Managing Complexity: Lessons from Growing and Maintaining the Psi4 Ecosystem

LORI A. BURNS
SHERRILL GROUP, GEORGIA TECH, USA
REUSEABLE LIBRARIES FOR QUANTUM CHEMISTRY, MAJVIK, FINLAND

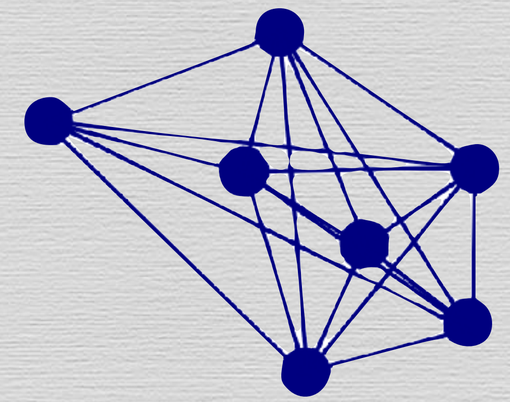
1 JULY 2025

MolSSI

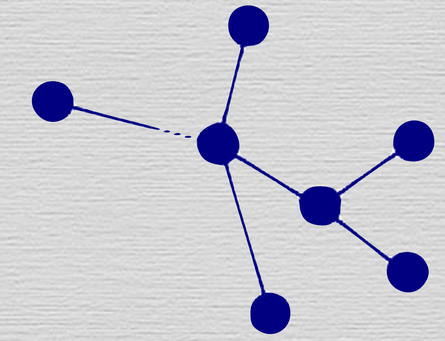
REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

STRONG COUPLING



LOOSE COUPLING



- easier to understand
- easier to compose

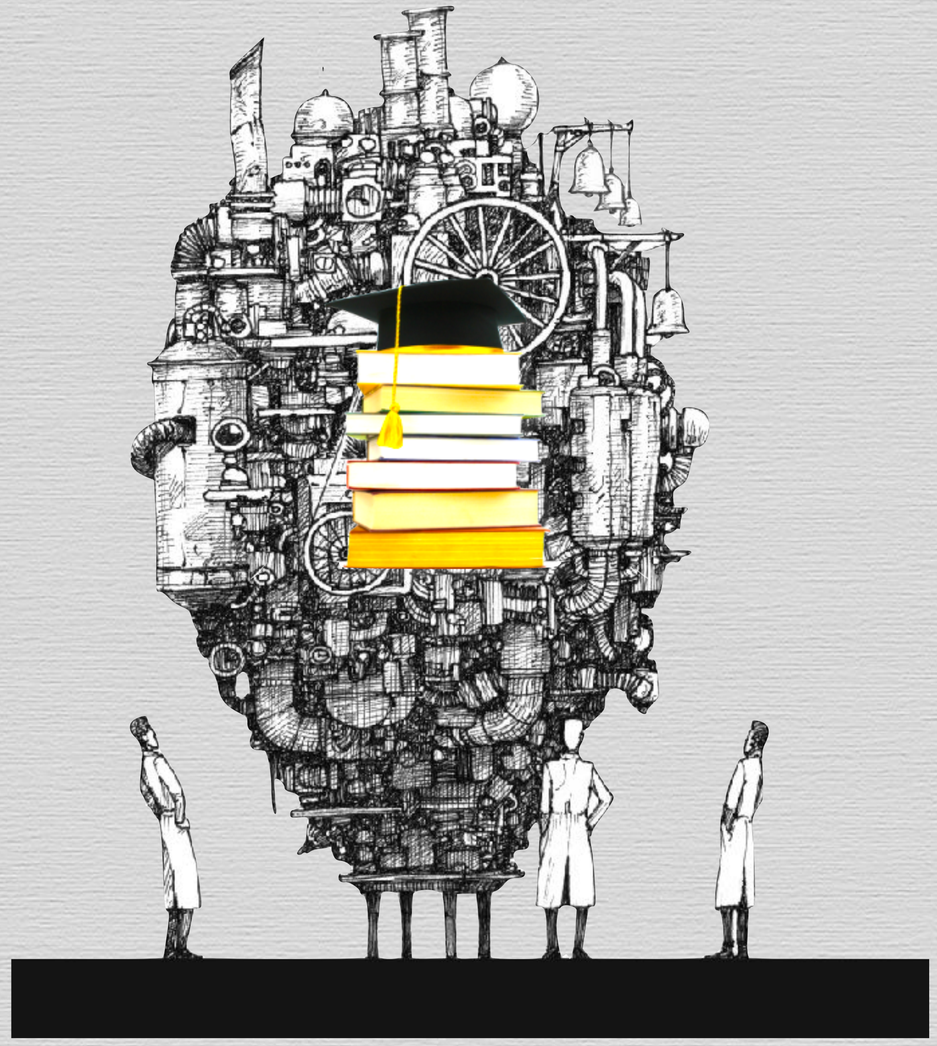
LOW COHESION

- difficult to maintain, test, read
- Swiss army knife modules

HIGH COHESION

- robust, reuseable, understandable
- stable APIs
- do one thing only and do it well (e.g., Unix command line)

QC PROGRAMS

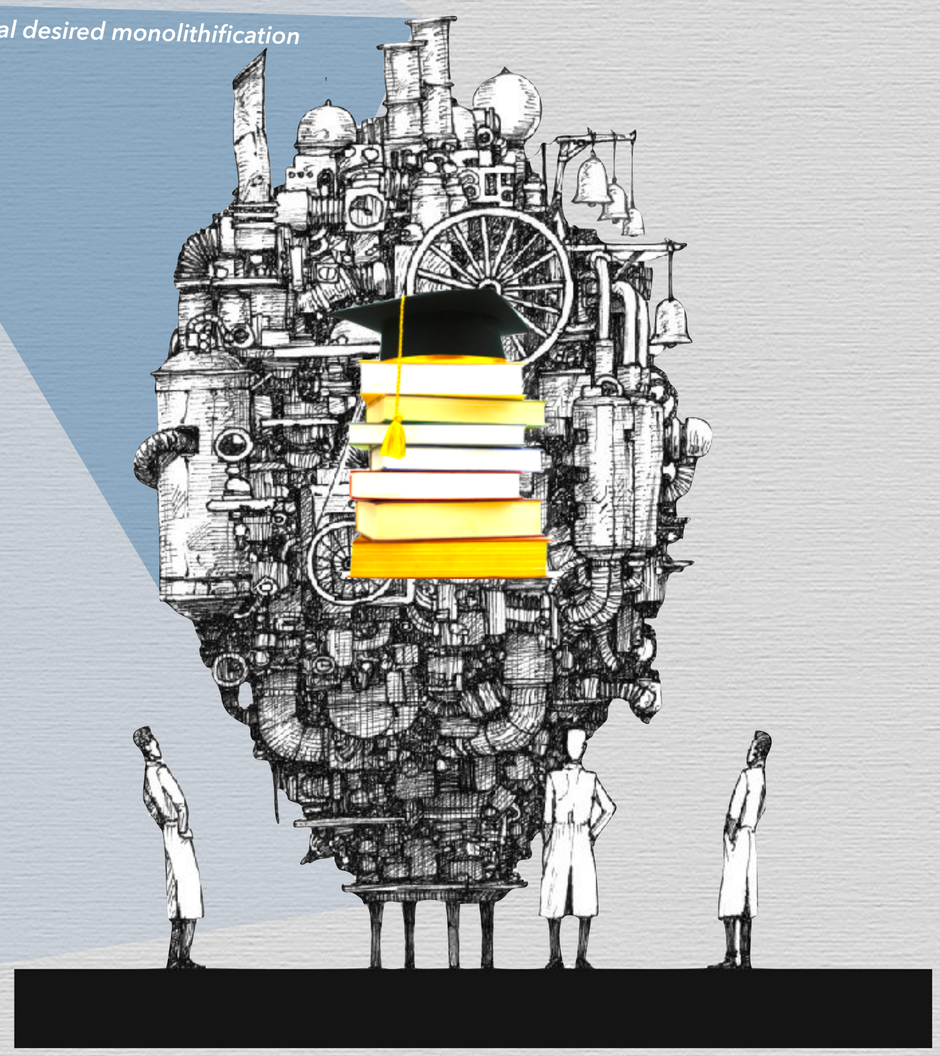


Slide LHS taken from Managing Complexity and Modular Code Development by Radovan Bast

QC PROGRAMS

maximal observed monolithification

minimal desired monolithification



drivers of whole calcs
geometry optimization/IRC • MD driver
crystal lattice energies • torsion driver

OPTKING CUBY PYVPT2
CRYSTALATTE MDI BIGCHEM

processors of whole calcs
finite diff. deriv. • vibrational analysis
many-body decomposition

FRAGMENT
FINDIF QCMANYBODY
BASISOPT

wfn processors, Molden file comm.
orbital analysis • multipole analysis
natural population analysis

MULTIWFN
GDMA CCLIB

user interface

workhorses
SCF methods • post-SCF methods
method support libraries

QUTREE COMPOSITEJK
HORTON DFTB+
FORTE AMBER

separable method components
dispersion corr. • basis set effects

DFTD3 BLOCK2
LIBMBD

post-SCF, SCF wfn comm.
DMRG/multiref. PT2 • arb.-order CC/CI
active-space CC & CR-CC • ADC mtds
GW • linear-response TDDFT

MRCC
ADCC
CHEMPS2

SCF-time (part of wfn), API comm.
eff. core (ECP) & frag. (EFP) potentials
relativistic contrib. • solvation corr.
polarizable embedding

LIBEFP
DKH
CPPE PCMSOLVER

tools
DFT quadrature grids • orbital opt.
symmetry analysis • k-point grid gen.

OPENORBITALOPTIMIZER
MOLSYM

resources, API or file copy/paste
physical constants • periodic table
basis sets • norm-conserving PP

QCELEMENTAL

intensive libs, fast API comm.
$2e^-$ & property integrals
density functionals • grid evaluations
tensor manipulations

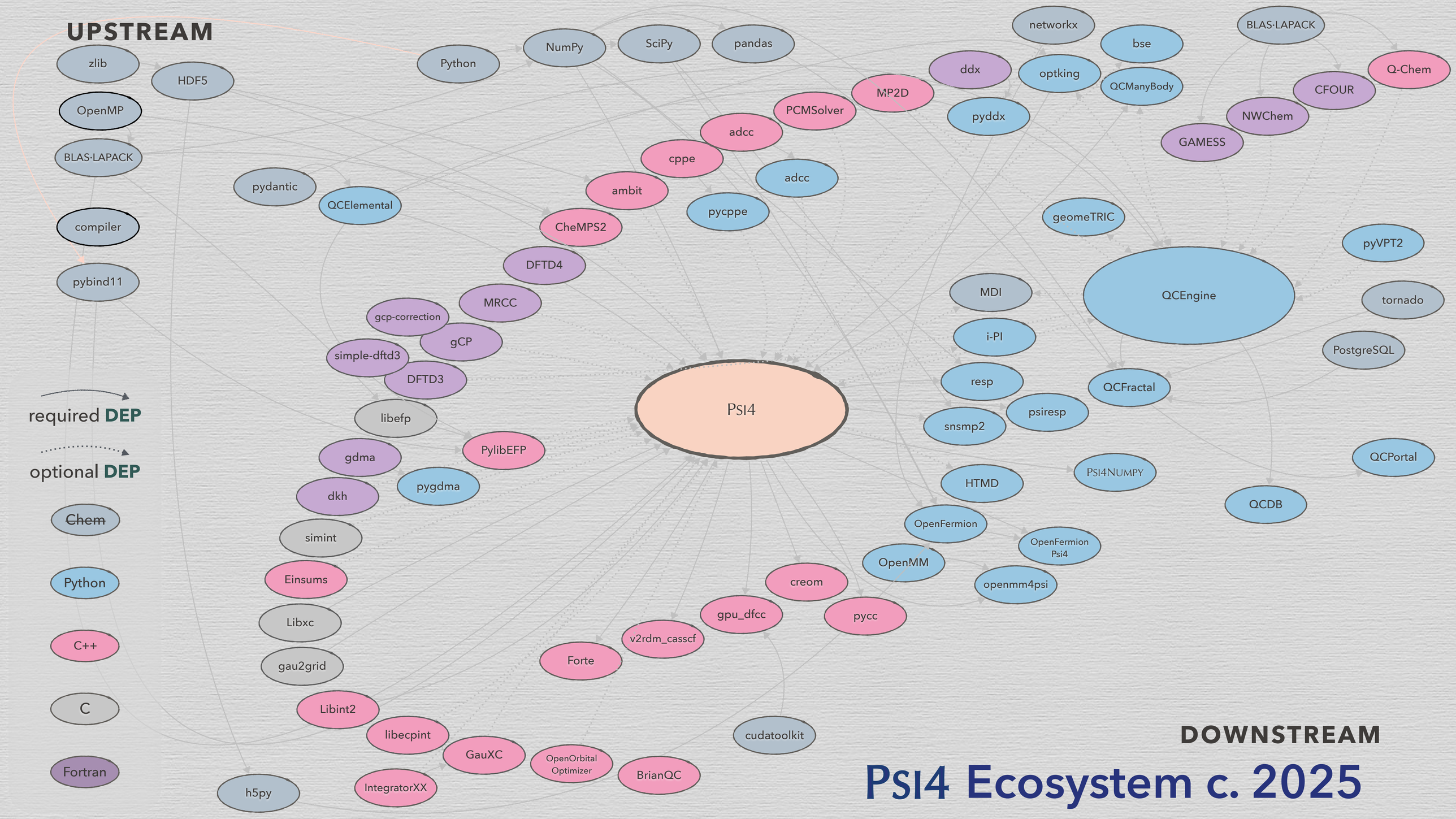
LIBXC LIBINTX LIBCINT
GAU2GRID LIBERI

non-chem. intensive libs, fast API
linear algebra • binary file format
parallelism frameworks • FFT

BML
LAPACK

software engineering tools
compilers • build systems
version control • testing frameworks
continuous integration services

figure from Electronic Structure (2024) chapter
Blum, Windus, Burns, Oliveira, Pritchard, Slipchenko, et al.


UPSTREAM
zlib
HDF5
OpenMP
BLAS·LAPACK
compiler
pybind11
Python
NumPy
SciPy
pandas
pydantic
QCElemental
networkx
bse
BLAS·LAPACK
ddx
optking
Q-Chem
MP2D
QCManyBody
CFOUR
PCMSolver
pyddx
NWChem
adcc
GAMESS
cppe
adcc
ambit
pycppe
geomeTRIC
CheMPS2
pyVPT2
DFTD4
MDI
QCEngine
tornado
MRCC
gcp-correction
i-PI
gCP
simple-dftd3
PostgreSQL
DFTD3
resp
QCFractal
libefp
psiresp
Psi4
snsmp2
PylibEFP
QCPortal
gdma
Psi4Numpy
pygdma
HTMD
dkh
QCDB
OpenFermion
simint
OpenFermion
Psi4
OpenMM
Einsums
creom
openmm4psi
Libxc
gpu_dfcc
pycc
v2rdm_casscf
gau2grid
Forte
Libint2
cudatoolkit
libecpint
GauXC
OpenOrbital
Optimizer
BrianQC
IntegratorXX
h5py
required DEP
optional DEP
Chem
Python
C++
C
Fortran
DOWNSTREAM
Psi4 Ecosystem c. 2025

