VarPy: A python library for volcanic and rock physics data analysis

Rosa Filgueira
Andrew Bell
Malcolm Atkinson
Ian Main

University of Edinburgh, School of Informatics
University of Edinburgh, School of Geosciences
Introduction- EFFORT

Multi-disciplinary project: Geoscientists, Rock-Physicists, and Computer Scientists

• **EFFORT aims:**
  – To predict brittle failure of rock samples
  – To scale these predictions with geo-system complexity
  – To provide facility for moving an storing big data.
  – To provide facility for developing and testing codes
Rock Physics Laboratory experiments

Pressure Vessel

UCL- RP Laboratory

Rock Samples
Complex laboratory experiment:

Initial target: 30 months
Deploy under the sea- Mediterranean
8 rock samples- different features
Different interval of times and data sizes
EFFORT Science Gateway

• **Science gateway for rock physicists and volcanologists, which supports:**
  – sharing data from laboratories and observatories in real-time
  – sharing models and methods for analyzing data
  – supporting recurrent operational tasks
Volcanology and Rock Physics Communities

• Increase of digital instrumentation in volcanology and rock physics
  – Huge amount of seismicity data that need for computational analyses and models.

• Those communities they have recently started to share:
  – Data & Models

• Each researcher develops each own codes

• Not library designed specifically for volcanic earthquake and rock physics data
Proposal: VarPy

• New open-source python toolbox
• Aim: facilitate rapid application development for those communities
  – Focus in seismicity deformation data
  – Full repertoire of commonly required actions
    • Analysis and modeling in real time and retrospective
    • Capabilities for data exploration, data analysis, quality check
  – Users can define their own workflows to develop
    • models, analyses and visualizations
EFFORT and VarPy

• VarPy will help run the user’s models in the gateway automatically.
• Two variants of the VarPy with many of the functions identical: Gateway & Developers versions
Expected Benefits

• Easy method to analyze seismicity data
• Standardize different tasks/procedures
• Using the same functions by different researches → easier to compare results and performance of models
• Cost of maintaining the library is shared among the community
VarPy design

• Library style from **ObsPy:**
  – Open-source Python toolbox for **seismological data processing.**
  – Well documented, many user, grows progressively, users support

• VarPy does not attempt to replace the functions provided by other python libraries (NumPy, SciPy) → complementary of them

• VarPy & Ipython notebooks
VarPy datatypes

- **Seismicity deformation**: strain and stress data
- Time-series data of two classes
  - *Event catalogue data* (ECD)
    - Series of events (acoustic emissions, earthquakes, volcanic eruptions)
    - Occur at discrete times
    - Specific attributes (location, depth, magnitude, duration)
  - *Sampled continuous data* (SCD)
    - Series of times at which a continuous variable has been measured, and the value of that variable
    - Sample times are defined by the instruments’ operator
    - May (or may not) be evenly spaced (daily, every second).

- Volcanic observatories and rock physics laboratories can produce data of both classes in a single experiment.
VarPy datatypes

• Representation by using 4 different datatypes
  – Event catalogue laboratory data (ECLD)
  – Event catalogue volcanic earthquake data (ECVD)
  – Sampled continuous laboratory data (SCLD)
  – Sampled continuous volcanic earthquake data (SCVD)

• Volcanic data also have:
  – Eruption volcanic data (EVD):
    • Time-series data of volcanic eruptions, intrusions, or other events
    • with descriptions (type, size, duration,
VarPy Volcanic object

Volcanic (obj)

- ecvd (obj)
- evd (obj)
- scvd (Dict of obj)

- Models (Dictionary)
  - Mode name: Model (obj)
    - type
      - dataset
      - metadata
      - header
  - Model name: Model (obj)
  - Output (List)
    - Model Output (obj)
      - dataset
      - metadata
      - .
      - .
      - .
VarPy Laboratory object

- Laboratory (obj)
- eclD (obj)
- sclD (obj)
- type

Models (Dictionary)
- Model name: Model (obj)
- Mode name: Model (obj)
- type
- dataset
- metadata
- header

Output (List)
- Model Output (obj)
- Model Output (obj)
- Model Output (obj)
- dataset
- metadata
Example of VarPy objects

• Volcanic object with *ecvd* datatype:

```python
from varpy.management import core
ID = 'Tjornes_ex1'
ecvd_data_file = 'Iceland_IMO_C1_95-onwards.txt'
ecvd_metadata_file = 'Iceland_IMO_C1_meta.txt'
d1 = core.Volcanic(ID)
d1.add_datatype('ecvd',ecvd_data_file,ecvd_metadata_file)
```

• Laboratory object with *scld* datatype:

```python
from varpy.management import core
ID = 'UCL_Lab'
scld_data_file = 'UCL-exp1.txt'
scld_metadata_file = 'UCL-exp1-meta.txt'
d2 = core.Laboratory(ID)
d2.add_datatype('scld',scld_data_file,scld_metadata_file)
```
VarPy environment

