The SDE: A General Computational Chemistry Software Framework

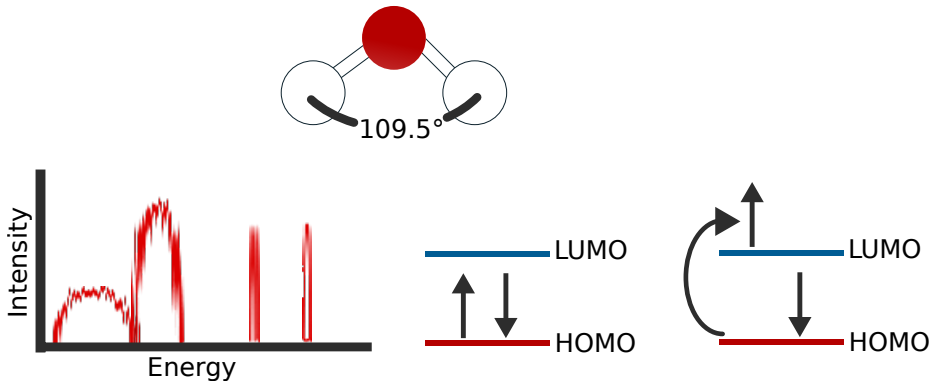
Ryan M. Richard
Windus Group

Ames Laboratory, Ames, IA

March 27, 2019

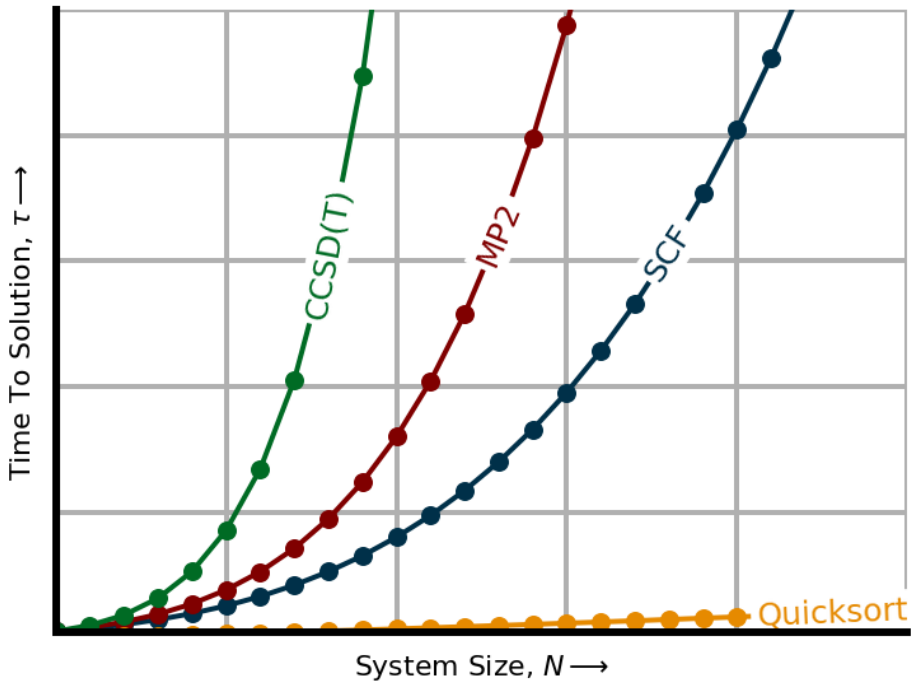
Computational Chemistry

- Predict and explain chemistry via numeric models



Computational Chemistry

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