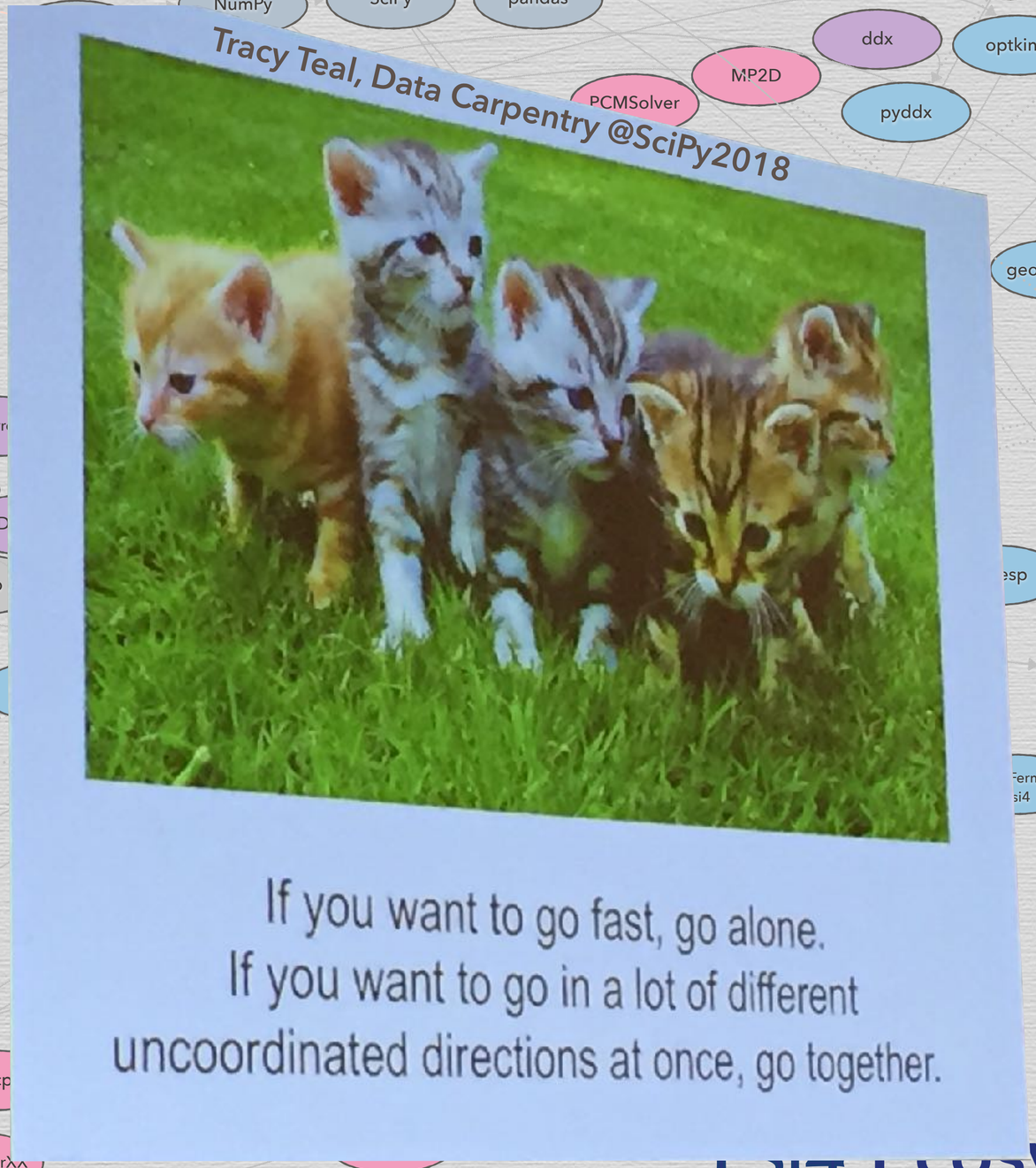
UPSTREAM

zlib
HDF5
OpenMP
BLAS·LAPACK
compiler
pybind11
NumPy
SciPy
pandas
pydantic
QCElemental
networkx
bse
ddx
optking
QCManyBody
MP2D
PCMSolver
pyddx
BLAS·LAPACK
Q-Chem
CFOUR
NWChem
GAMESS
geomeTRIC
pyVPT2
QCEngine
tornado
PostgreSQL
QCFractal
QCPortal
PSI4NUMPY
QCDB
gcp-corr
simple-dftd3
libefp
gdma
dkh
simint
Einsums
Libxc
gau2grid
Libint2
libecp
h5py
Integratorxx

required **DEP**

optional **DEP**

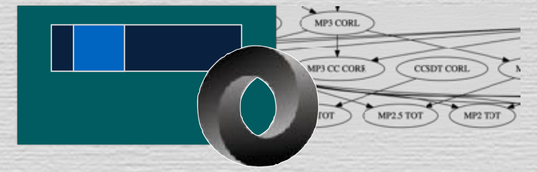
~~Chem~~
Python
C++
C
Fortran



DOWNSTREAM

PSI4 Ecosystem c. 2025

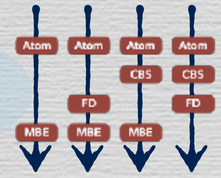
QCVARIABLES



QCSCHEMA

LIBINT

QCEngine

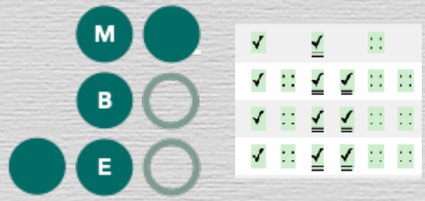


DFT-D3

PSI4 CONDA INSTALL

PSI4

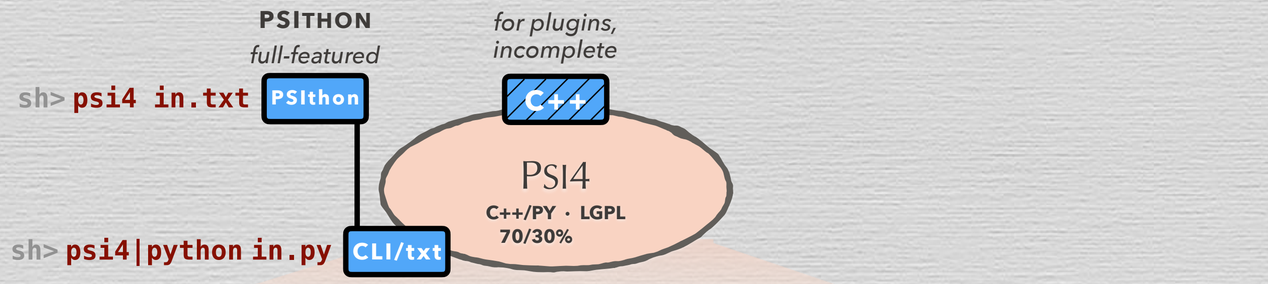
OPEN-SOURCE QUANTUM CHEMISTRY

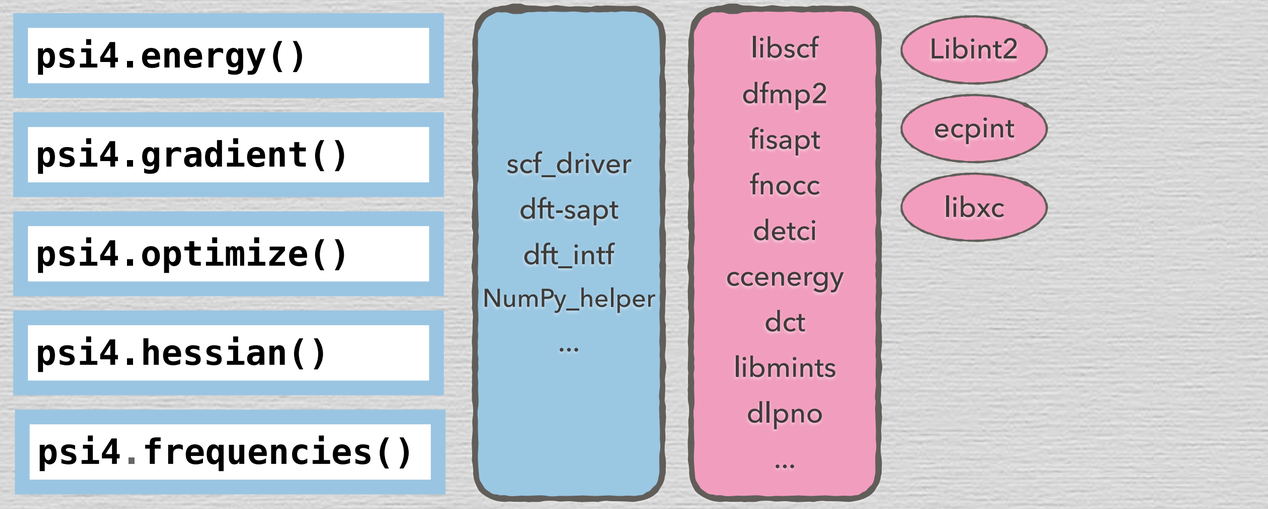


A QCSchema STORY

but first, an input mode overview

- **FULL-FEATURED** (lots of initialisms)
- **SMP** parallelism
- **HYBRID** C++/Python
- **INPUT** by DSL, API, or JSON
- **STRIVE** for easy production-quality user input with five driver "user-facing" functions through which 99+% of QC is run in Psi4, so easy to guess command. Minimal entry points.
- **PY/C++** interface layer below the user layer makes it simple to shift methods btwn languages without disturbing user.
  - **PY → C++** when a reference implementation is optimized in compiled language.
  - **C++ → PY** when a legacy code is refactored so that logic in Python and intensive parts in compiled.
- **SPECIALTIES** include density-fitting & intermolecular interactions





A QCSchema story

but first, an input mode overview

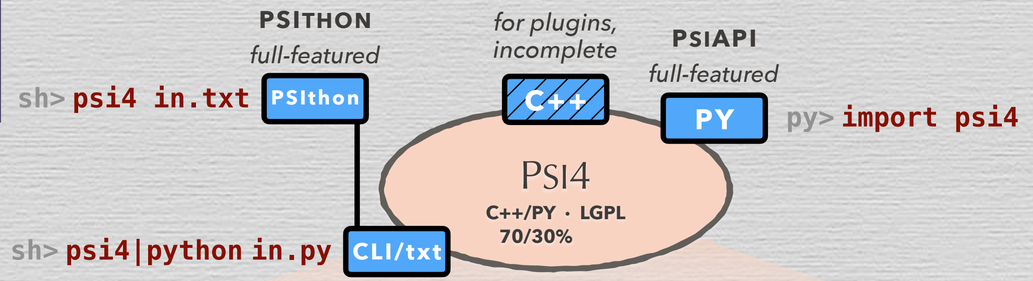
PSIthon

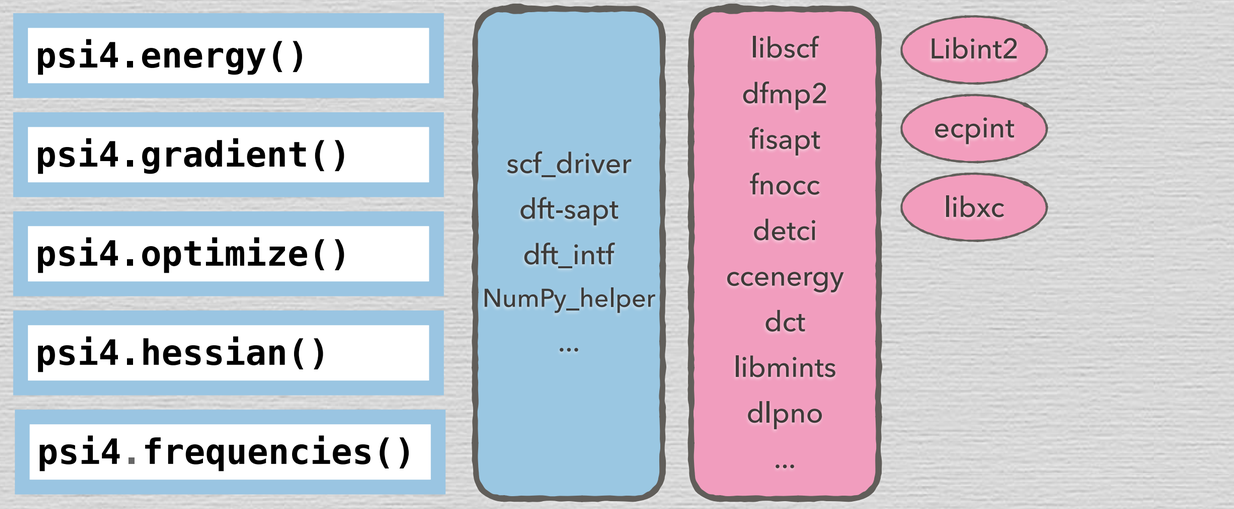
```
molecule {
Ne
Ne 1 3.0
}

set freeze_core True

energy('ccsd(t)/cc-pvtz')
```

PSIAPI

```
import psi4

psi4.geometry('''
Ne
Ne 1 3.0
''')

psi4.set_options({
 'freeze_core': 'True'})

psi4.energy('ccsd(t)/cc-pvtz')
```

added in 2016





A QCSchema STORY

github.com/MolSSI/QCSchema

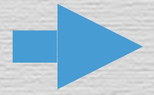
- **COMMUNICATION** channel between all pieces of the QCArchive ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

```
O 0 0 0
H 2 0 0
--
@22Ne 5 0 0
units bohr
```

Snippet 1:




```
{"atom_labels": ["", "", ""],
 "atomic_numbers": [ 8,  1, 10],
 "fix_com": False,
 "fix_orientation": False,
 "fragment_charges": [0.0, 0.0],
 "fragment_multiplicities": [2, 1],
 "fragments": [[0, 1], [2]],
 "geometry": [[0., 0., 0.],
       [2., 0., 0.],
       [5., 0., 0.]],
 "mass_numbers": [16,  1, 22],
 "masses": [15.99491462,  1.00782503, 21.99138511],
 "molecular_charge": 0.0,
 "molecular_multiplicity": 2,
 "name": "HNeO",
 "provenance": {"creator": "QCElemental",
                "routine": "qcelemental.molparse.from_schema",
                "version": "v0.8.0"},
 "real": [ True,  True, False],
 "schema_name": "qcschema_molecule",
 "schema_version": 2,
 "symbols": ["O", "H", "Ne"],
 "validated": True}
```


Snippet 2: QCSchema Molecule from Snippet 1.

A QCSchema STORY

github.com/MolSSI/QCSchema

- **COMMUNICATION** channel between all pieces of the QCArchive ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

AtomicInput

```
{
   "molecule": {
        "geometry": [0, 0, 0, 0, 0, 1],
        "symbols": ["He", "He"]
    },
    "driver": "energy",
    "model": {
        "method": "CCSD(T)",
        "basis": "aug-cc-pVDZ",
    },
    "keywords": {},
}
```

AtomicResult

```
{
    ... AtomicInput ...
    "provenance": {
        "creator": "My QM Program",
        "version": "1.1rc1",
    },
    "properties": {
        "calcinfo_nalpha": 5,
        "scf_total_energy": -5.433191881443323,
        "nuclear_repulsion_energy": 2.11670883436,
        "scf_iterations": 8.0,
    },
    "error": null,
    "return_result": -6.5432123456,
    "success": true,
    "stdout": "...",
    "wavefunction": "..."
}
```

OptimizationResult

QCSchema ATOMIC

analytical single-point computations

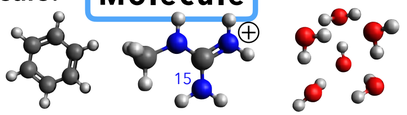
- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

AtomicResultProtocols

stdout: save text output
wavefunction: save orbital info
native_files: save raw prog. files
error_correction: auto-edit&rerun

AtomicInput

schema_version: 1
provenance: creator, version, & RT info
`QCElemental, 0.28.0`
id: tracker for databases
extras: free-form storage
molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF
method: B3LYP-D3, CCSD(T)
basis: 6-31G*, cc-pVQZ
keywords: knobs in SP program
`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

QCSchema ATOMIC

analytical single-point computations


```
'properties': {'calcinfo_nalpha': 10,
               'calcinfo_natom': 2,
               'calcinfo_nbasis': 28,
               'calcinfo_nbeta': 10,
               'calcinfo_nmo': 28,
               'nuclear_repulsion_energy': 16.666666666666668,
               'return_energy': -256.977621071098,
               'return_gradient': array([[0,  0, -6.54470201e-05],
                                         [0,  0,  6.54470212e-05]]),
               'return_hessian': None,
               'scf_dipole_moment': array([0,  0, 4.97379915e-14]),
               'scf_iterations': 5,
               'scf_one_electron_energy': -398.5650589792168,
               'scf_total_energy': -256.977621071098,
               'scf_total_gradient': array([[0,  0, -6.54470201e-05],
                                            [0,  0,  6.54470212e-05]]),
               'scf_total_hessian': None,
               'scf_two_electron_energy': 124.92077124145213},

'extras': {'qcvars': {'CURRENT DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                      'CURRENT ENERGY': -256.977621071098,
                      'CURRENT GRADIENT': array([[0,  0, -6.54470201e-05],
                                                 [0,  0,  6.54470212e-05]]),
                      'CURRENT REFERENCE ENERGY': -256.977621071098,
                      'DD SOLVATION ENERGY': 0.0,
                      'HF KINETIC ENERGY': 256.97713564856093,
                      'HF POTENTIAL ENERGY': -513.9547567196589,
                      'HF TOTAL ENERGY': -256.977621071098,
                      'HF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                  [0,  0,  6.54470212e-05]]),
                      'HF VIRIAL RATIO': 2.000001888971701,
                      'NUCLEAR REPULSION ENERGY': 16.666666666666668,
                      'ONE-ELECTRON ENERGY': -398.5650589792168,
                      'PCM POLARIZATION ENERGY': 0.0,
                      'PE ENERGY': 0.0,
                      'SCF DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]
                      'SCF ITERATION ENERGY': -256.977621071098,
                      'SCF ITERATIONS': 5.0,
                      'SCF TOTAL ENERGIES': array([-256.97759612, -256.97762005, -256.9776
                                                   -256.97762107, -256.97762107, -256.9776
                      'SCF TOTAL ENERGY': -256.977621071098,
                      'SCF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                   [0,  0,  6.54470212e-05]]),
                      'TWO-ELECTRON ENERGY': 124.92077124145213}},
```


AtomicResultProperties (abr.)

