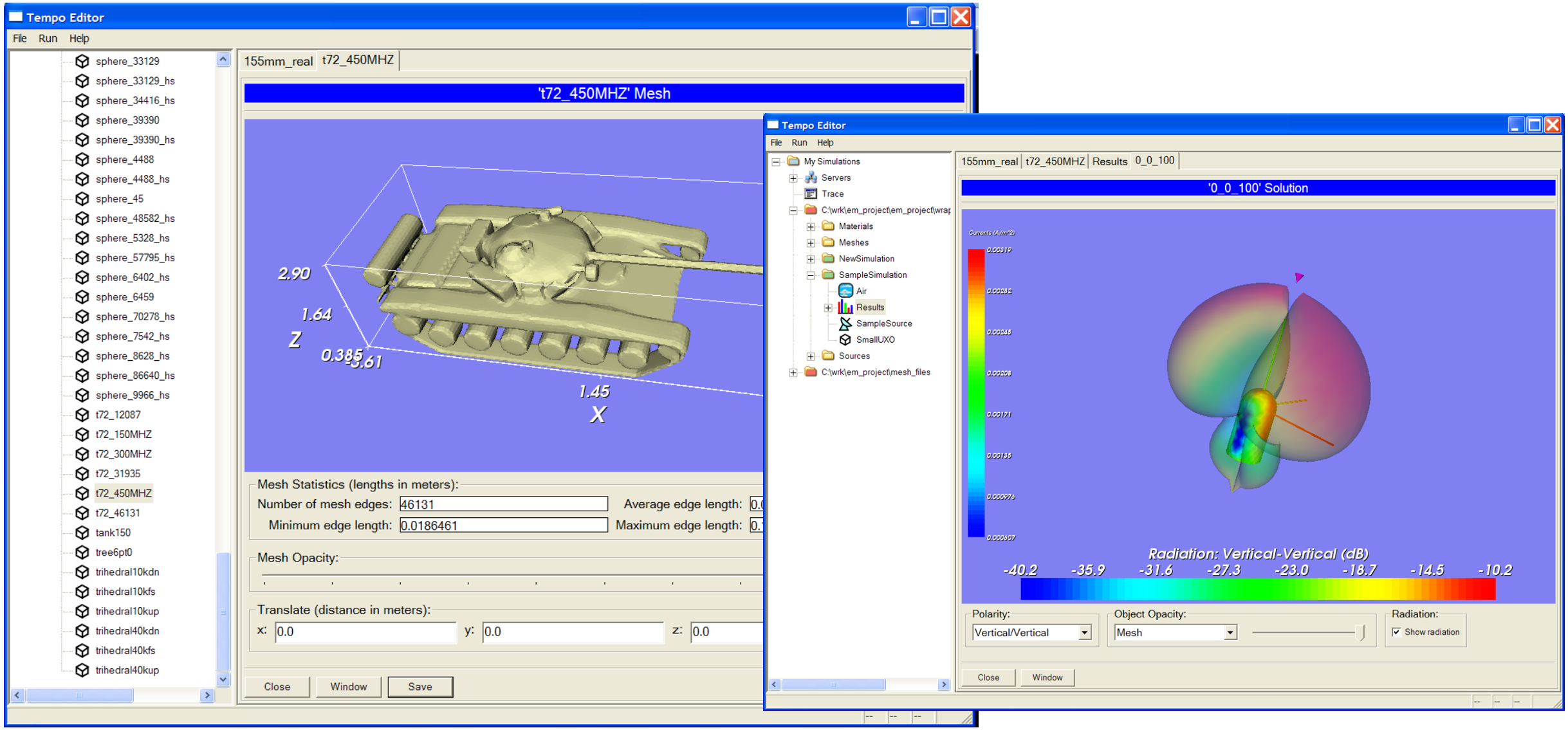


Seismic Visualization

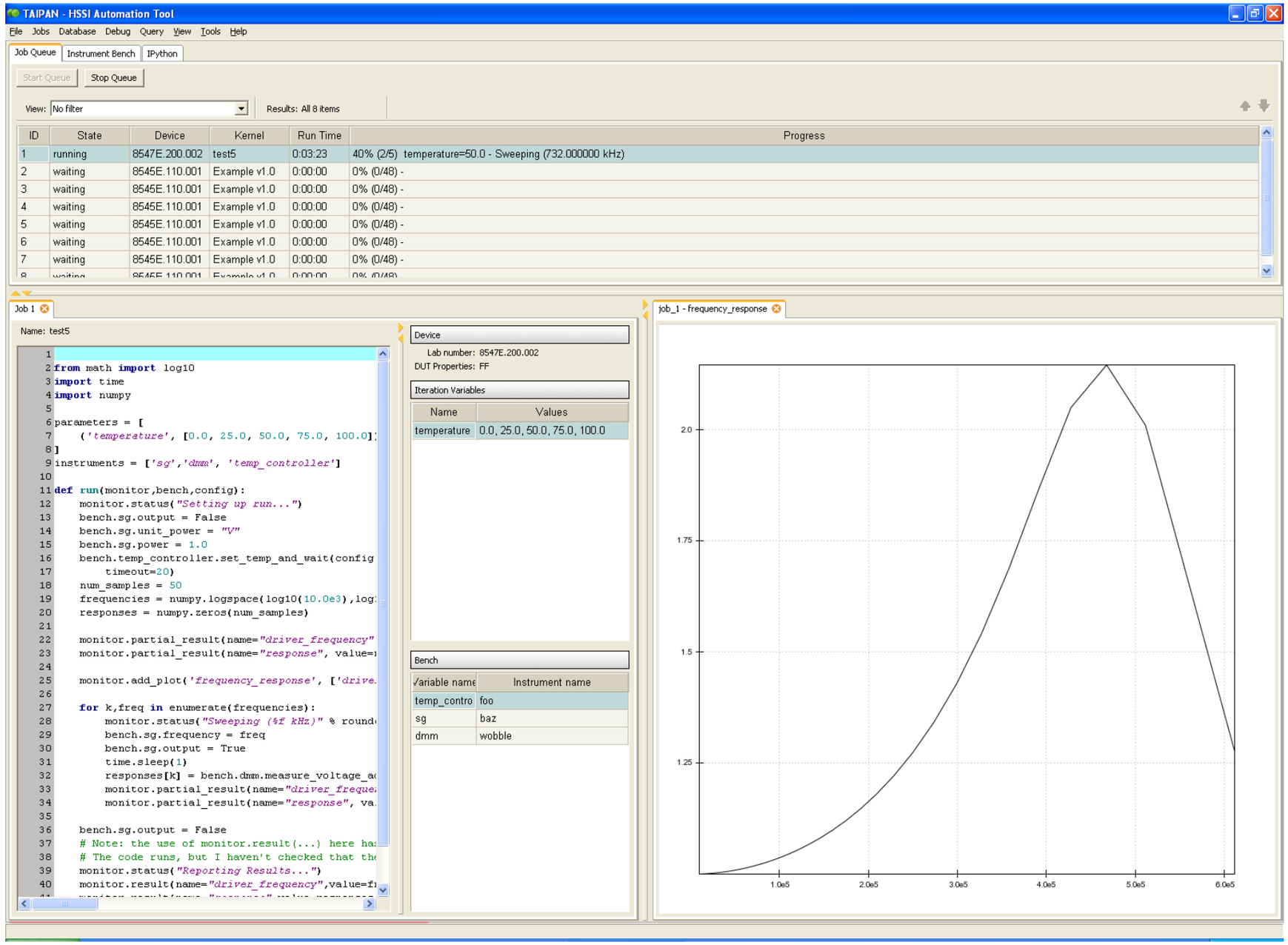