• $d_1 = \text{core.Volcanic}(\text{ID})$ or
• $d_1 = \text{core.Laboratory}(\text{ID})$
• Creates a tree-directory in the current directory
## General Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>varpy.mangement</td>
<td>Core classes</td>
</tr>
<tr>
<td>varpy.data_preparation</td>
<td>Filtering routines</td>
</tr>
<tr>
<td>varpy.analysis</td>
<td>Analyzing filtered data routines</td>
</tr>
<tr>
<td>varpy.modeling</td>
<td>Modeling routines</td>
</tr>
<tr>
<td>varpy.simulation</td>
<td>Simulating routines of seismic data</td>
</tr>
<tr>
<td>varpy.statistics</td>
<td>Statistical routines for assisting with other routines</td>
</tr>
<tr>
<td>varpy.visualization</td>
<td>Plotting routines</td>
</tr>
<tr>
<td>varpy.write</td>
<td>Writing results routines</td>
</tr>
</tbody>
</table>
VarPy

management

core

conversion

data_feed

user_data_feed

data_preparation

data_conversion

window

analysis

magnitudes

<table>
<thead>
<tr>
<th>Module</th>
<th>Functionality: Module for</th>
</tr>
</thead>
<tbody>
<tr>
<td>core</td>
<td>Handling varpy objects and methods.</td>
</tr>
<tr>
<td>conversion</td>
<td>Converting dates and time into different formats.</td>
</tr>
<tr>
<td>data deed &amp; user_data_feed</td>
<td>Importing metadata and storing it into the varpy object</td>
</tr>
<tr>
<td>data_conversion</td>
<td>Converting values to another type of data (AE energy to magnitude)</td>
</tr>
<tr>
<td>window</td>
<td>Selecting a smaller sample based on a single (time window between two given dates) or on a combination of variables</td>
</tr>
<tr>
<td>magnitudes</td>
<td>Analyzing the filtered data (calculates completeness magnitude)</td>
</tr>
<tr>
<td>Sub-package</td>
<td>Functionality: Contains modules with</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>models</td>
<td>Various models which can be fitted to data.</td>
</tr>
<tr>
<td>model_application</td>
<td>Experiments: single analysis and multiple analysis</td>
</tr>
<tr>
<td>lab_data</td>
<td>Rock physics seismic data simulators</td>
</tr>
<tr>
<td>volcanic_data</td>
<td>Volcanic seismic data simulators</td>
</tr>
<tr>
<td>earthquake</td>
<td>Earthquake data simulators</td>
</tr>
</tbody>
</table>
**Package**  | **Functionality:** Contains modules for
--- | ---
statistics | Assisting with the generation of synthetic data, fitting models and comparing models
visualization | Plotting filtered data, the results of analyses, simulated data and model
write | Writing the results of analyses and experiments into text files
Varpy Experiments

• Experiments : Model + Filtered data + Parameters
• Experiment types:
  – Single Analysis ➔ Apply once a model
    • Retrospective Analysis:
      – Known failure/eruption time (ft)
      – Output: How well the model explains the data
    • Single Forecast
      – ft is not known
      – Output: Prediction of the failure/eruption.
  – Multiple Analysis ➔ Apply several time a model ➔ Data is streamed
    • Prospective Forecast
      – ft not known
      – Output: Prediction in real time of the failure/eruption.
    • Retrospective Forecast:
      – Simulate ft not known
      – Output: Prediction of the failure/eruption
Type of methods

- **M0 method**
  - Aim: to create a volcanic or laboratory object
  - Input: ID
  - Output: new object with and tree-directory.

- **M1 method**:
  - Aim: to add an attribute to an object
  - Input: data and metadata files
  - Output: Add the attribute to the object, and copy the files into the tree-directory

- **M2 method**
  - Aim: to modify an attribute (or several attributes) of an object.
  - Input: object
  - Return: new object (copy of the input object ) with the modifications.
    NO creates another tree-directory

- **M3 method**
  - Aim: to transform data.
  - Input: data
  - Output: data transformed

- **M4 method**
  - Aim: to write an attribute of an object to file.
  - Input: attribute of the object
  - Output: store a new file in the tree-directory

- **M5 method**
  - Aim: to plot into a figure an attribute of an object.
  - Input: attribute of the object
  - Output: store new figure the tree-directory
VarPy Jupyter Notebooks

• Notebook-1
  – Example of data exploration and visualization based on the Tjornes fracture zone (Iceland)

• Notebook-2
  – Example of applying forecasting methods using volcanic data from Mt. Etna
VarPy contributions

• VarPy already allows:
  – Data exploration
  – Quality check
  – Data analysis

• Researchers could contribute:
  – Developing standards for:
    • Data format
    • Methodology for processing data
  – New models, simulators of seismic data, filters ...
Dispel4Py

Dispel4Py is a library used to describe abstract workflows for distributed data-intensive applications

• Python language for describing tasks and connections
• PEs represent the basic computational blocks of any dispel4Py workflow: algorithm, service, data transformation
  – Shared: Storing them into the registry
  – Reusable: To be recombined in many different applications
• Graph: topology of the workflow
• Partitions, Groupings, Composite Pes
• Automatic mappings: MPI, STORM, Multiprocessing, Sequential
  – Scalability
Conclusions

• New Python open-source toolbox for volcanologists and rock physicists
• Study seismicity data
• Full repertory of common actions:
  – Analysis and Modelling
  – Real-time and retrospective-time
• Encourage collaboration between scientists
Questions

• Contacts:
  – Rosa Filgueira: rosa.filgueira@ed.ac.uk
  – Andrew Bell: a.bell@ed.ac.uk