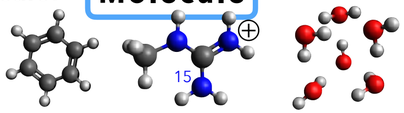
calcinfo_natom
calcinfo_nmo
nuclear_repulsion_energy
return_energy: *E*
return_gradient: *G*

scf_total_*egh*: ref. energy, gradient, etc.
***mtd*_correlation_energy:** post-scf energy
***mtd*_iterations:** count to convergence
***mtd*_mpole_moment:** dipole etc. array

AtomicResult

AtomicInput

schema_version: 1
provenance: creator, version, & RT info
`QCElemental, 0.28.0`
id: tracker for databases
extras: free-form storage
molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

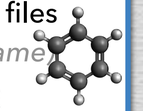
driver: single-point derivative

- energy
- gradient
- Hessian
- properties

E *G* *H*

model: model chemistry or FF
method: B3LYP-D3, CCSD(T)
basis: 6-31G*, cc-pVQZ
keywords: knobs in SP program
`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True
provenance: *(overwrites input)*
`xtb, 1.0.0, psinet, CPU E5-2630`
stdout/stderr: program's text output for ppl
native_files: requested raw program files
molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative

E · *G* ⋮⋮ *H* ▦

properties: key results for single-point

AtomicResultProperties

wavefunction: orbital-level results

WavefunctionProperties

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

AtomicResult

AtomicInput

schema_version: 1

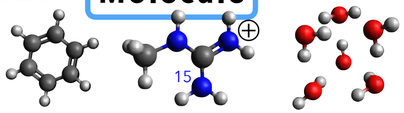
provenance: creator, version, & RT info

`QCElemental, 0.28.0`

id: tracker for databases

extras: free-form storage

molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True

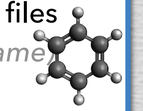
provenance: *(overwrites input)*

`xtb, 1.0.0, psinet, CPU E5-2630`

stdout/stderr: program's text output for ppl

native_files: requested raw program files

molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative

E · *G* ⋮⋮ *H*

properties: key results for single-point

AtomicResultProperties

wavefunction: orbital-level results

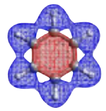
WavefunctionProperties

WavefunctionProperties (abr.)

basis: **BasisSet**

restricted: alpha == beta

orbitals_a/b



density_a/b

fock_a/b

eigenvalues_a/b

occupations_a/b

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

⟨CC|CC⟩

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

GAMESS

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

PSI4

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

NWChem

```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

DATA LAYOUT TRANSLATION

QCSchema: AtomicInput

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {


                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {


                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {


                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {


                        }}
```

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

⟨CC|CC⟩

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

GAMESS

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

PSI4

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

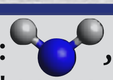
NWChem

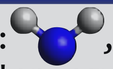
```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

DATA LAYOUT TRANSLATION

MOL & DRIVER STANDARDIZATION

QCSchema: AtomicInput

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method':       ,
    'basis':                },
  'keywords': {



                          }}
```

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method':       ,
    'basis':                },
  'keywords': {



                          }}
```

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method':       ,
    'basis':                },
  'keywords': {



                          }}
```

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method':       ,
    'basis':                },
  'keywords': {



                          }}
```

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

⟨CC|CC⟩

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

GAMESS

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

PSI4

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

NWChem

```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

DATA LAYOUT TRANSLATION

MOL & DRIVER STANDARDIZATION

BASIS & KEYWORDS STANDARDIZATION

QCSchema: AtomicInput

```
{ 'molecule': <molecule>,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-pvdz'    },
  'keywords': {
    'reference': 'rohf'

                          }}
```

```
{ 'molecule': <molecule>,
  'driver': 'energy',
  'model': {
    'method':'ccsd' ,
    'basis': 'accd'        },
  'keywords': {
    'contrl__ispher': 1,
    'contrl__scftyp': 'rohf',
    'ccinp__ncore': 0      }}
```

```
{ 'molecule': <molecule>,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-cc-pvdz'},
  'keywords': {
    'reference': 'rohf'

                          }}
```

```
{ 'molecule': <molecule>,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-cc-pvdz'},
  'keywords': {
    'basis__spherical': True,
    'scf__rohf': True,
    'qc_module': 'tce'     }}
```

A QCSchema story

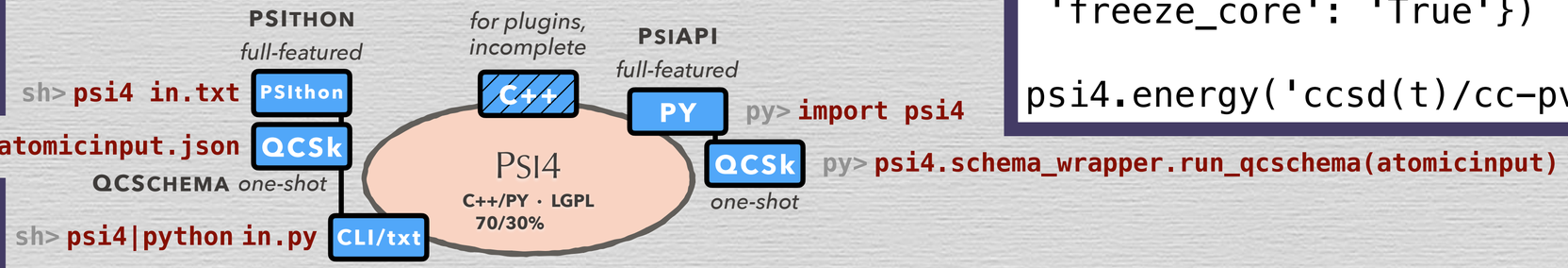
one-shot inputs now possible

PSIAPI

```
import psi4

psi4.geometry('''
Ne
Ne 1 3.0
''')

psi4.set_options({
 'freeze_core': 'True'})

psi4.energy('ccsd(t)/cc-pvtz')
```

PSITHON

```
molecule {
Ne
Ne 1 3.0
}

set freeze_core True

energy('ccsd(t)/cc-pvtz')
```




```
{
 'molecule': {
  'symbols': ['Ne', 'Ne'],
  'geometry':
    [0, 0, 0, 5.67, 0, 0]
 },
 'driver': 'energy',
 'model': {
  'method': 'ccsd(t)',
  'basis': 'cc-pvtz'
 },
 'keywords':
  {'freeze_core': 'True'}
}
```

QCSCHEMA

- **PAYOFF** Psi4 gets a a structured, single-blob means of launching and recording the great majority of computations. And it's a community format, not our own.

★ Mapping calculation input into a structured format (harder for API-based than early QC programs) can open new ways of interacting with users & workflows.

QCSchema ATOMIC

analytical single-point computations

- As soon as move beyond Psi4 domain-specific output and start writing out QCSchema **AtomicResult** files, it's rather like entering into a contract that fields mean defined things.


```
'properties': {'calcinfo_nalpha': 10,
               'calcinfo_natom': 2,
               'calcinfo_nbasis': 28,
               'calcinfo_nbeta': 10,
               'calcinfo_nmo': 28,
               'nuclear_repulsion_energy': 16.666666666666668,
               'return_energy': -256.977621071098,
               'return_gradient': array([[0,  0, -6.54470201e-05],
                                         [0,  0,  6.54470212e-05]]),
               'return_hessian': None,
               'scf_dipole_moment': array([0,  0, 4.97379915e-14]),
               'scf_iterations': 5,
               'scf_one_electron_energy': -398.5650589792168,
               'scf_total_energy': -256.977621071098,
               'scf_total_gradient': array([[0,  0, -6.54470201e-05],
                                            [0,  0,  6.54470212e-05]]),
               'scf_total_hessian': None,
               'scf_two_electron_energy': 124.92077124145213},

 'extras': {'qcvars': {'CURRENT DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                       'CURRENT ENERGY': -256.977621071098,
                       'CURRENT GRADIENT': array([[0,  0, -6.54470201e-05],
                                                  [0,  0,  6.54470212e-05]]),
                       'CURRENT REFERENCE ENERGY': -256.977621071098,
                       'DD SOLVATION ENERGY': 0.0,
                       'HF KINETIC ENERGY': 256.97713564856093,
                       'HF POTENTIAL ENERGY': -513.9547567196589,
                       'HF TOTAL ENERGY': -256.977621071098,
                       'HF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                   [0,  0,  6.54470212e-05]]),
                       'HF VIRIAL RATIO': 2.000001888971701,
                       'NUCLEAR REPULSION ENERGY': 16.666666666666668,
                       'ONE-ELECTRON ENERGY': -398.5650589792168,
                       'PCM POLARIZATION ENERGY': 0.0,
                       'PE ENERGY': 0.0,
                       'SCF DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                       'SCF ITERATION ENERGY': -256.977621071098,
                       'SCF ITERATIONS': 5.0,
                       'SCF TOTAL ENERGIES': array([-256.97759612, -256.97762005, -256.9776209 ,
                                                    -256.97762107, -256.97762107, -256.97762107]),
                       'SCF TOTAL ENERGY': -256.977621071098,
                       'SCF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                    [0,  0,  6.54470212e-05]]),
                       'TWO-ELECTRON ENERGY': 124.92077124145213}},
```


AtomicResultProperties (abr.)

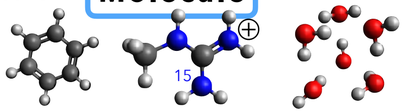
calcinfo_natom
calcinfo_nmo
nuclear_repulsion_energy
return_energy: *E* ·
return_gradient: *G* ⋮⋮⋮

scf_total_*egh*: ref. energy, gradient, etc.
***mtd*_correlation_energy:** post-scf energy
***mtd*_iterations:** count to convergence
***mtd*_mpole_moment:** dipole etc. array

AtomicResult

AtomicInput

schema_version: 1
provenance: creator, version, & RT info
`QCElemental, 0.28.0`
id: tracker for databases
extras: free-form storage
molecule: **Molecule**



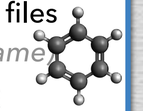
protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative
- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF
method: B3LYP-D3, CCSD(T)
basis: 6-31G*, cc-pVQZ
keywords: knobs in SP program
`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True
provenance: *(overwrites input)*
`xtb, 1.0.0, psinet, CPU E5-2630`
stdout/stderr: program's text output for ppl
native_files: requested raw program files
molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative
E · *G* ⋮⋮⋮ *H*

properties: key results for single-point

AtomicResultProperties

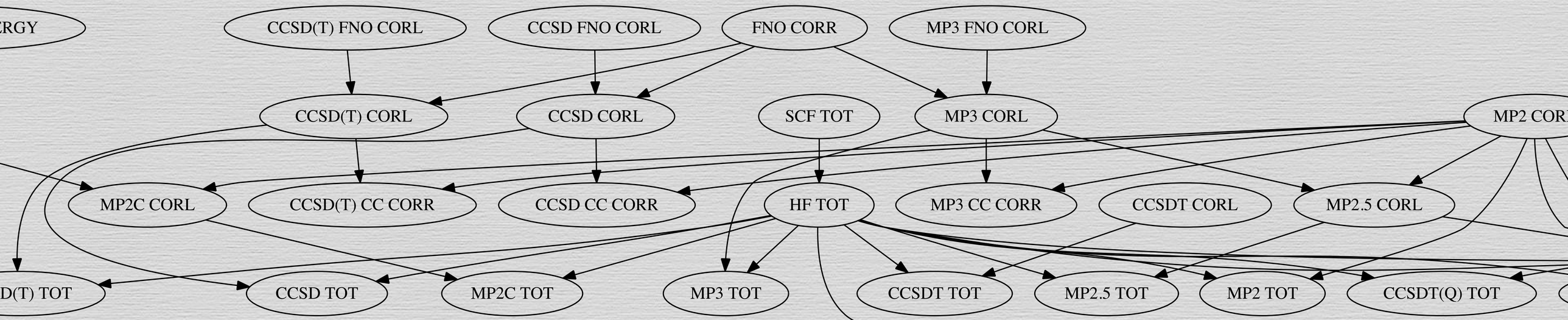
wavefunction: orbital-level results

WavefunctionProperties

QCVARIABLES

common QC definitions & combining rules

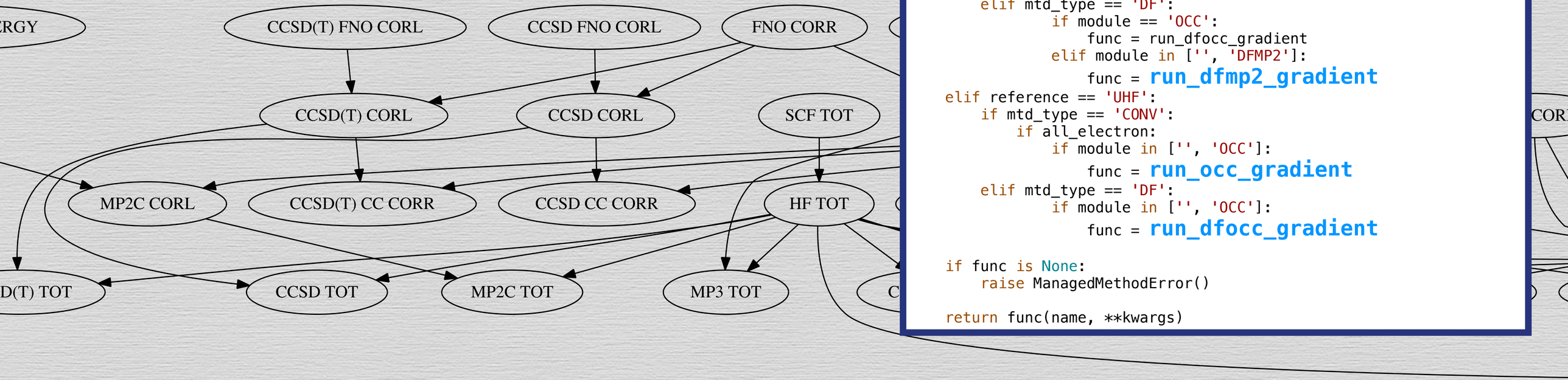
- As soon as move beyond Psi4 domain-specific output and start writing out QCSchema **AtomicResult** files, it's rather like entering into a contract that fields mean defined things.



QCVARIABLES

common QC definitions & combining rules

- As soon as move beyond Psi4 domain-specific output and start writing out QCSchema **AtomicResult** files, it's rather like entering into a contract that fields mean defined things.
- When dealing with modular software, there's going to be certain modules with different capabilities and strengths, even for the same method. This is true for internal or external modules.
- Instituted "managed methods" so users could remain oblivious: **`gradient('mp2')`** gets the most efficient module -or- intentional: **`set qc_module dfocc; gradient('mp2')`** gets the directed module.
- How can we fulfill this contract and help manage complexity at the same time?




```python
def select_mp2_gradient(name, **kwargs):
    """Function selecting the algorithm for a MP2 gradient call
    and directing to specified or best-performance default modules.
    """
    reference = core.get_option('SCF', 'REFERENCE')
    type_var, _, mtd_type = method_algorithm_type(name)
    module = core.get_global_option('QC_MODULE')
    all_electron = (core.get_global_option('FREEZE_CORE') == "FALSE")

    func = None
    if reference == 'RHF':
        if mtd_type == 'CONV':
            if all_electron:
                if module in ['', 'OCC']:
                    func = run_occ_gradient
        elif mtd_type == 'DF':
                if module == 'OCC':
                    func = run_dfocc_gradient
                elif module in ['', 'DFMP2']:
                    func = run_dfmp2_gradient
    elif reference == 'UHF':
        if mtd_type == 'CONV':
            if all_electron:
                if module in ['', 'OCC']:
                    func = run_occ_gradient
        elif mtd_type == 'DF':
                if module in ['', 'OCC']:
                    func = run_dfocc_gradient

    if func is None:
        raise ManagedMethodError()

    return func(name, **kwargs)
```

QCVARIABLES

common QC definitions & combining rules

- As soon as move beyond Psi4 domain-specific output and start writing out QCSchema **AtomicResult** files, it's rather like entering into a contract that fields mean defined things.

★ Prefer and define programmatic output for data standards

```
'extras': {'qcvars': {'CURRENT DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                      'CURRENT ENERGY': -256.977621071098,
                      'CURRENT GRADIENT': array([[0,  0, -6.54470201e-05],
                                                 [0,  0,  6.54470212e-05]]),
                      'CURRENT REFERENCE ENERGY': -256.977621071098,
                      'DD SOLVATION ENERGY': 0.0,
                      'HF KINETIC ENERGY': 256.97713564856093,
                      'HF POTENTIAL ENERGY': -513.9547567196589,
                      'HF TOTAL ENERGY': -256.977621071098,
                      'HF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                  [0,  0,  6.54470212e-05]]),
                      'HF VIRIAL RATIO': 2.000001888971701,
                      'NUCLEAR REPULSION ENERGY': 16.666666666666668,
                      'ONE-ELECTRON ENERGY': -398.5650589792168,
                      'PCM POLARIZATION ENERGY': 0.0,
                      'PE ENERGY': 0.0,
                      'SCF DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                      'SCF ITERATION ENERGY': -256.977621071098,
                      'SCF ITERATIONS': 5.0,
                      'SCF TOTAL ENERGIES': array([-256.97759612, -256.97762005, -256.9776209 ,
                                                   -256.97762107, -256.97762107, -256.97762107]),
                      'SCF TOTAL ENERGY': -256.977621071098,
                      'SCF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                   [0,  0,  6.54470212e-05]]),
                      'TWO-ELECTRON ENERGY': 124.92077124145213}},
```

QCVariables GLOSSARY

DFT FUNCTIONAL TOTAL ENERGY

The total electronic energy [E_h] for the underlying functional of the requested DFT method, without any dispersion correction; the first four terms in Eq. (4) or (1). Quantity E_{FCTL} in Eqs. (4) and (1). Unless the method includes a dispersion correction, this quantity is equal to `SCF TOTAL ENERGY`.