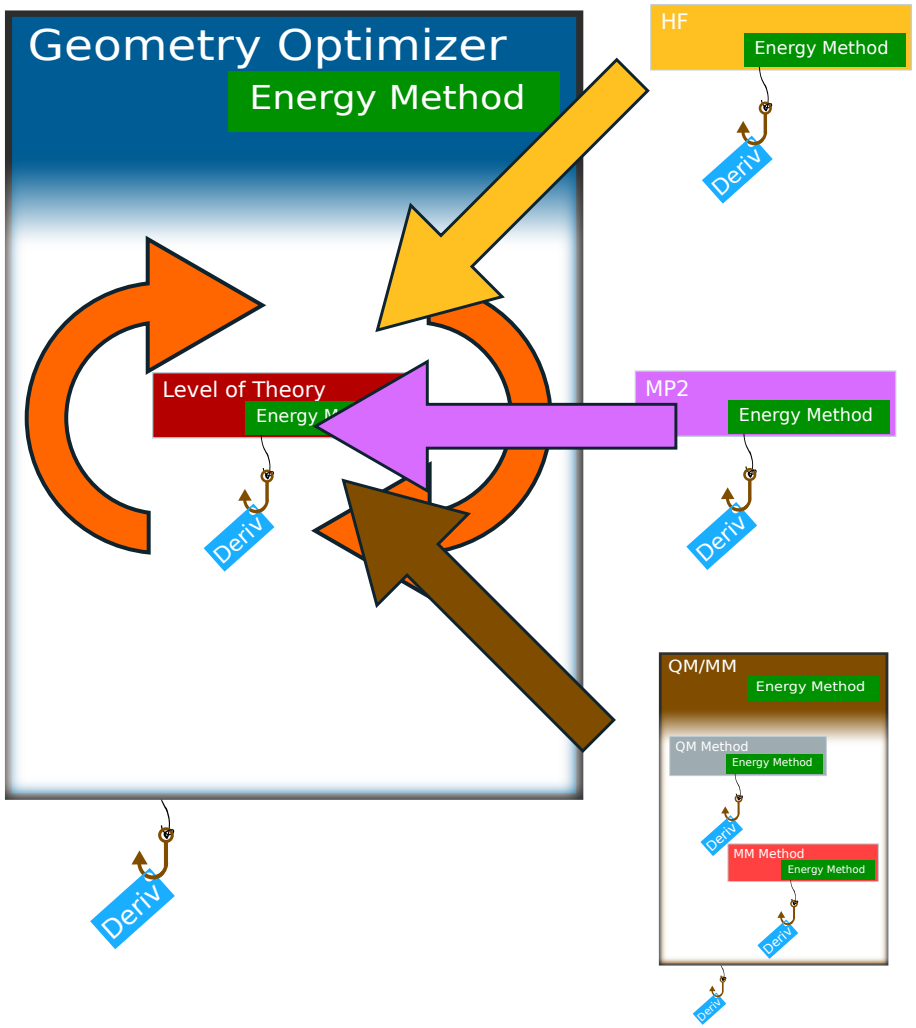
Computational Chemistry

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

Computational Chemistry

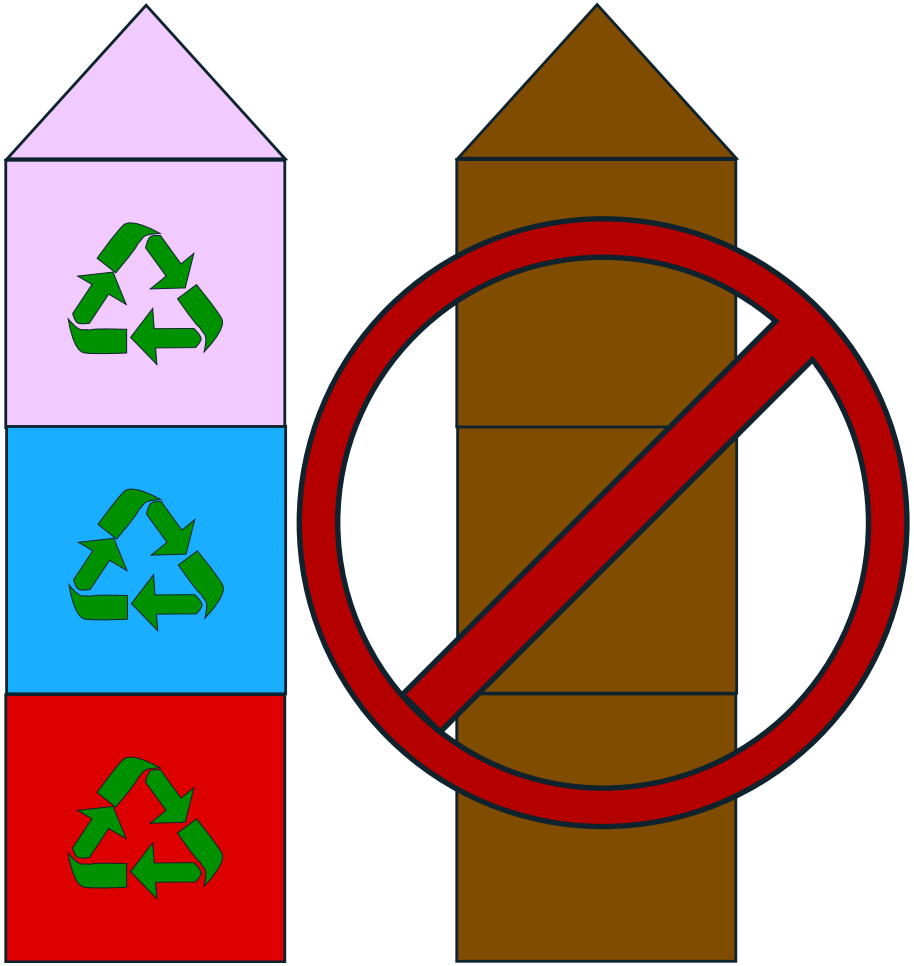
- Predict and explain chemistry via numeric models
- High-accuracy, but at high-cost
- Approximations and/or HPC required for larger systems
- Plethora of packages/libraries

