SERPENT-TOOLS

A PYTHON PACKAGE FOR INTERACTING WITH SERPENT OUTPUTS

ANDREW JOHNSON

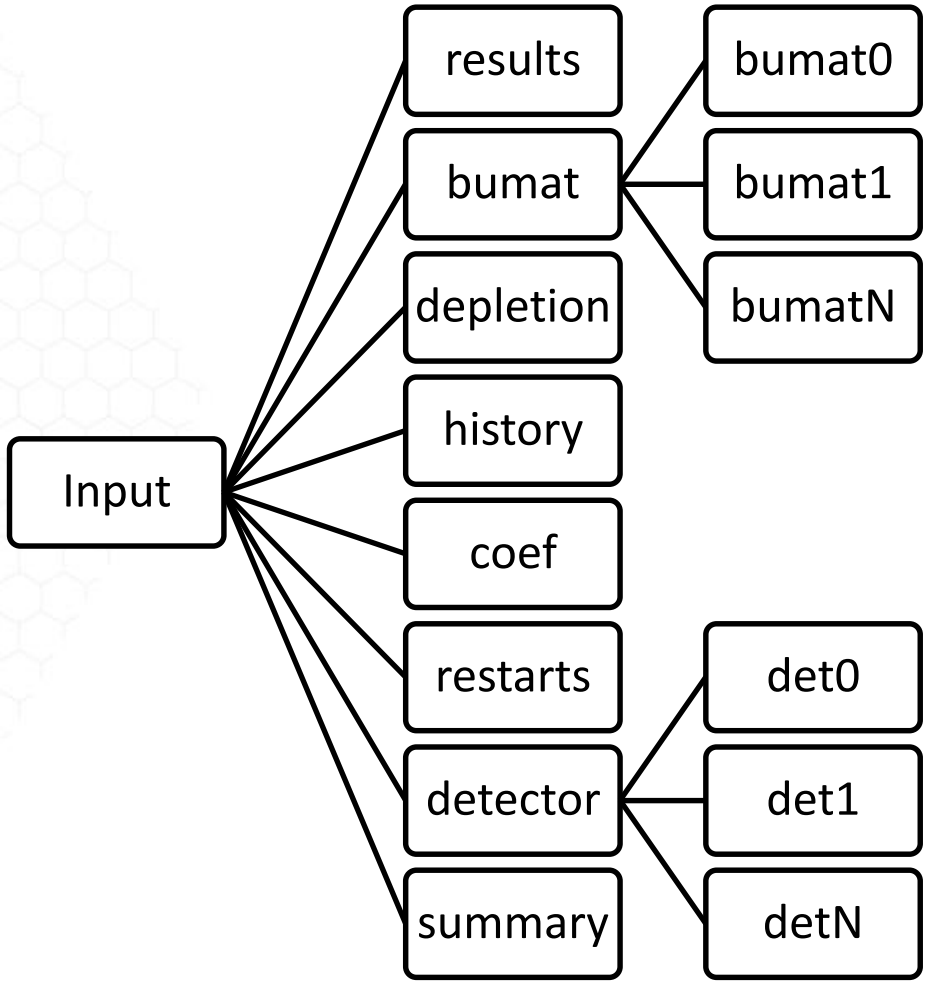
CREATING THE NEXT®

ORGANIZATION

1. Motivation
2. Requirements
3. Introduce the package
    1. Status
    2. Readers and Containers
4. Demonstration
5. Conclusion

MOTIVATION

- SERPENT produces a lot of data
  - ASCII/Binary
  - Matlab syntax/plain text
  - Some outputs are produced per time-step
- MATLAB
  - Files load automatically into workspace
  - MATLAB scripts require little/no conversion
  - Many built in functions that ease analysis
  - **Resource consumption for large files**