DFT TOTAL ENERGY

The total electronic energy [E_h] for the requested DFT method, E_{DFT} in Eq. (1).

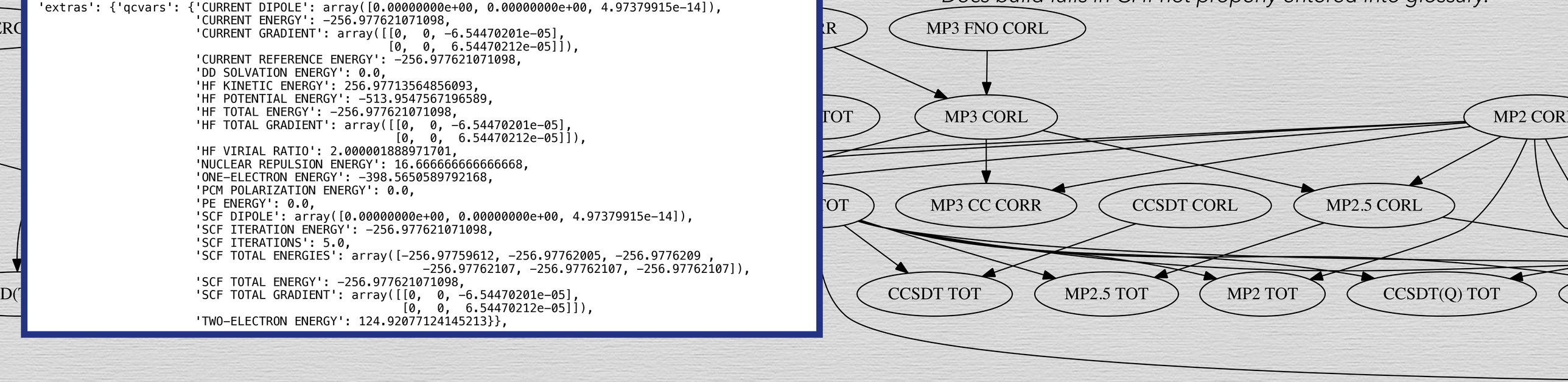
$$\begin{aligned} E_{\mathrm{DFT}} &= E_{NN} + E_{1e^-} + E_{2e^-} + E_{\mathrm{xc}} + E_{\mathrm{-D}} + E_{\mathrm{DH}} \\ &= E_{\mathrm{FCTL}} + E_{\mathrm{-D}} + E_{\mathrm{DH}} \\ &= E_{\mathrm{SCF}} + E_{\mathrm{DH}} \end{aligned}$$

Unless the method is a DFT double-hybrid, this quantity is equal to `SCF TOTAL ENERGY`. If the method is neither a double-hybrid, nor dispersion corrected, this quantity is equal to `DFT FUNCTIONAL TOTAL ENERGY`.

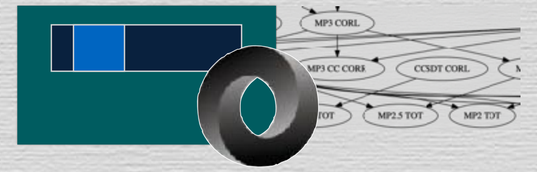
DFT TOTAL GRADIENT

The total electronic gradient [E_h/a0] of the requested DFT method, (3*nat, 3).

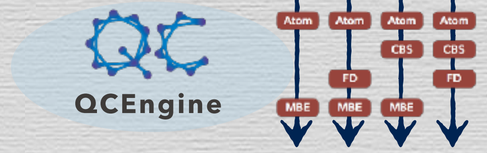
Modules register a var with `# P::e MODULENAME`.
Docs build fails in CI if not properly entered into glossary.



METADATA

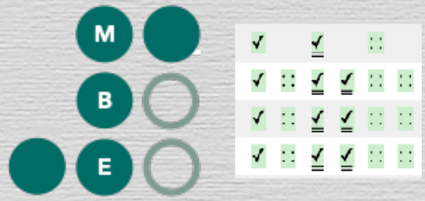


LIBINT



PSI4

OPEN-SOURCE QUANTUM CHEMISTRY



LIBINT

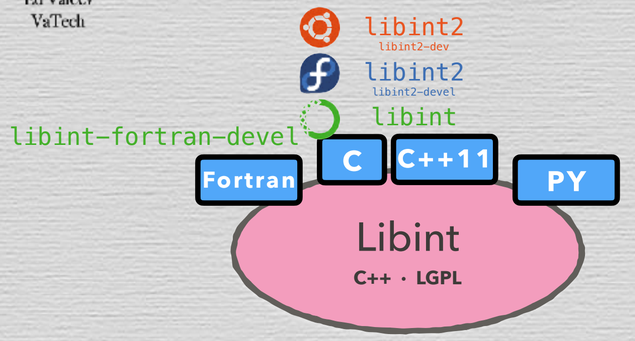
Ed Valeev
VaTech

A LIBINT STORY

play nicely in the *runtime* software stack

Andy Simmonett
NIH → QC Ware

Maximilian Scheurer
Heidelberg → Covestro



- **DEVELOPED** by E. Valeev for 20y
- **FULL** service integrals library
- **INTERFACES** in multiple languages/APIs
- **HIGHLY** configurable by integral classes, derivatives, orderings, & angular momentum levels
- **LARGE** size at production configurations, >GB generated source or built library

In **2020** Psi4 started updating from 15yo Libint1 to maintained Libint2

Open **Libint2 2022 Strategy** #2442

step	status	libint ver & branch	Psi4	tarball[1]	order	component style
1[3]	longstanding L1	L1 evaleev:5c89451	v1.3	–	gss	`s`
2[4]	TEI L2	loriab:l2cmake evaleev/libint#148	20Nov20, after #1721, v1.4, 1.5	L: `7-7-4-7-7-5_1`, MW: `5-4-3-6-5-4_1`	gss	`g5`
3[5]	OEI L2	ditto step 2	11Mar22, after #2388	L: `5-4-3-6-5-4_mm25f12ob2`, MW: `5-4-3-6-5-4_mm4ob2`	gss	`g5`
B[6]	upstream L2 cmake	loriab:new-cmake-harness-lab-rb1 evaleev/libint#233	23Mar22, after #2413, v1.6, v1.7	`5-4-3-6-5-4_mm4f12ob2.tgz`	gss	`eri_c4_d1_l5`
C[2]	McMurchie Davidson	any	31Mar22, after #2414, v1.6, v1.7			
~~A[7]~~	~~standard ze~~ ordering	ditto step B	#2537	ditto step B	sss	`g5`
A' [8]	flex solid-harm ordering	loriab:new-cmake-2023-take2-b evaleev/libint#259	8May23, after #2861, v1.8, v1.9 (see also A'')	`libint-2.7.2-pcst1-5-4-3-5-5-4_mm4f12ob2_1`	sss	`eri_c4_d1_l5`
A'' [9]	flex solid-harm ordering, libtool-based	upstream master any pcint after evaleev/libint@`10ca429`	3Dec23, after #3047, v1.9 (see also A')	libint-2.8.0-dev2-5-4-3-6-5-4_mm10f12ob2_0.tgz	ss	`eri_hhhh_d1`
A''' [10]	flex solid-harm ordering, libtool-based, released	any upstream release >=2.8.1	18Jan24, after #3107, v1.9.1		ss	`eri_hhhh_d1`

LIBINT

A LIBINT STORY

play nicely in the *runtime* software stack

both projects subject to own expectations:

Libint assumed each instance targeted toward a **SINGLE** QC consumer

Psi4 vulnerable to **SEGFAULT** if meagerly configured

Psi4 vulnerable to **WRONG RESULTS** if configured w/std orderings

Portable libint #190

Open · susilehtola opened this issue on Sep 3, 2020 · 26 comments · May be fixed by #205

susilehtola commented on Sep 3, 2020 — Contributor

Hi,

it looks like libint2 can be compiled in `Nsph x Ncart x Nshell_set = 2 x 5 x 2 = 20` different possible ways. Even though only 5 of these are currently used by major codes, this is a nightmare for distribution packaging, since there's no way we can ship 20 mutually incompatible copies of a library. I'm horrified about shipping even two since it takes forever to compile libint on the Fedora build system.

Would it be possible to modify libint so that the orderings for the cartesians, solid harmonics, and shell sets are set by some function, and libint does all the necessary shuffles behind the scenes? This would be a minimal solution to the problem, since one could still set the default orderings at configure time, thus avoiding any extra shuffles if the compiled ordering matches the wanted one.

loria — Collaborator

psi is non-standard for solid harmonic and probably precipitated this thread. (below is harvested from libint wiki notes.) small mercy is that psi differs in the one dimension that can be chosen at library build time, not generator build time.

```
#                  sh         cart       shell_set  used_by
#                  --------   --------   ---------  -------
#   sss - search for standard + standard + standard = mpqc4, cp2k
#   sso - search for                     + orca
#   sis - search for          + intv3    + standard
#   sio - search for                     + orca
#   sgs - search for          + gamess   + standard = gamess
#   sgo - search for                     + orca
#   sos - search for          + orca     + standard
#   soo - search for                     + orca     = orca
#   sbs - search for          + bagel    + standard = bagel
#   sbo - search for                     + orca
#   gss - search for gaussian + standard + standard = psi4
#   gso - search for                     + orca
#   gis - search for          + intv3    + standard
#   gio - search for                     + orca
#   ggs - search for          + gamess   + standard
#   ggo - search for                     + orca
#   gos - search for          + orca     + standard
#   goo - search for                     + orca
#   gbs - search for          + bagel    + standard
#   gbo - search for                     + orca
```

LIBINT

A LIBINT STORY

play nicely in the *runtime* software stack

. . . so Psi4 has contributed back upstream

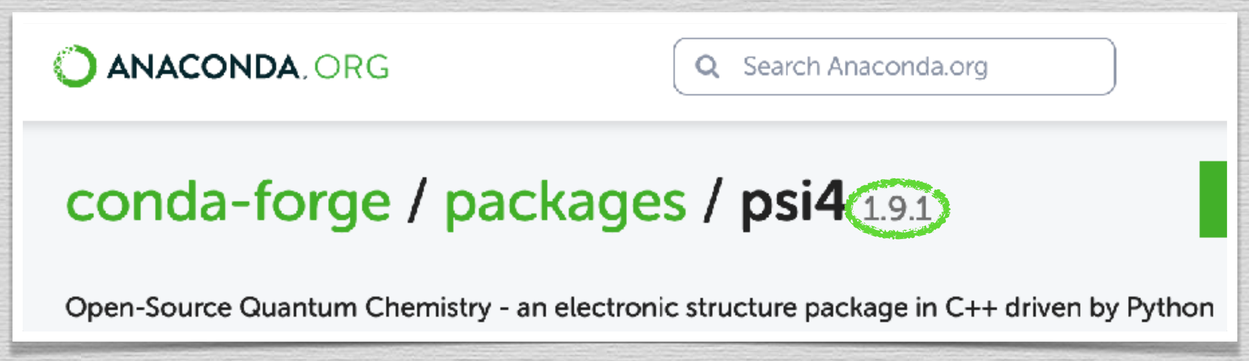
- expanded to Windows **ARCH** & CMake **BUILDSYS**
- added **LINK-TIME** & **RUN-TIME** configuration checking
- maintain a **HIGH-AM** (12), high-deriv build on conda
- enabled **RUN-TIME** ordering selection for OSS consumers

```
                                  cart       shell_set   used_by
                                  --------   ---------   -------
ss - library integrals use ordering  standard + standard  = mpqc4, cp2k, psi4
so - library integrals use ordering           + orca
is - library integrals use ordering  intv3    + standard  = mpqc3
io - library integrals use ordering           + orca
gs - library integrals use ordering  gamess   + standard  = gamess
go - library integrals use ordering           + orca
os - library integrals use ordering  orca     + standard
oo - library integrals use ordering           + orca      = orca
bs - library integrals use ordering  bagel    + standard  = bagel
bo - library integrals use ordering           + orca
```

(psi4 requires runtime-setting of solid harmonic ordering to Gaussian)

- **PAYOFF** delete much home-grown code
- **PAYOFF** expand capabilities with new integrals types
- **PAYOFF** return to upstream as of Dec 2023 v2.8, so new Psi4 releases packageable w/other OSS QC
- **PAYOFF** conda-forge packaging

incompatibilities frustrate **PACKAGING**, so was stuck c. 2019



Fedora 42	2025	Psi4 c.2024	1.9.1-6.fc42
Fedora 41			1.9.1-3.fc41
Fedora 40	2024	Psi4 c.2019	1.3.2-22.fc40

★ build broadly for consumers & defer configuration to runtime where possible

A LIBINT STORY

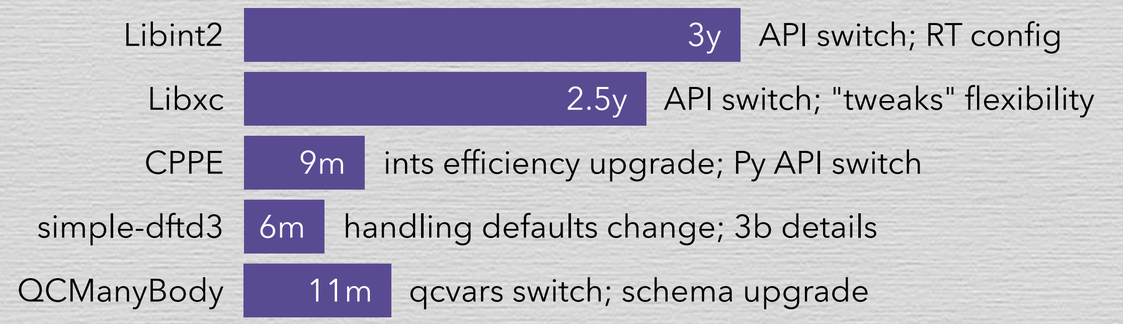
play nicely in the *runtime* software stack

Open **Libint2 2022 Strategy** #2442

step	status	libint ver & branch	Psi4	tarball[1]	order	component style
1[3]	longstanding L1	L1 evaleev:5c89451	v1.3	—	gss	`s`
2[4]	TEI L2	loriab:l2cmake evaleev/libint#148	20Nov20, after #1721, v1.4, 1.5	L: `7-7-4-7-7-5_1`, MW: `5-4-3-6-5-4_1`	gss	`g5`
3[5]	OEI L2	ditto step 2	11Mar22, after #2388	L: `5-4-3-6-5-4_mm25f12ob2`, MW: `5-4-3-6-5-4_mm4ob2`	gss	`g5`
B[6]	upstream L2 cmake	loriab:new-cmake-harness-lab-rb1 evaleev/libint#233	23Mar22, after #2413, v1.6, v1.7	`5-4-3-6-5-4_mm4f12ob2.tgz`	gss	`eri_c4_d1_l5`
C[2]	McMurchie Davidson	any	31Mar22, after #2414, v1.6, v1.7			
~~A~~[7]	~~standard ze~~ ordering	ditto step B	#2537	ditto step B	sss	`g5`
A' [8]	flex solid-harm ordering	loriab:new-cmake-2023-take2-b evaleev/libint#259	8May23, after #2861, v1.8, v1.9 (see also A'')	`libint-2.7.2-pcst1-5-4-3-6-5-4_mm4f12ob2_1`	sss	`eri_c4_d1_l5`
A'' [9]	flex solid-harm ordering, libtool-based	upstream master any pcint after evaleev/libint@`10ca429`	3Dec23, after #3047, v1.9 (see also A')	libint-2.8.0-dev2-5-4-3-6-5-4_mm10f12ob2_0.tgz	ss	`eri_hhhh_d1`
A''' [10]	flex solid-harm ordering, libtool-based, released	any upstream release >=2.8.1	18Jan24, after #3107, v1.9.1		ss	`eri_hhhh_d1`

- Libint1 to Libint2 conversion took many stages, not at all surprising for a change of this magnitude even to a cleaner API with negative thousands LOC
- lag and needing mutual changes and then consolidating those changes into tags/releases is a general trend, though.