Overview



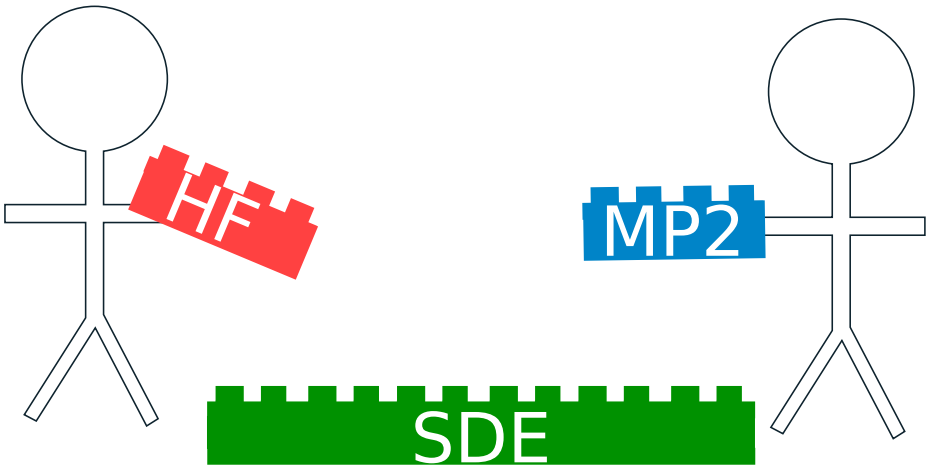
- Easily refactorable
  - Novel hardware
  - reuse ecosystem
- Study new properties

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

Input Layer

1) Load modules

```
ModuleManager mm();
mm.insert("SCF", SCFLoader());
mm.insert("CoreGuess", CoreGuess());
...
```

2) Set parameters

```
auto mod = mm.at("SCF");
mod.change_parameter("Energy Conv.",
                     10E-6);
...
```

3) Define molecular system

```
auto mol = make_molecule("
                    H 0.0 0.0 0.0
                    H 0.0 0.0 0.9
                    ");
mol = apply_basis("aug-cc-pVDZ", mol);
```

4) Compute properties of interest

```
mod = mm.at("SCF");
auto egy = mod.run_as<Energy>(mol);
```

Writing a Module

SCF::run(Molecule mol)

1) Obtain guess

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

2) Build core Hamiltonian

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

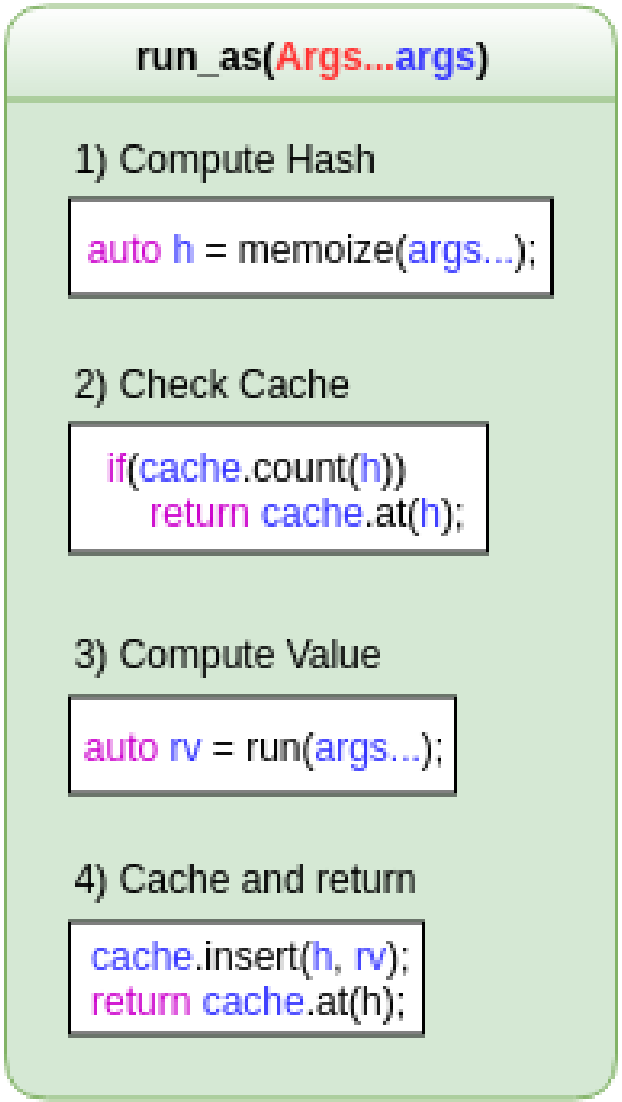
3) Define molecular system

```
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_convergance(new_guess, guess, newE, E);
   E = newE;
   guess = new_guess;
} while(!conv);
return E;
```

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

Cacheing and Checkpointing

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



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...)

Acknowledgments

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