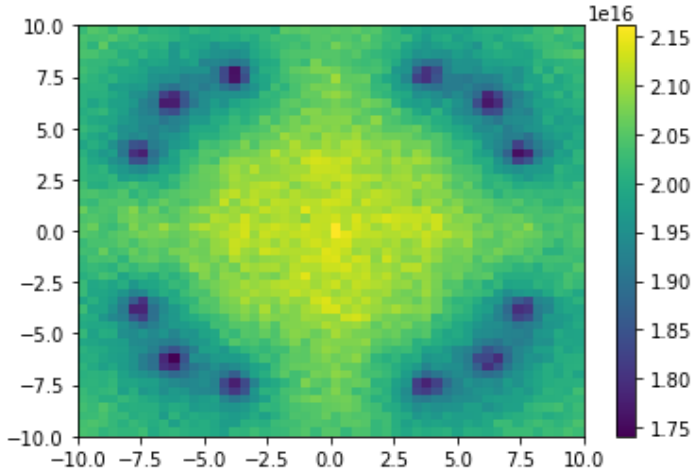
REQUIREMENTS

1. Maintain all data stored in outputs
2. Easy learning curve
3. Automate and simplify common analyses
4. Allow flexibility in user control
    1. Outputs for SERPENT 1 vs. 2
    2. Selectively accept/reject some data

SERPENT-TOOLS

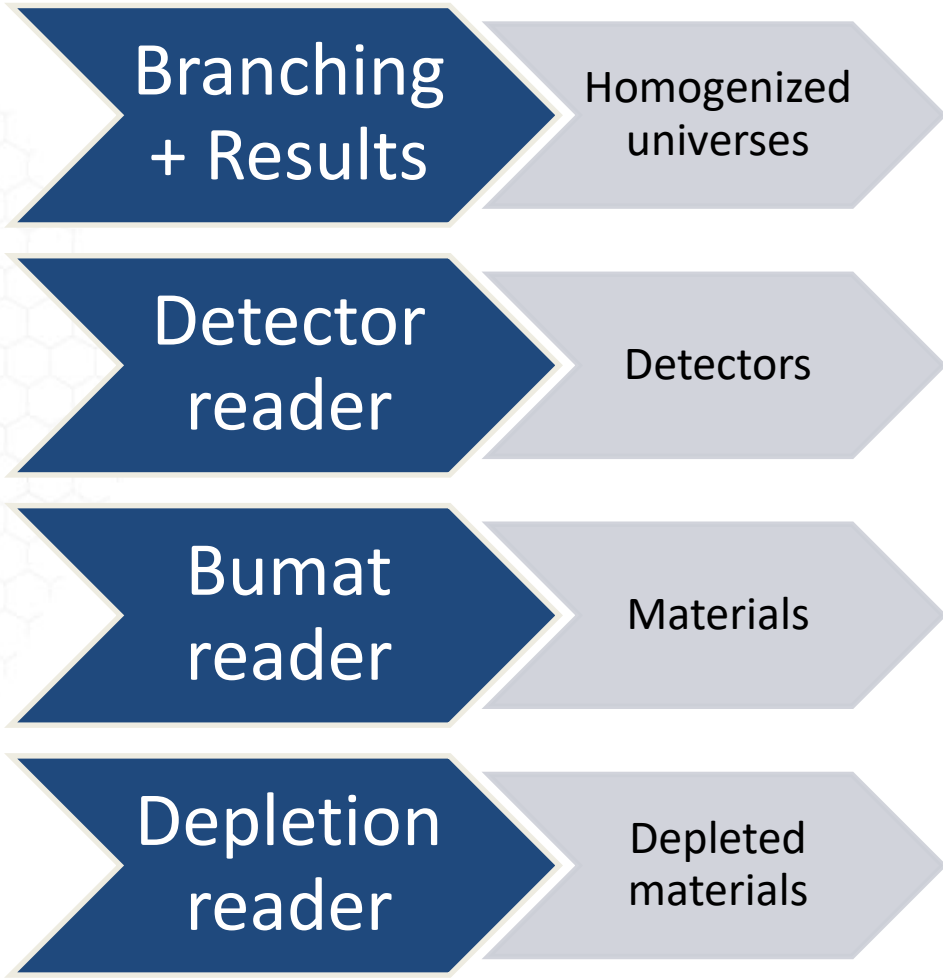
- Object-oriented python
  - Python 2.7 and 3.5+
- Suite of parsing tools and data structures
- High-level of control over processing
- Numpy and matplotlib for analysis and plotting
  - 80-95% shared functionality with MATLAB

```
In [1]: %matplotlib inline
        from serpentTools.parsers import DetectorReader

        INFO     : serpentTools: Using version 1.0b0+36.g4ad19c8

In [2]: det = DetectorReader('demo_det.m')

In [3]: det.read()

In [4]: det.detectors['xymesh'].meshPlot('X', 'Y');
```