TIME ELAPSED BEFORE PSI4 MASTER USING TAGGED DEPENDENCY

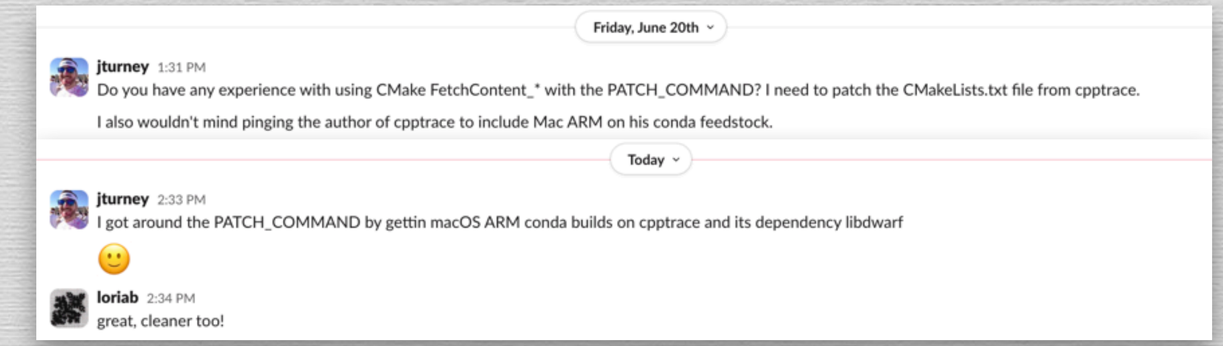


LIB/INT

A LIBINT STORY

play nicely in the *runtime* software stack

- Libint1 to Libint2 conversion took many stages, not at all surprising for a change of this magnitude even to a cleaner API with negative thousands LOC
- lag and needing mutual changes and then consolidating those changes into tags/releases is a general trend, though.



needing tweaks of upstream is ubiquitous in an ecosystem

TIME ELAPSED BEFORE PSI4 MASTER USING TAGGED DEPENDENCY

Libint2	3y	API switch; RT config
Libxc	2.5y	API switch; "tweaks" flexibility
CPPE	9m	ints efficiency upgrade; Py API switch
simple-dftd3	6m	handling defaults change; 3b details
QCManyBody	11m	qcvars switch; schema upgrade

★ interfacing two codes is rarely unilateral. Seize any opportunity to work with other devs in real time.

PROGRAMMATIC METADATA

cater to tools as well as people

- We've solved the uncertainty over ordering configuration of published Libint packages for OSS
- But there are several other things that consumers may need or want to know from their consumed software. Modular software needs to expose details to all of people, runtime artifact, and build system.
- Some may be needed at consumer **COMPILE-TIME**:
  - require an installation to supply certain derivatives or classes of integrals or compiled extensions
  - be sure to define your options via CMake components
  - packaging works now because packagers & project devs communicate (e.g., I tell Susi to enable 1-body Hessians). Give computers the tools to check without oversight.

REQUIREMENTS FOR PSI4 CONSUMING: CMakeLists.txt

```
find_package(gdma 2.3.3 CONFIG COMPONENTS Python)

find_package(Libxc 7.0.0 CONFIG COMPONENTS C)

find_package(Libint2 2.8.1 CONFIG
  COMPONENTS
    ss
    CXX_ho
    impure_sh
    "eri_${amchar}_d0" eri_ggg_d0 eri_gg_d0 onebody_g_d0
     eri_dddd_d1       eri_fff_d1 eri_ff_d1 onebody_f_d1
                                            onebody_f_d2)

find_package(simint 0.7 CONFIG
  COMPONENTS am${MAX_AM_ERI} der0)
```

CONSUMED: simintConfig.cmake.in

```
...
set (simint_der0_FOUND 1)  # always present

foreach(_d RANGE ${simint_MAXAM})  # simint_MAXAM
   set (simint_am${d}_FOUND 1)
endforeach()

check_required_components(simint)
...
```

PSI4 EXPORTED FEATURES

```
find_package(psi4)
message("psi4 components: ${psi4_FOUND_COMPONENTS}")
#> psi4 components: simint;einsums
```

PROGRAMMATIC METADATA

cater to tools as well as people

- We've solved the uncertainty over ordering configuration of published Libint packages for OSS
- But there are several other things that consumers may need or want to know from their consumed software. Modular software needs to expose details to all of people, runtime artifact, and build system.
- Some may be needed at consumer **COMPILE-TIME**:
  - require an installation to supply certain derivatives or classes of integrals or compiled extensions
  - be sure to define your options via CMake components
  - packaging works now because packagers & project devs communicate (e.g., I tell Susi to enable 1-body Hessians). Give computers the tools to check without oversight.
- Additionally, they may be needed at consumer **RUN-TIME**:
  - whether Hessian integrals are available or else redirect to finite difference or whether a rung of functionals is available or else disable those methods
  - be sure to provide accessors so consumers can apply logic.

LIBINT2 CAPABILITIES

```
libint2::initialize();
printf("SHGShell: %d\n", libint2::solid_harmonics_ordering());
// if patched
printf("Configuration: %s\n",
    libint2::configuration_accessor().c_str());
printf("Supports: dddd=%d mmmm=%d\n",
    libint2::supports("eri_dddd_d0"),
    libint2::supports("eri_mmmm_d0"));
libint2::finalize();

#> SHGShell: 1
#> Configuration: eri_dddd_d0;eri_ffff_d0;ss;...
#> Supports: dddd=1 mmmm=0
```

PSI4 EXTRA CAPABILITIES

```
print(psi4.addons())
#> ['adcc', 'ambit', 'bse', 'cct3', 'cfour', 'chemps2',
'cppe', 'ddx', 'dftd3', 'dftd4', 'dkh', 'ecpint',
'einsums', 'forte', 'gauxc', 'gcp', 'gdma', 'geometric',
'gpu_dfcc', 'integratorxx', 'ipi', 'libefp', 'mdi',
'mp2d', 'mrcc', 'openfermionpsi4', 'pcmsolver',
'psi4fockci', 'psixas', 'resp', 'simint', 'snsmp2',
'v2rdm_casscf']
```

PROGRAMMATIC METADATA

cater to tools as well as people

- We've solved the uncertainty over ordering configuration of published Libint packages for OSS
- But there are several other things that consumers may need or want to know from their consumed software. Modular software needs to expose details to all of people, runtime artifact, and build system.
- Some need to be brought to **PEOPLE**'s attention:
  - modular software can get buried behind the front-end — I'm sure Libxc has many users who have never heard of it. Psi4 uses some addons anonymously.
  - be sure to make your name, version, & citation accessible, so consuming programs can toast you.

OPTKING

```
def welcome():
    return """
    \t\t\t-----------------------------------------\n
    \t\t\t OPTKING 3.0: for geometry optimizations \n
    \t\t\t     By R.A. King, Bethel University     \n
    \t\t\t        with contributions from          \n
    \t\t\t    A.V. Copan, J. Cayton, A. Heide      \n
    \t\t\t-----------------------------------------\n
    """
```

LIBINT

```
const char *libint_reference(void) {
    std::string ref =
      "Libint: @LIBINT_DESCRIPTION@, Version " +
      std::string(libint_version_string(true)) +
      " Edward F. Valeev, http://libint.valeyev.net/";
    return ref;
}
```

LIBXC

```
   => LibXC <=

    Version 6.2.2
    S. Lehtola, C. Steigemann, M. J.T. Oliveira, and M. A.L.
Marques.,  SoftwareX 7, 1–5 (2018) (10.1016/j.softx.2017.11.002)
```

PSI4

```
print(psi4.citation_formatter())
#> Psi4 v1.10a1.dev86, J. Chem. Phys. 152(18) 184108 (2020)
   (10.1063/5.0006002)

print(psi4.citation_formatter("DOI: {doi}"))
#> DOI: 10.1063/5.0006002
```

PROGRAMMATIC METADATA

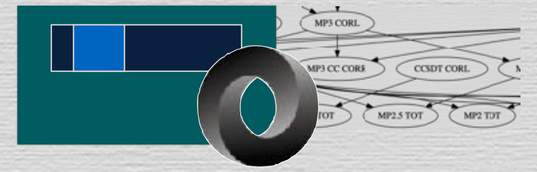
cater to tools as well as people

- We've solved the uncertainty over ordering configuration of published Libint packages for OSS
- But there are several other things that consumers may need or want to know from their consumed software. Modular software needs to expose details to all of people, runtime artifact, and build system.
- Some need to be brought to **PEOPLE**'s attention:
  - modular software can get buried behind the front-end — I'm sure Libxc has many users who have never heard of it. Psi4 uses some addons anonymously.
  - be sure to make your name, version, & citation accessible, so consuming programs can toast you.
- Projects increasingly good at recording their citation and authorship and license somewhere, and GitHub conventions have helped a lot. Psi4 uses codemeta.json, but I've incresingly used **CITATION.CFF** that facilitates both a journal and a repository citation.

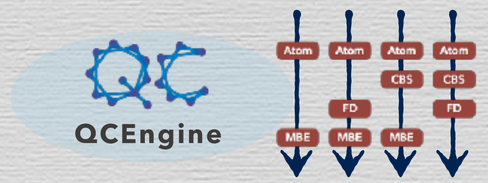
TORSIONDRIVE: CITATION.cff

```
cff-version: 1.2.0
title: TorsionDrive
type: software
authors:
  - given-names: Lee-Ping
    family-names: Wang
    affiliation: University of California Davis
    orcid: 'https://orcid.org/0000-0003-3072-9946'
  - given-names: Yudong
    family-names: Qiu
    orcid: 'https://orcid.org/0000-0003-4345-8356'
repository-code: 'https://github.com/lpwgroup/torsiondrive'
abstract: Dihedral scanner with wavefront propagation.
keywords:
  - molecular geometry optimization
  - molecular force-field fitting
license: MIT
version: 1.2.0
date-released: '2025-05-03'
preferred-citation:
  type: article
  authors:
  - family-names: "Yudong"
    given-names: "Qiu"
  - family-names: "Wang"
    given-names: "Lee-Ping"
  doi: "10.1063/5.0009232"
  journal: "The Journal of Chemical Physics"
  month: 6
  start: 244116
  title: "Driving torsion scans with wavefront propagation"
  issue: 24
  volume: 152
  year: 2020
```

★ Give your module the tools to convey its requirements, capabilities, & attributions



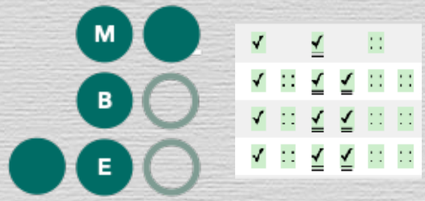
DISTRIBUTED DRIVER



QCENGINE

PSI4

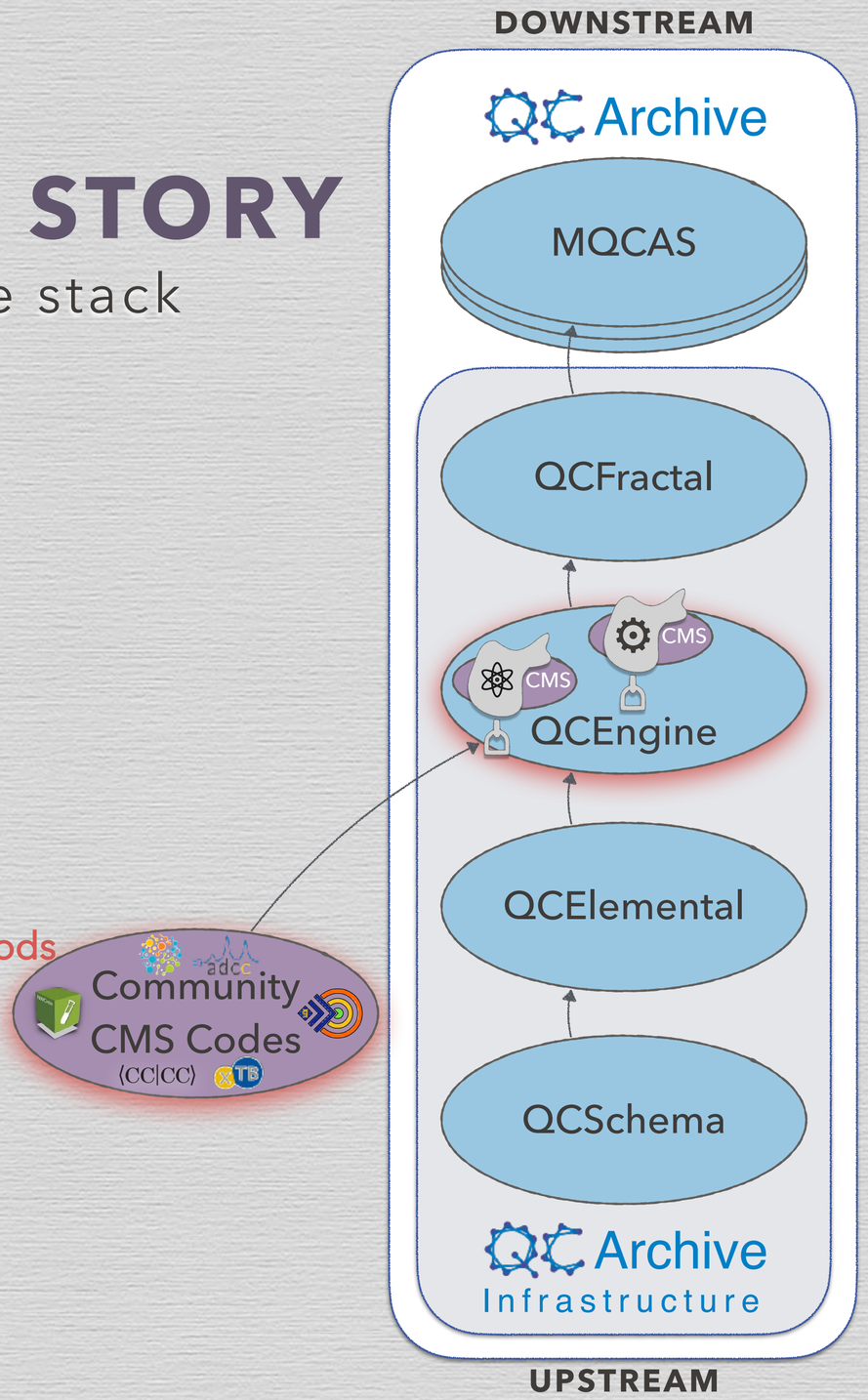
OPEN-SOURCE QUANTUM CHEMISTRY



QCEngine

A QCENGINE STORY

the QCArchive stack



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- database storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

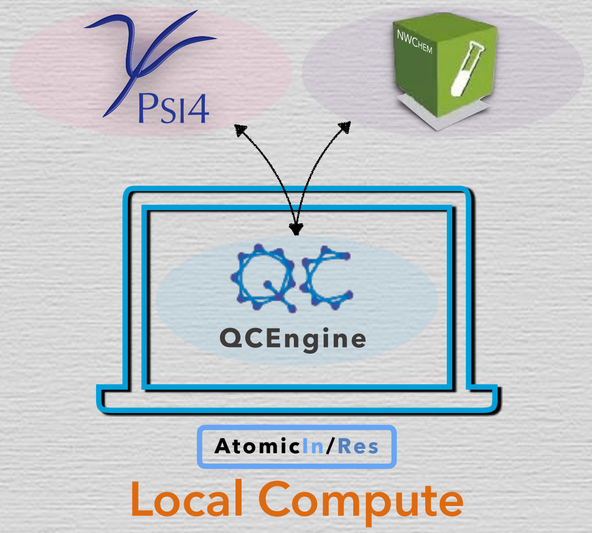
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

- the difficult part – coded QC methods
- structured output uncommon
- DSL input; API/schema uncommon

A QCEngine STORY: PYTHON QCSchema RUNNER

github.com/MolSSI/QCEngine

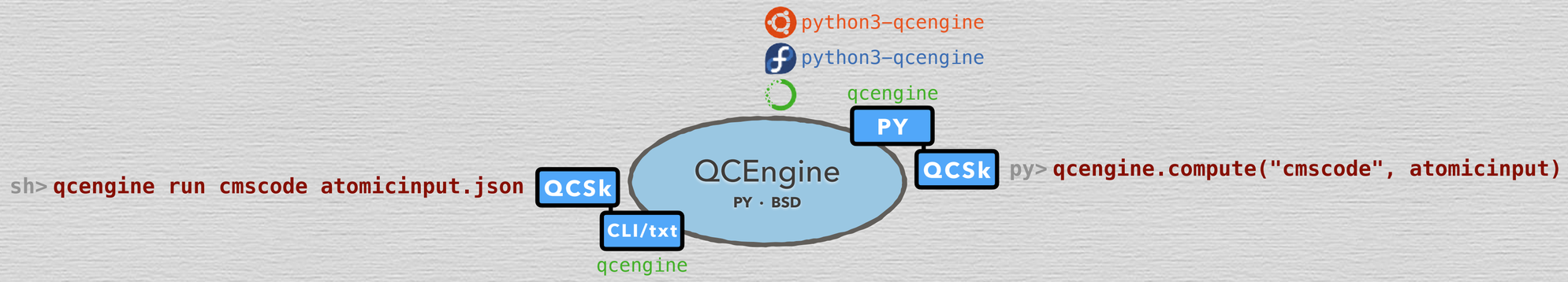


CMDLINE

```
                  cfour
> qcengine run    molpro  atomicinput
                  psi4
                  dftd4
```

