The SDE: A General Computational Chemistry Software Framework

Ryan M. Richard
Windus Group

Ames Laboratory, Ames, IA

March 27, 2019
Computational Chemistry

- Predict and explain chemistry via numeric models

![Diagram showing energy levels and intensity]

- High-accuracy, but at high-cost
  - Approximations and/or HPC required for larger systems

- Plethora of packages/libraries

- Angle: 109.5°
Computational Chemistry

- Predict and explain chemistry via numeric models
- High-accuracy, but at high-cost

![Graph showing the relationship between system size and time to solution for different methods like CCSD(T), MP2, SCF, and Quicksort.](image)
Computational Chemistry

- Predict and explain chemistry via numeric models
- High-accuracy, but at high-cost
- Approximations and/or HPC required for larger systems
Computational Chemistry

- Predict and explain chemistry via numeric models
- High-accuracy, but at high-cost
- Approximations and/or HPC required for larger systems
- Plethora of packages/libraries
Overview

- Easily refactorable
  - Novel hardware
  - Reuse ecosystem
- Study new properties
Overview

- Easily refactorable
  - Novel hardware
  - Reuse ecosystem
- Study new properties
- Contribute back to ecosystem
Overview

- Easily refactorable
  - Novel hardware
  - Reuse ecosystem
- Study new properties
- Contribute back to ecosystem
- Rapid prototype
Input Layer

- relies on C++17
- Possible modules discovered at runtime
- Set a module’s parameters directly
- Coupling is to properties
Writing a Module

Inputs and results type-erased
Automatic domain checking for inputs
”sandboxes” for developers
Call other modules through ”property types”

1) Obtain guess

```cpp
auto guess = run_submodule<GuessDensity>("CoreGuess", mol);
```

2) Build core Hamiltonian

```cpp
auto h = run_submodule<CoreHamiltonian>("CoreHamiltonian", mol);
```

3) Define molecular system

```cpp
bool conv = false;
double E = 0.0;
do {
    auto F = run_submodule<FockBuild>("FockMatrix", mol, guess);
    auto new_guess = run_submodule<UpdateDensity>("UpdateGuess", mol, guess);
    auto H("mu", "nu") = h("mu", "nu") + F("mu", "nu");
    auto newE = new_guess("mu", "nu") * H("mu", "nu");
    conv = check_convergence(new_guess, guess, newE, E);
    E = newE;
    guess = new_guess;
} while(!conv);
return E;
```
Cacheing and Checkpointing

- SDE records module calls
- Scientific record
- Checkpointing
  - Module calls are memoized
  - Repeated calls return cached result
  - Includes rerunning of calculation

```
run_as(Args...args)

1) Compute Hash
   auto h = memoize(args...);

2) Check Cache
   if(cache.count(h))
      return cache.at(h);

3) Compute Value
   auto rv = run(args...);

4) Cache and return
   cache.insert(h, rv);
   return cache.at(h);
```
Summary and Outlook

- SDE is the software framework of NWChemEx
- Leverage for interoperability
- Open source and available on GitHub (will be once licensing is worked out...)

RMR (Ames Lab)
Acknowledgments

- NWChemEx team
- Windus group
- Funding
- Organizers
- You (the audience)