READERS AND CONTAINERS

- Will support all SERPENT ASCII outputs
  - Unique reader classes for each file type
- Readers simply read the files
  - Produce custom data structures



Georgia Tech

Completed

- Depletion

In-progress

- Detector
- Branching
- Results
- History

To do

- Bumat
- Xsplot
- Fission matrix
- Sensitivities

Detector and branching readers are complete pending testing

DEMONSTRATION - SETTINGS

- Allow user control over what data to extract
  - Default to obtaining all data

```
In [2]: import serpentTools

        INFO     : serpentTools: Using version 1.0b0+36.g4ad19c8

In [3]: from serpentTools.settings import rc
        rc['verbosity'] = 'info'

In [4]: for k, v in rc.items():
            print(k,v)

        detector.names []
        depletion.metadataKeys ['ZAI', 'NAMES', 'DAYS', 'BU']
        branching.intVariables []
        branching.strVariables []
        xs.variableGroups []
        depletion.processTotal True
        serpentVersion 2.1.29
        branching.floatVariables []
        depletion.materials []
        verbosity info
        xs.variableExtras []
        depletion.materialVariables []
        detector.sigma 3
```

DEMONSTRATION – BRANCHING READER

- Creates `branchContainer` objects
  - Store group constants for each universe at that branch
- Eventually support exporting data to nodal diffusion formats
  - Generate branch coefficients

DEMONSTRATION – DEPLETION READER

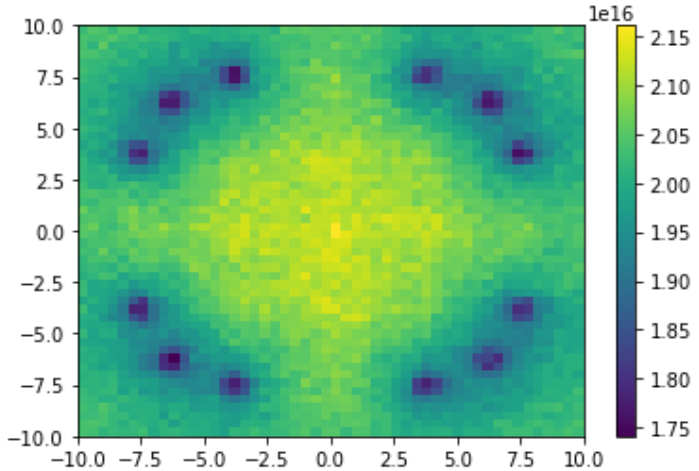
- Stores all data in `depletedMaterial` containers
- Control over
  - Which materials to process
  - What material data to process
- Powerful retrieval tool
- Simple plotting routines
- Easy to extend/subclass for follow-on analysis

```
In [6]: rc['depletion.materials'] = ['fuel']
        rc['depletion.metadataKeys'] = ['NAMES', 'DAYS', 'BU']
        rc['depletion.materialVariables'] = ['ADENS', 'MDENS', 'BURNUP', 'ING_TOX']
        rc['depletion.processTotal'] = True
```

DEMONSTRATION – DETECTOR READER

- Stores all data in `detector` containers
- Access to tally data and additional grids
  - Energy
  - Position
- Automatically convert uncertainties
  - Control over confidence interval
- Plot routines for common analysis
  - Spectrum
  - Meshes

```
In [1]: %matplotlib inline
        from serpentTools.parsers import DetectorReader

        INFO    : serpentTools: Using version 1.0b0+36.g4ad19c8

In [2]: det = DetectorReader('demo_det.m')

In [3]: det.read()

In [4]: det.detectors['xymesh'].meshPlot('X', 'Y');
```



CONCLUSION

- Demonstrated support for a few output files
- All output files to be supported in coming months
- More effective data retrieval

Future Work

- Automate/simplify procedure for preparing branching cross sections
- Compress output from repeated runs to object
  - Calculate actual uncertainties
- Straight-forward integration into external programs
- Output data to alternative file types
  - hd5, plain text, LaTeX, etc.

THANK YOU!

Georgia Tech

CREATING THE NEXT®