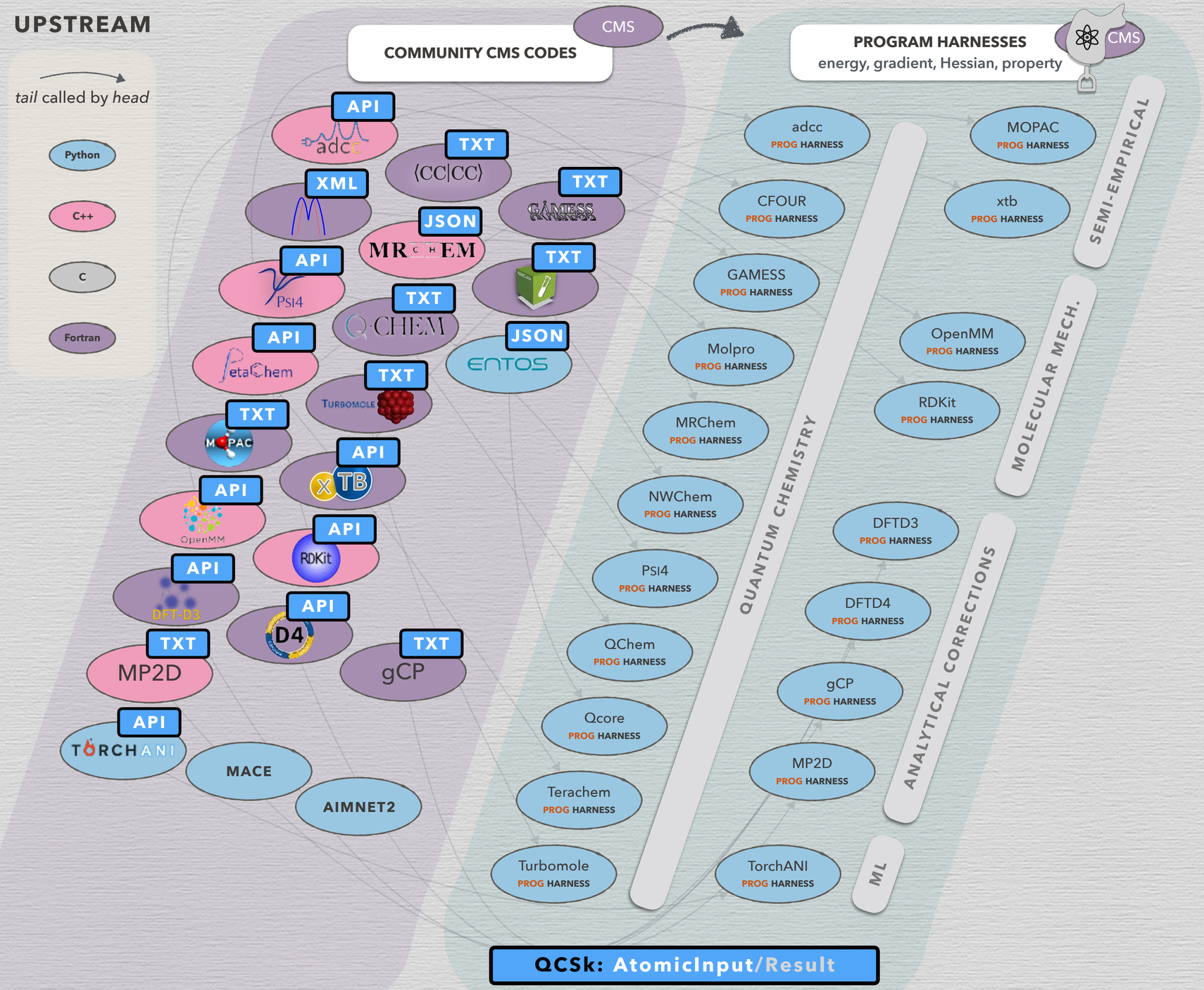
PYAPI

```
                                  cfour
qcengine.compute(atomicinput,  "molpro", task_config({"nnodes": 4, "memory": 10})
                                  psi4
                                  dftd4
```

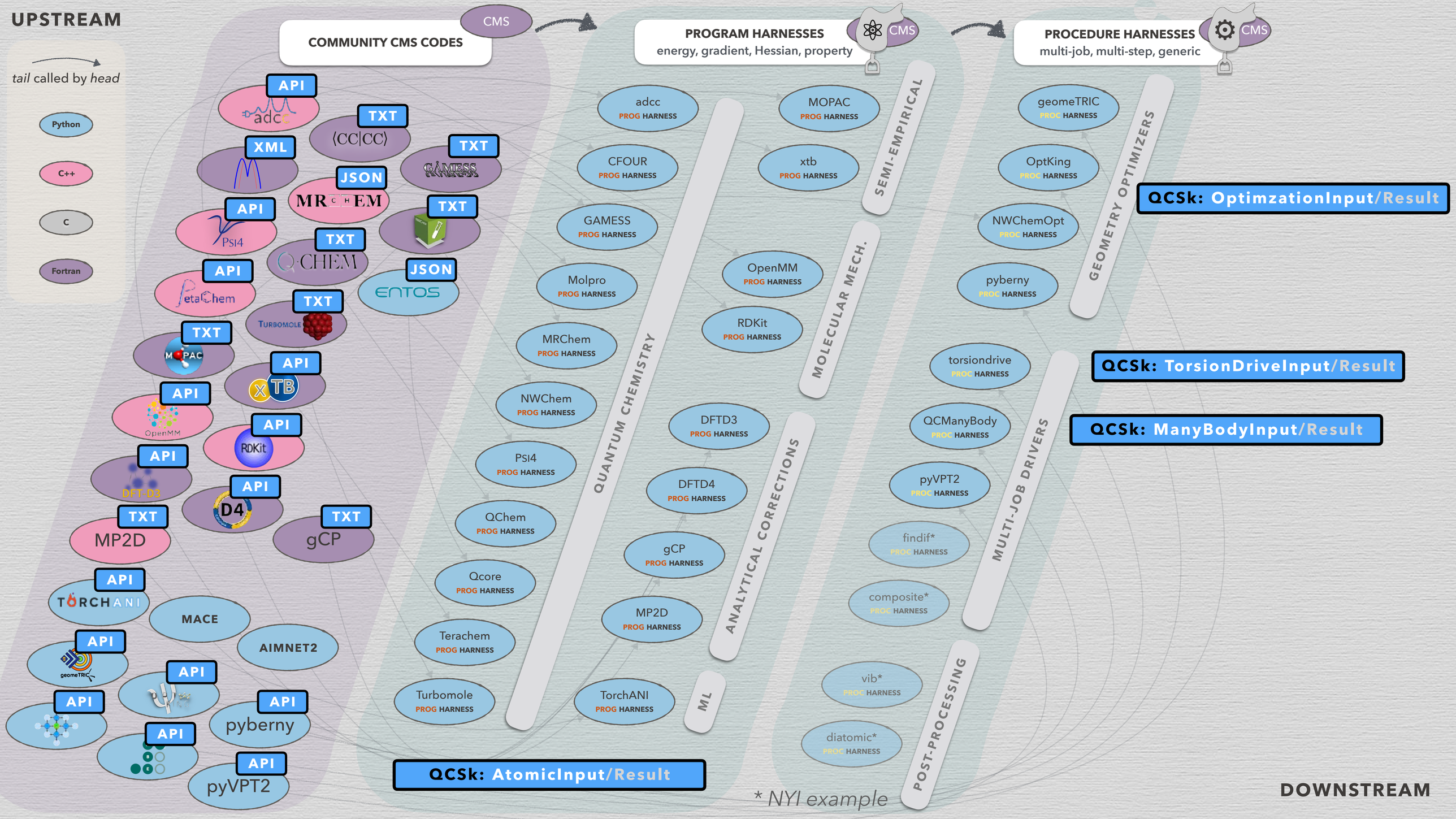


QCEngine: INTERNAL MAP

github.com/MolSSI/QCEngine



- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.


UPSTREAM
tail called by head
Python
C++
C
Fortran
COMMUNITY CMS CODES
CMS
PROGRAM HARNESSES
energy, gradient, Hessian, property
CMS
PROCEDURE HARNESSES
multi-job, multi-step, generic
CMS
API
adcc
TXT
CC
XML
TXT
GAMESS
JSON
MRCHEM
API
PSI4
TXT
TXT
Q-CHEM
JSON
ENTOS
API
PetaChem
TXT
TURBOMOLE
TXT
MOPAC
API
XTB
API
OpenMM
API
RDKit
API
DFT-D3
API
D4
TXT
MP2D
TXT
gCP
API
TORCHANI
MACE
AIMNET2
API
geomeTRIC
API
API
API
pyberny
API
API
pyVPT2
adcc
PROG HARNESS
CFOUR
PROG HARNESS
GAMESS
PROG HARNESS
Molpro
PROG HARNESS
MRChem
PROG HARNESS
NWChem
PROG HARNESS
PSI4
PROG HARNESS
QChem
PROG HARNESS
Qcore
PROG HARNESS
Terachem
PROG HARNESS
Turbomole
PROG HARNESS
QUANTUM CHEMISTRY
MOPAC
PROG HARNESS
xtb
PROG HARNESS
SEMI-EMPIRICAL
OpenMM
PROG HARNESS
RDKit
PROG HARNESS
MOLECULAR MECH.
DFTD3
PROG HARNESS
DFTD4
PROG HARNESS
gCP
PROG HARNESS
MP2D
PROG HARNESS
ANALYTICAL CORRECTIONS
TorchANI
PROG HARNESS
ML
QCSk: AtomicInput/Result
geomeTRIC
PROC HARNESS
OptKing
PROC HARNESS
NWChemOpt
PROC HARNESS
pyberny
PROC HARNESS
GEOMETRY OPTIMIZERS
QCSk: OptimzationInput/Result
torsiondrive
PROC HARNESS
QCManyBody
PROC HARNESS
pyVPT2
PROC HARNESS
findif*
PROC HARNESS
composite*
PROC HARNESS
MULTI-JOB DRIVERS
QCSk: TorsionDriveInput/Result
QCSk: ManyBodyInput/Result
vib*
PROC HARNESS
diatomic*
PROC HARNESS
POST-PROCESSING
* NYI example
DOWNSTREAM

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

Atom = analytical single-point

PSI4
C++/PYTHON LIBRARY

E · *G* ⋮ *H*

start
QC calculation
end

`class AtomicComputer ():`

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

MBE FD CBS Atom

POOL

I
QCSchema:
AtomicInput

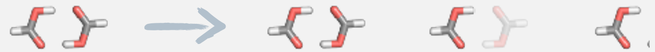
PSI4
C++/PYTHON LIBRARY

E · G ⋮ H

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis.
`method` unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil.
Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

`class CompositeComputer ():`

PLAN Separate **`method`** into method, basis, & extrapolations.
`molecule` unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

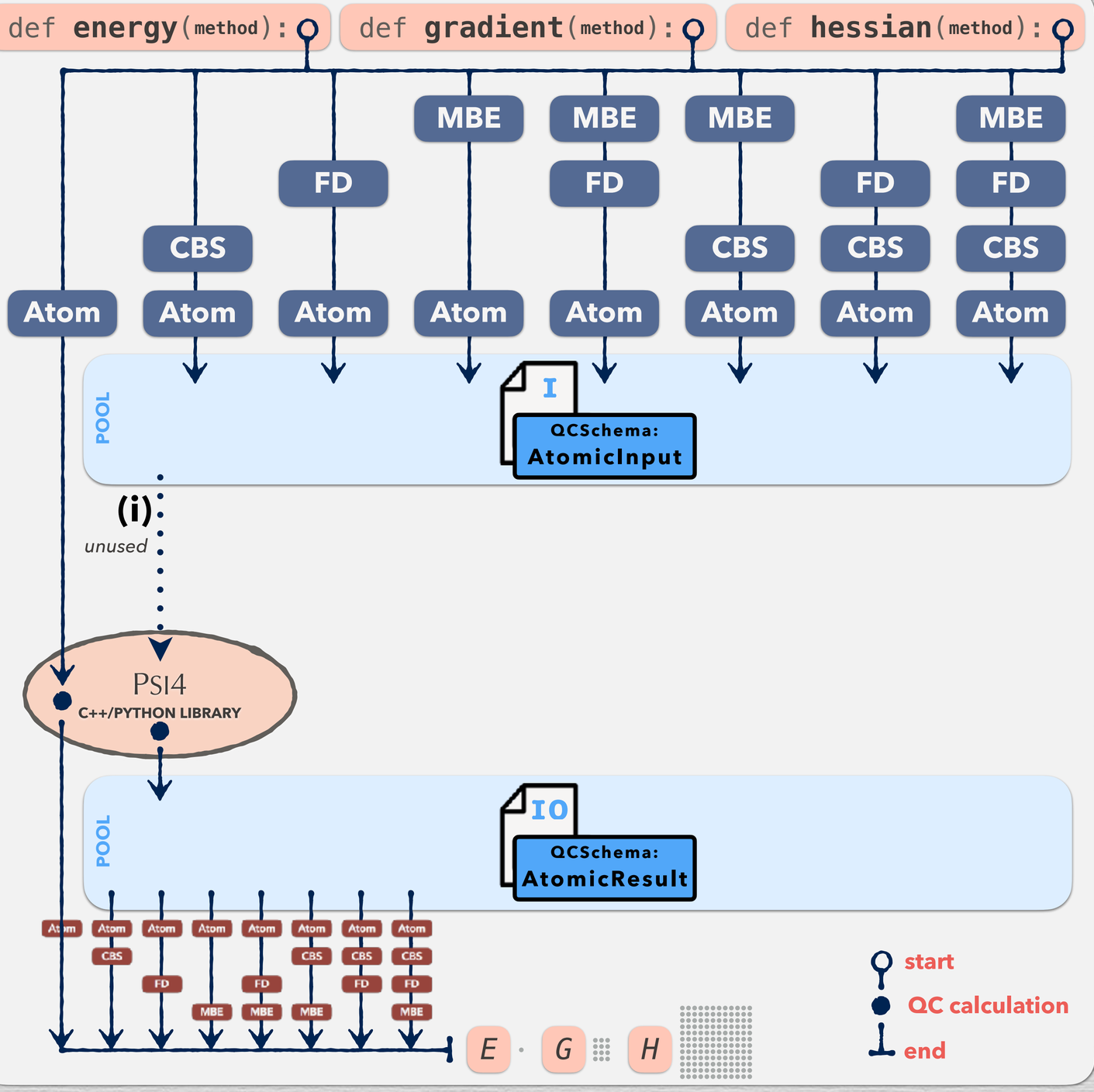
ASM Assemble extrapolations & total results from modelchems.

`class AtomicComputer ():`

PLAN **`molecule`** & **`method`** unchanged. `return qcschema` QCSchema

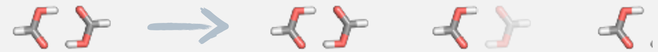
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

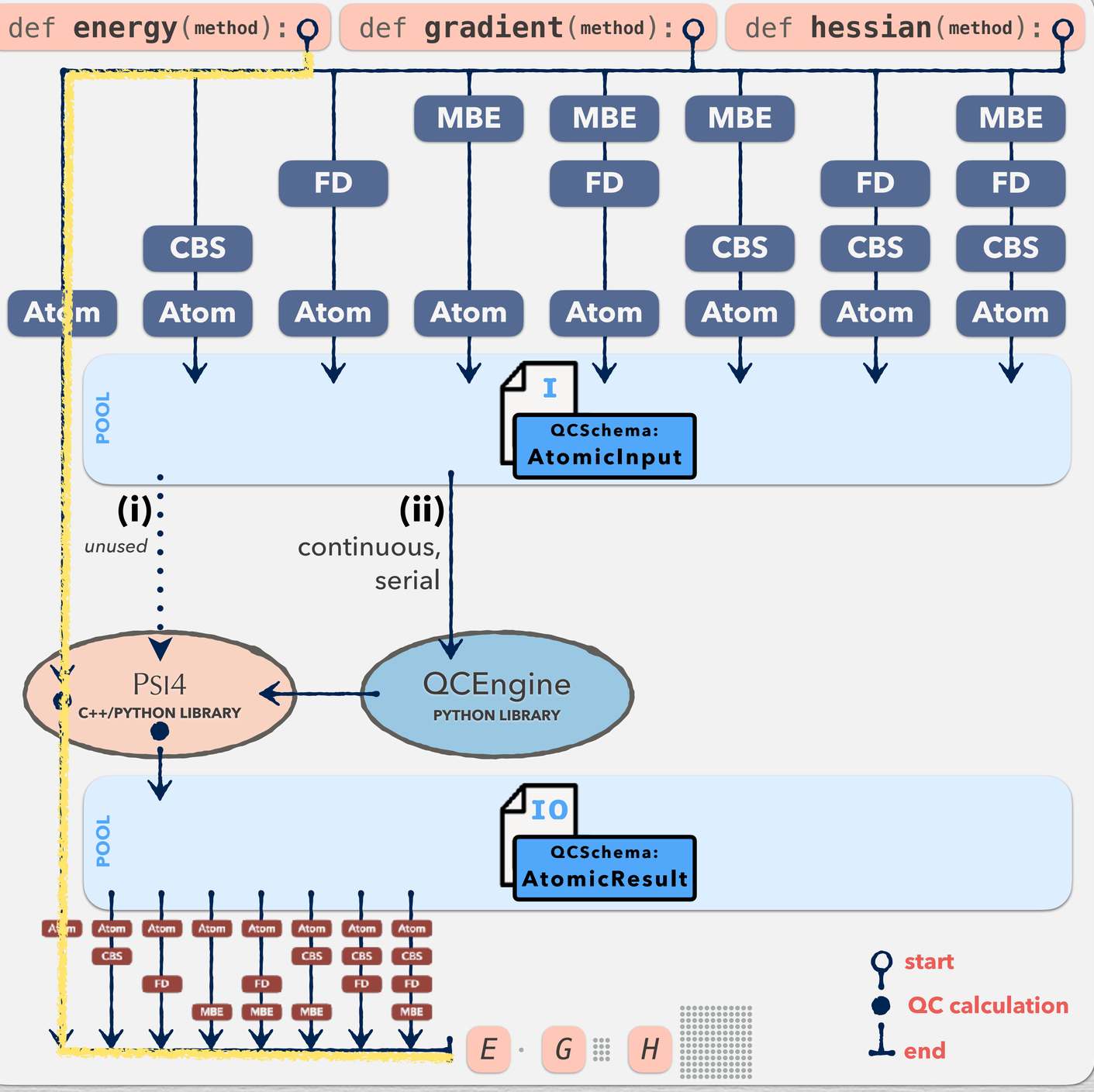
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

