Computational Tools: Atomistic Modeling of Molecular Structures

- CCP-SAS provides the infrastructure and tools for analyzing small-angle scattering (SAS) data from complex systems using advanced modeling techniques
- Many advanced modeling techniques are not accessible to experimental researchers
- Often valuable tools are developed but become unusable when the developer leaves the research group

SASSIE

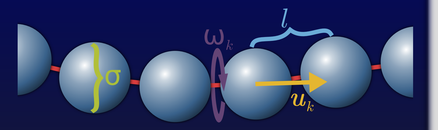
- The CCP-SAS codebase
- Consolidated and sustainable code
- Easily integrate new modules
- Complete workflow for advanced SAS data analysis

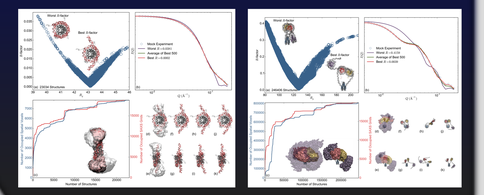
Monte Carlo Module for Simulating B-DNA

- Metropolis Monte Carlo sampling of B-DNA as a worm-like chain
- Rapidly generate ensembles of atomistic structures

$$U_{\text{bend}} = k_B T \frac{L_p}{l} \sum (1 - \mathbf{u}_k \cdot \mathbf{u}_{k+1})$$

$$U_{\text{twist}} = \frac{k_B T}{2} \kappa \sum_{k=1}^{N-1} (\omega_k - \bar{\omega})^2$$



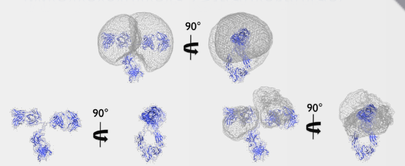


Molecular Structure

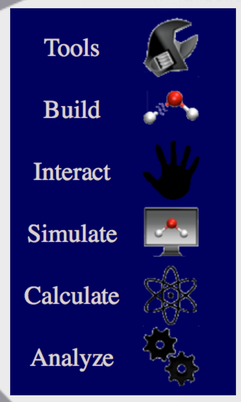
Determines

Function

SAS Probes Molecular Structure



Complete SAS Workflow



Contrast Calculator Align Data Interpolation	Extract Utilities Merge Utilities
PDB Scan	PDB Rx (alpha)
Molecular Viewer	
Monomer Monte Carlo Complex Monte Carlo Energy Minimization Docking (alpha)	Torsion Angle Monte Carlo (beta) Torsion Angle Molecular Dynamics Two-Body Grid
SasCalc (beta) Capriqorn (alpha) SCT Calculate SCT Optimize	SLD MOL EM to SANS Crysol
Chi-Square Filter SCT Analyze	Density Plot APBS

SASSIE

$E=mc^2$ Demo Application:

./bin/energy.py

```
#!/usr/bin/python
import json
import sys
import StringIO

if __name__=='__main__':

    # read input from JSON
    argv_io = StringIO.StringIO(sys.argv[1])
    json_variables = json.load(argv_io)

    # convert input strings to intended type
    mass = float(json_variables['m'])
    speed_of_light = float(json_variables['c'])

    # do science
    energy = mass * speed_of_light**2

    # format output as JSON
    output = {}
    output['e'] = energy
    print json.dumps(output)
```

Python: Accessible, Unified, and Sustainable Codebase

- Integrate/wrap existing code
- Access lower level languages: Fortran, C/C++, CUDA
- Low barrier to entry
- Rapidly develop, test, and debug new modules
- Scalability: Personal to HPC resources

GenApp Framework

- Simplify broad deployment
- Lower the entry barrier for deploying new algorithms
- Community governance
- Seamlessly generate GUI applications for a variety of hardware systems, including science gateways
- Preserve scientific code in an evolving software landscape
- Open code without onerous licensing

Using Python to Create
An Integrated Modular Framework for Atomistic Modeling of Molecular Structures

National Institute of Standards and Technology, U.S. Department of Commerce
UT Medicine Health Science Center San Antonio
CCP-SAS

Steven C. Howell, Emre Brookes, Joseph E. Curtis

For a list of CCP-SAS publications visit: http://ccpsas.org/impact.html

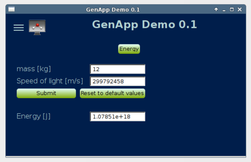
This work benefitted from CCP-SAS software developed through a joint EPSRC (EP/K039121/1) and NSF (CHE-1265821) grant.

Presented at the SciPy 2016, 13-15 July 2016

Run GenApp Compiler

```
> ./genapp.pl
```

Qt GUI Version



Java GUI Version

To view live demo visit: http://somo.chem.utk.edu/demo

Web Version

$E=mc^2$ GUI Definition:

./modules/energy.json

```
{
    "moduleid"   : "energy",
    "label"      : "Energy",
    "executable" : "energy.py",
    "fields"     : [
        {
            "role"    : "input",
            "id"      : "m",
            "label"   : "mass [kg]",
            "type"    : "float"
        },
        {
            "role"    : "input",
            "id"      : "c",
            "label"   : "Speed of light [m/s]",
            "type"    : "float",
            "default" : 299792458
        },
        {
            "role"    : "output",
            "id"      : "e",
            "label"   : "Energy [J]",
            "type"    : "text",
        }
    ]
}
```

Web-based Features

- User registration / login / verification
- User file system (download results)
- User job management (reattach job w/ inputs & outputs)
- Feedback (with job specific reporting option)
- Module documentation link on each page
- Simple configuration for HPC resources
  - Airavata
  - XSEDE (Indiana University)
  - JetStream
  - SCARF (UK)
  - Titan (ORNL)

Rapid GUI Development

1. Modify scientific module to accept and produce JSON input and output

2. Write a JSON definition file describing the i/o requirements of the module

3. Compile script interprets JSON definition file to build standalone and/or web-based user interface

- Qt3/C++ and Qt4/C++
- HTML5/PHP
- Android
- iOS
- Java