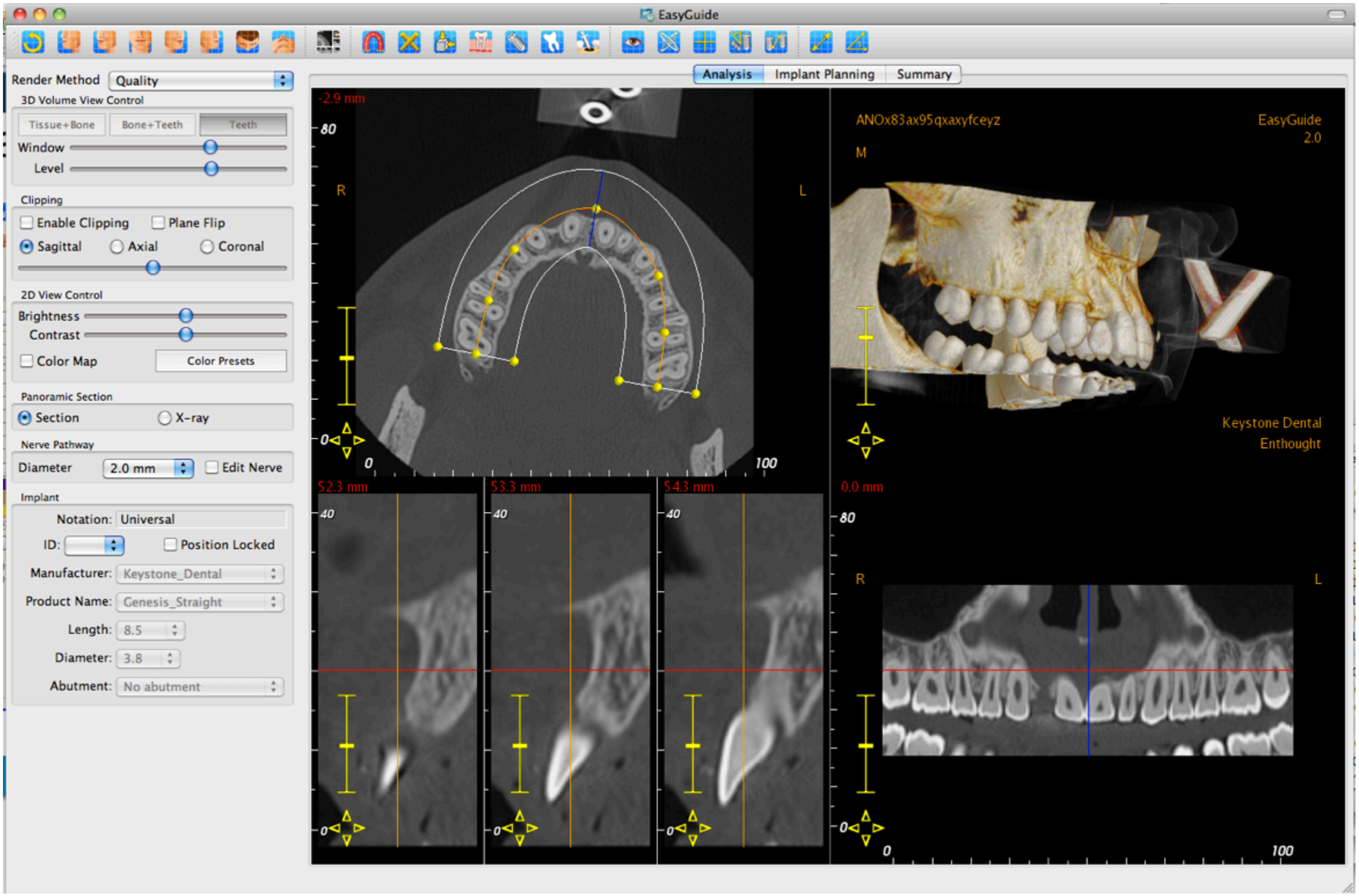


Army Research Labs (ARL)

- Electromagnetics Simulation
- Python/C/Fortan90
- Fine grained Parallelism (MPI based)
- Desktop Application
- Cluster Execution



High Speed Signal Integrity Testing



Dental Surgery Planning