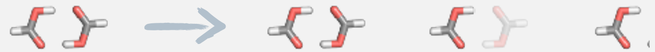
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

```
psi4.energy("mp2/cc-pvdz")
```

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

```
mp2/cc-pv[tq]z  →  MP2 TOTAL ENERGY/cc-pVTZ
                   MP2 TOTAL ENERGY/cc-pVQZ
for mc in modelchems: return qcschema
```

QCSchema

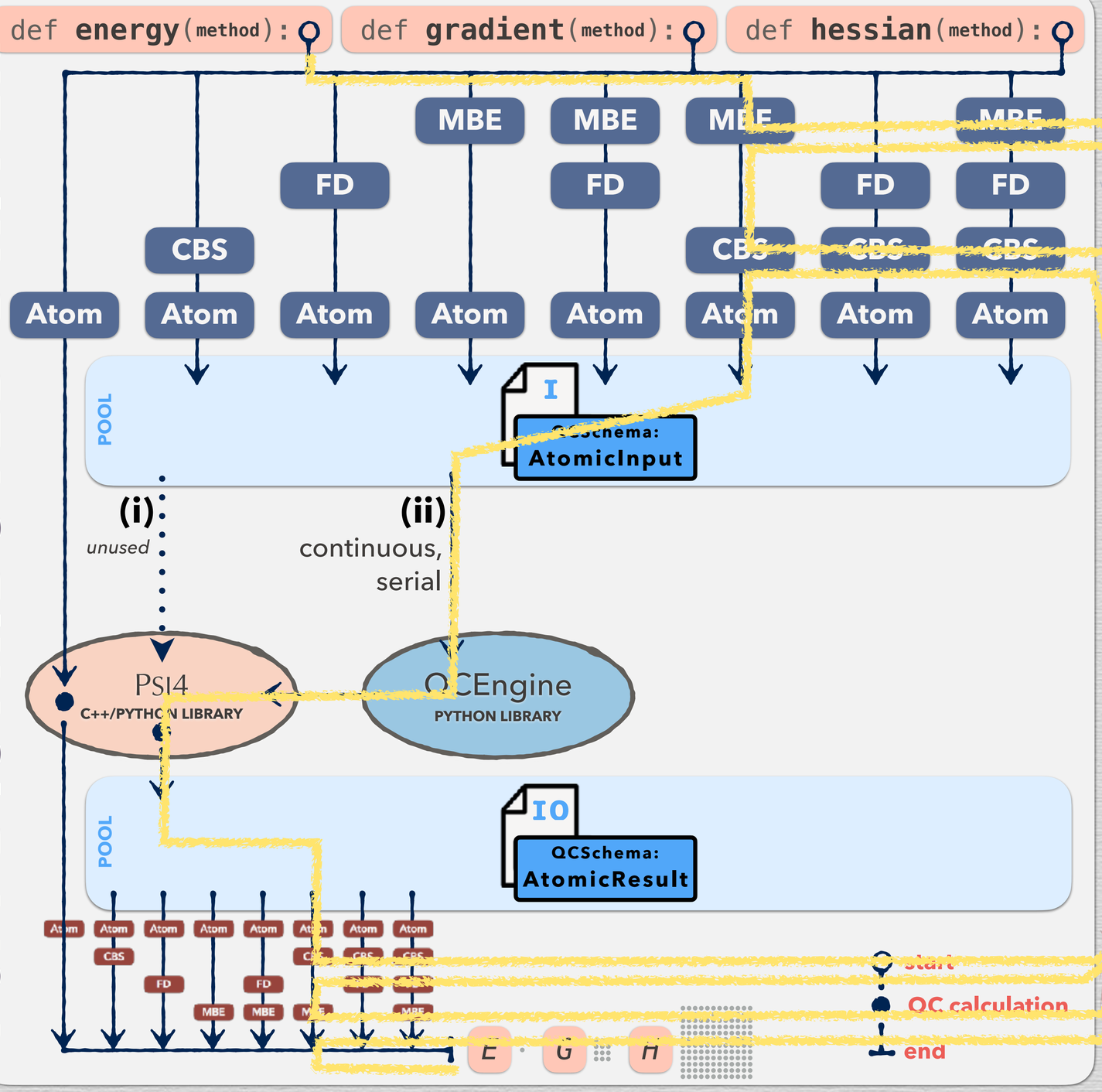
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class **ManyBodyComputer** ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.

ASM Assemble n-body & interaction results from fragments.

```
psi4.energy("mp2/cc-pvdz")
```

```
psi4.energy("mp2/cc-pv[dt]z", bsse_type="cp")
```

class **CompositeComputer** ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ` `MP2 TOTAL ENERGY/cc-pVQZ`

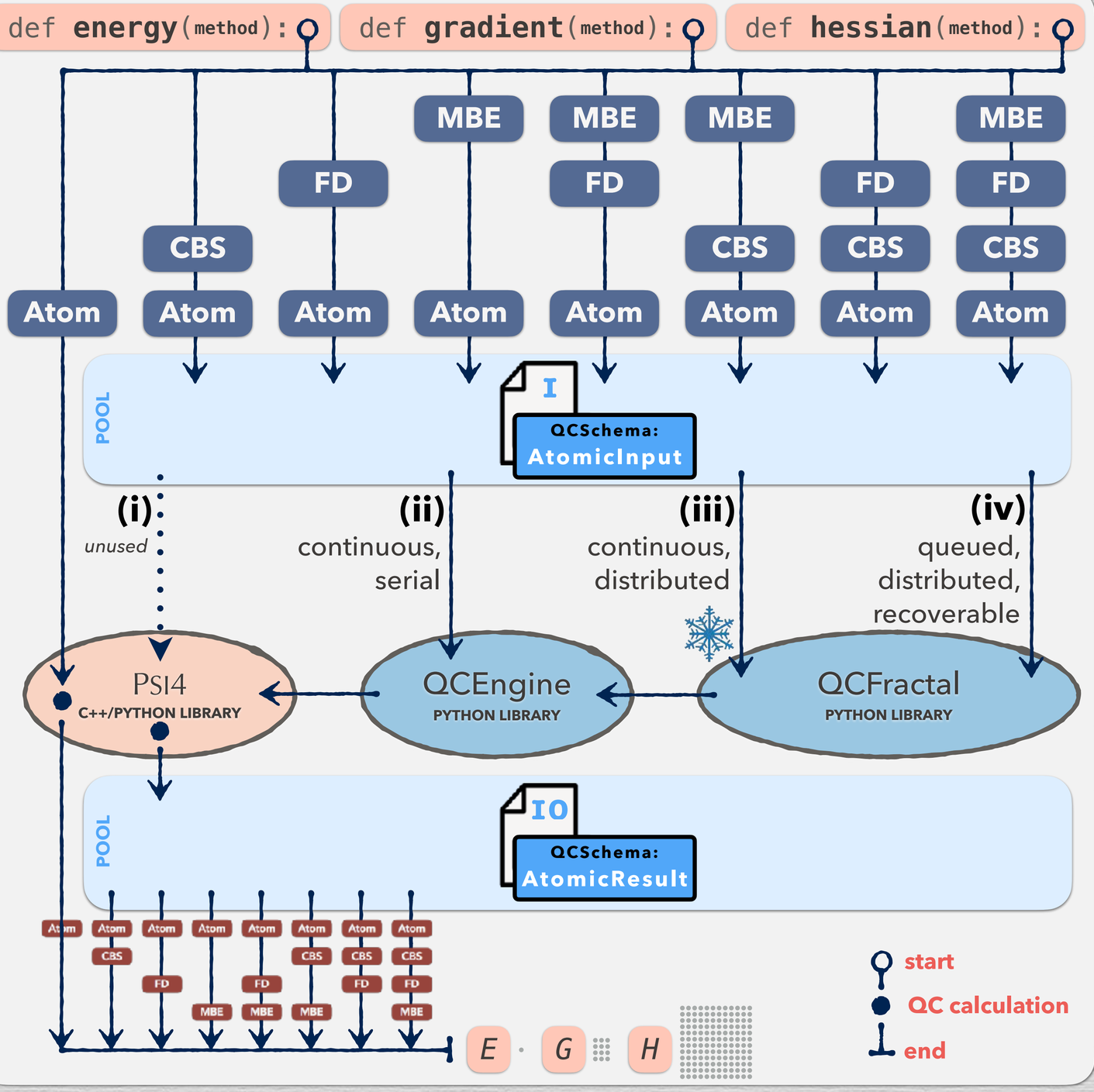
ASM Assemble extrapolations & total results from model chems.

class **AtomicComputer** ():

PLAN **molecule** & **method** unchanged. `return qcschema`

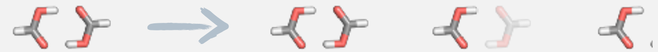
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

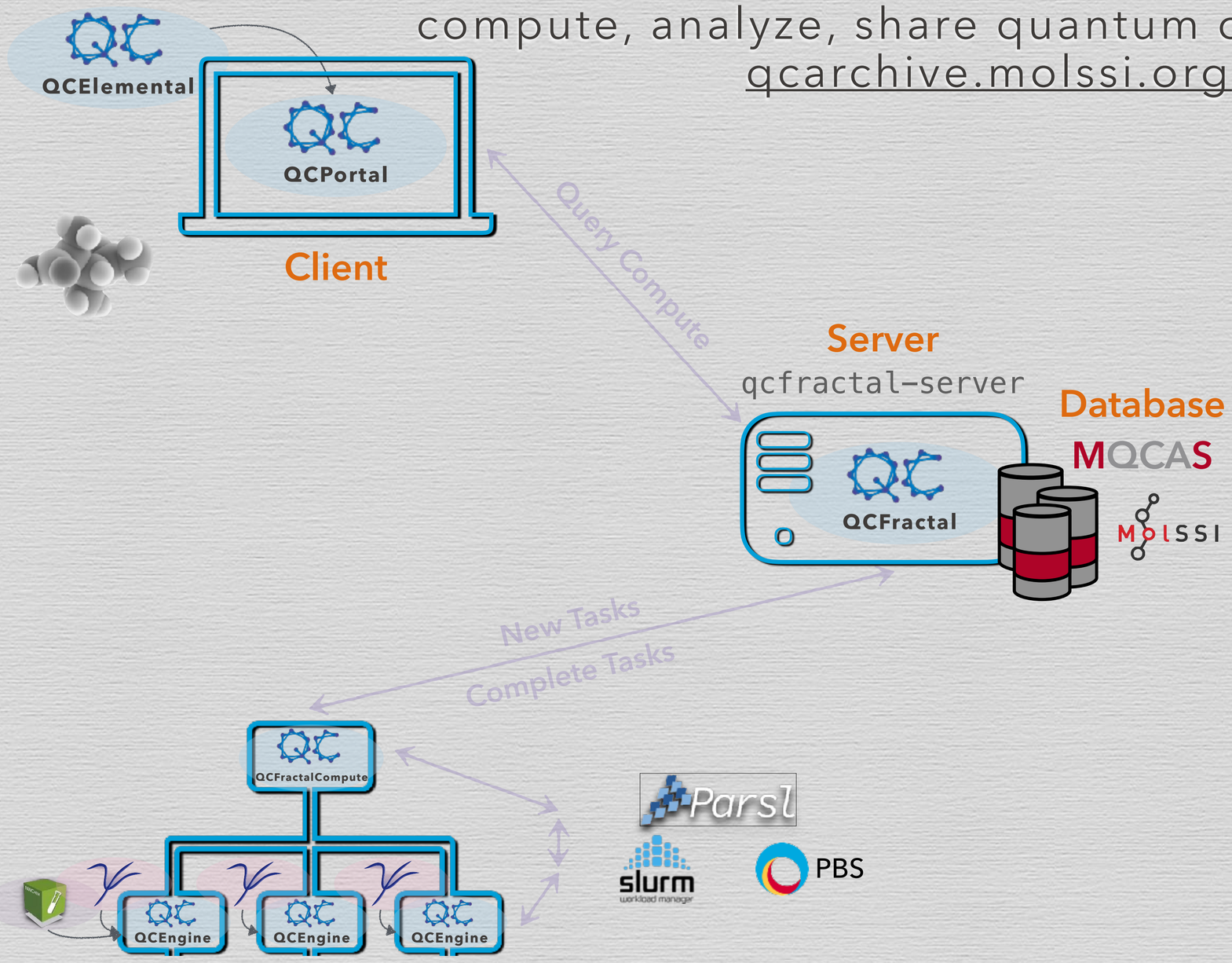
PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter Task Management

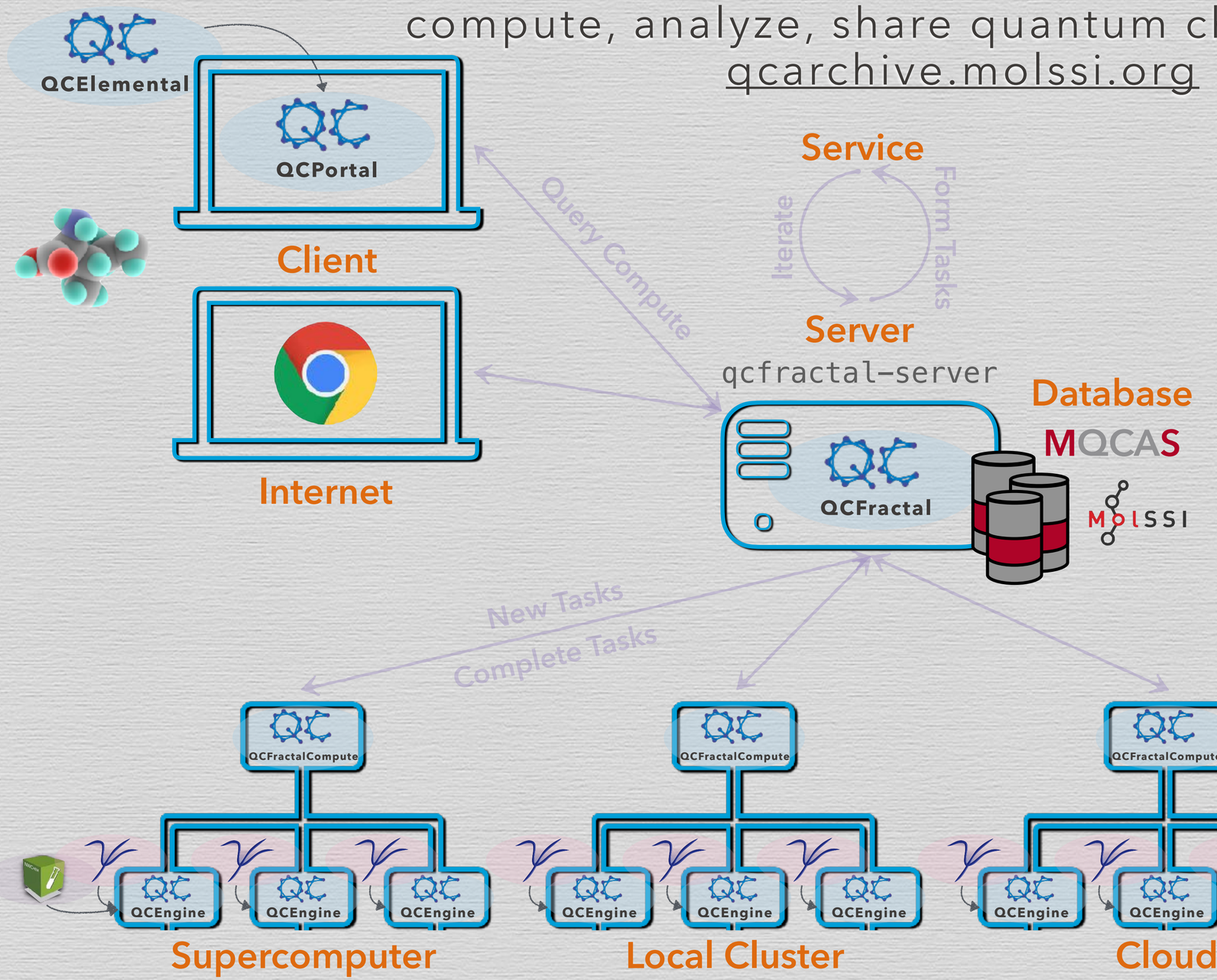
scheduler Task Scheduling

worker Compute Nodes

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

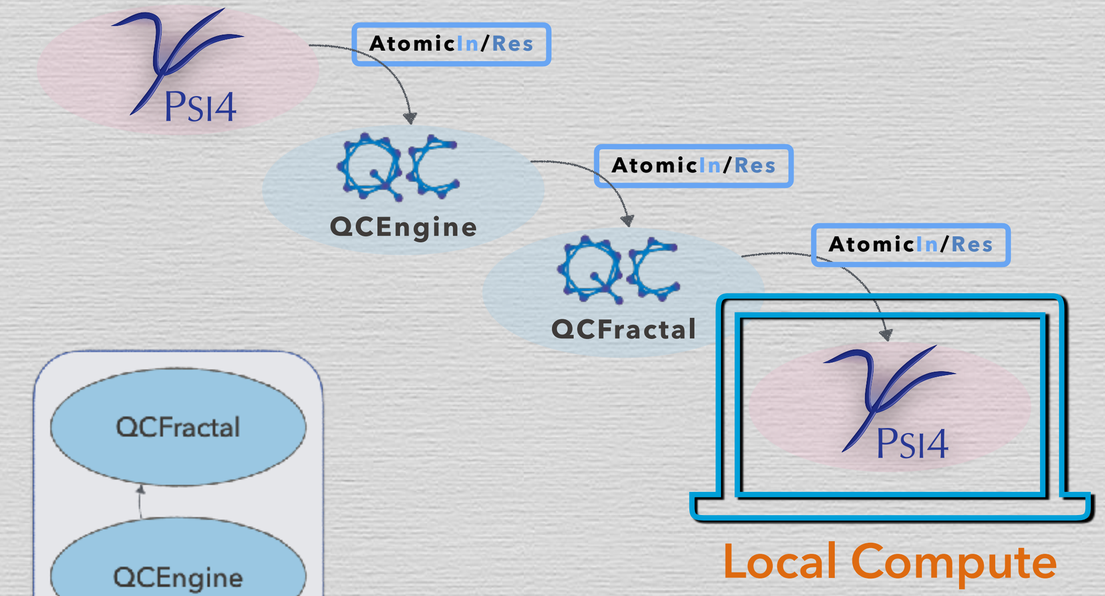
adapter Task Management

scheduler Task Scheduling

worker Compute Nodes

Psi4 DISTRIBUTED DRIVER + QCArchive

distributed, searchable, and error-resistant



(4) **Psi4** driver (core) runs single-point jobs

(5) **QCEngine** recieves jobs as **AtomicResult QCSchema**

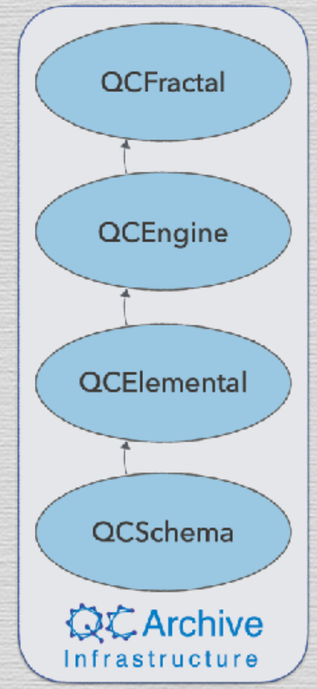
(3) **QCEngine** recieves jobs as **AtomicInput QCSchema**

(6) **QCFractal** stores jobs in DB & retrives on query

(2) **QCFractal** orchestrates job running among attached resources

(7) **Psi4** driver assembles results into usual API

(1) **Psi4** driver (post-proc. manager) plans single-point jobs from API input

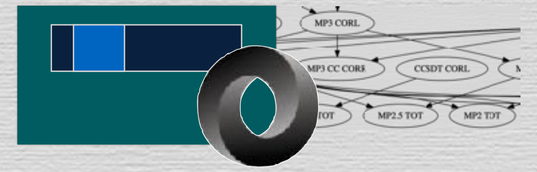


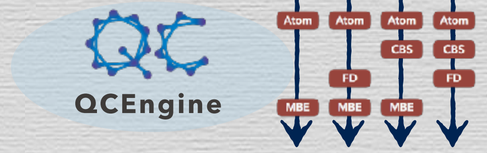
- **PAYOFF** Psi4 gets parallelism we don't have to manage.
- **PAYOFF** we had to become more robust to handle Psi4 calling Psi4.

★ Decoupling into more cohesive internal modules is progress, too.

★ Consume your own data standards to exercise them.

★ Find good separations of concern for software stack & build proj. around those missions.



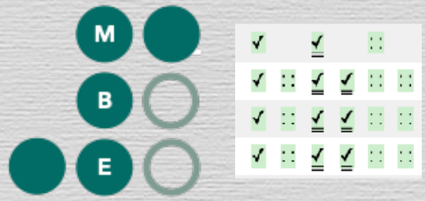


CONDA PACKAGING

DFTD3

Psi4

OPEN-SOURCE QUANTUM CHEMISTRY



A DFTD3 STORY

hold your interfaces and tests close

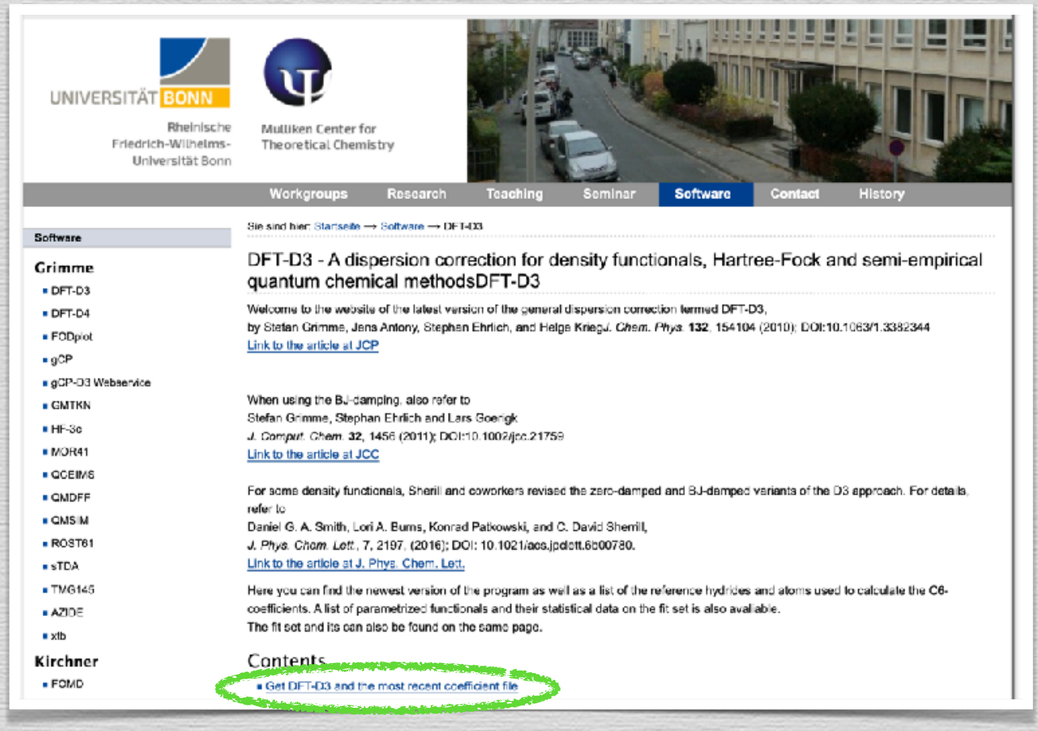
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT–D) for the 94 elements H–Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

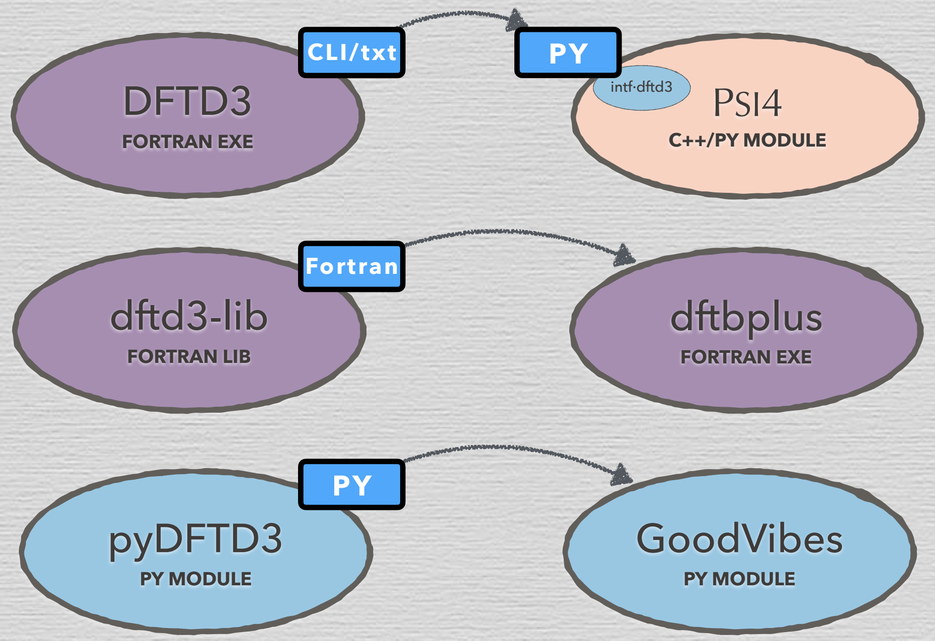
Stefan Grimme[a], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

Still better, the **CODE** for a Fortran executable



A DFTD3 STORY

hold your interfaces and tests close



… at least ten **TEN IMPLEMENTATIONS** were identified

Closed — Which dftd3 should one use? #4

zerothi opened this issue on Nov 18, 2021 · 10 comments

awvwgk commented on Nov 18, 2021 • edited — Member

Here is a quick search for `dftd3` versions which are currently around:

repo	version	license	note
http://mctc.uni-bonn.de/software/dft-d3	v3.2.0	GPL-1.0-or-later	reference implementation
https://github.com/awvwgk/simple-dftd3	v0.4.2	LGPL-3.0-or-later	this project
https://github.com/dftbplus/dftd3-lib	v0.10	GPL-1.0-or-later	CMake
https://github.com/ehermes/ased3	—	LGPL-3.0-or-later	ASE, f2py
https://github.com/pfnet-research/torch-dftd	v0.3.0	MIT	torch
https://github.com/cuanto/libdftd3	v3.1.1 (fork)	GPL-3.0-or-later	fork of dftd3-lib?, Python
https://github.com/cresset-group/dftd3	v3.2.0 (fork)	GPL-1.0-or-later	patched?
https://github.com/loriab/dftd3	v3.2.1 (fork)	GPL-1.0-or-later	CMake, patched
https://github.com/f3rmion/dftd3	v3.2.0 (fork)	GPL-1.0-or-later	patched
https://github.com/bobbypaton/pyDFTD3	v1.0.0	MIT	Python

OPEN-SOURCE projects proliferated until inevitably …

Which dftd3 should one use? #4

Closed — zerothi opened this issue on Nov 18, 2021 · 10 comments

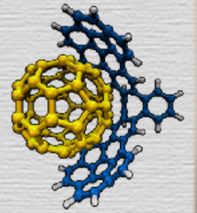
zerothi commented on Nov 18, 2021 • edited

Thanks for the work on this!

We are bit in a dead-lock now, since there exists 3 different libraries which all say do the same thing? :)
So which one should we support :)

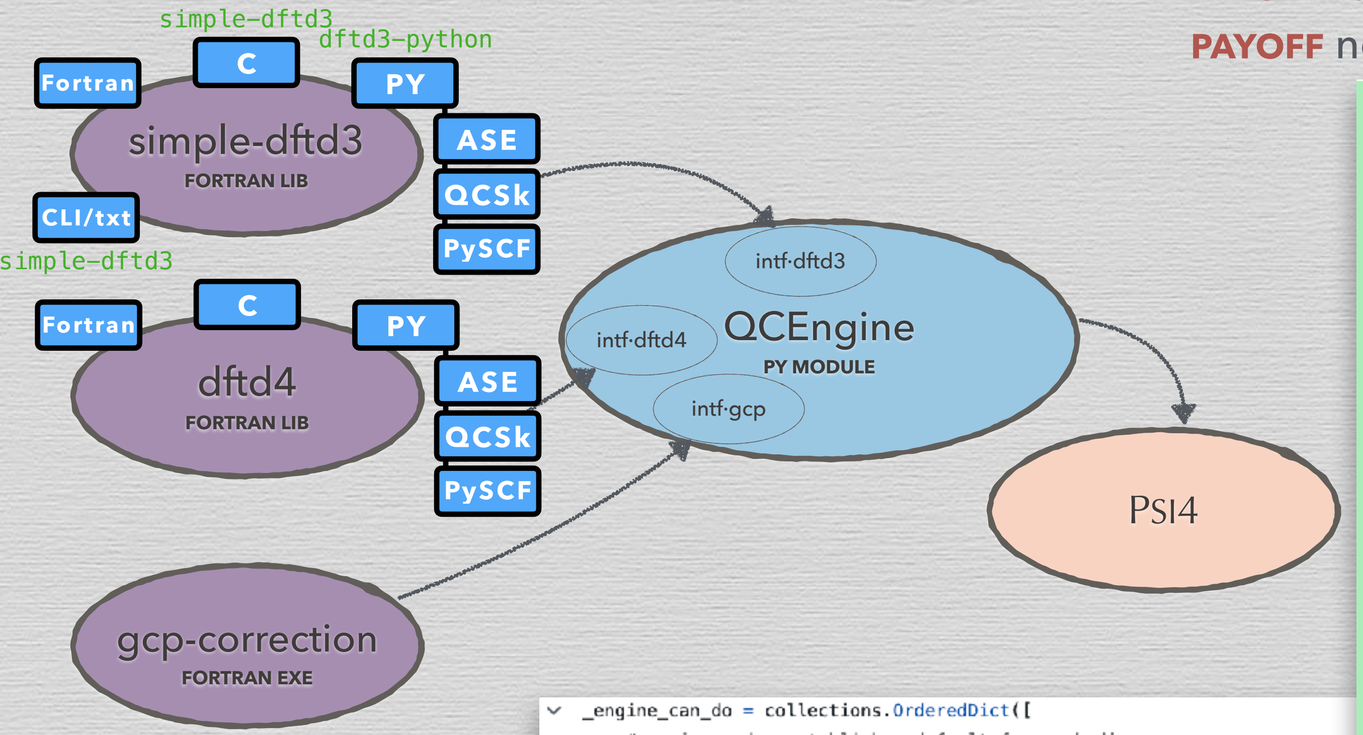
A DFTD3 STORY

★hold your interfaces and tests close



Sebastian Ehlert Holger Kruse
Bonn → MS Czech Acad. Sci.
→ Pending AI

RE-IMPLEMENTED upstream with abundant interfaces



PAYOFF Psi4 no longer maintains and packages forks of D3, D4, & gCP

PAYOFF new DFT capabilities easy to add

```
+ funcs.append({
+     "name": "R2SCAN3C",
+     "alias": ["R2SCAN-3C"],
+     "x_functionals": {
+         "MGGA_X_R2SCAN": {}
+     },
+     "c_functionals": {
+         "MGGA_C_R2SCAN": {}
+     },
+     "description": '    r2SCAN Meta-GGA based 3C composite method with a TZ basis set, gCP and D4\n',
+     "citation": '     S. Grimme,  A. Hansen,  S. Ehlert, J.-M. Mewes J. Chem. Phys. 154, 064103, 2021\n',
+     "doi": "10.1063/5.0040021",
+     "dispersion": {
+         "type": "d4bjeeqatm",
+         "params": {
+             'a1': 0.420,
+             'a2': 5.650,
+             's6': 1.000,
+             's8': 0.000,
+             's9': 2.000,
+             'ga': 2.000,
+             'gc': 1.000,
+         },
+     },
+ })
```

Adds r2SCAN hybrids; r2SCAN-3c and B97-3c #2842

Merged loriab merged 24 commits into psi4:master from hokru:r2scan on Dec 4, 2023

```
if self.engine in ["s-dftd3", 'dftd3', "mp2d", "dftd4"]:
    resi = AtomicInput(
        **{
            'driver': 'energy',
            'model': {
                'method': self.fctldash,
                'basis': '(auto)',
            },
            'keywords': {
                'level_hint': self.dashlevel,
                'params_tweaks': self.dashparams,
                'dashcoeff_supplement': self.dashcoeff_supplement,
                'pair_resolved': self.save_pairwise_disp,
                'apply_qcengine_aliases': True,  # for s-dftd3
                'verbose': 1,
            },
            'molecule': molecule.to_schema(dtype=2),
            'provenance': p4util.provenance_stamp(__name__),
        })
    jobrec = qcng.compute(
        resi,
        self.engine,
        raise_error=True,
        task_config={"scratch_directory": core.IOManager.shared_object().get_def
    dashd_part = float(jobrec.extras['qcvars']['DISPERSION CORRECTION ENERGY'])
```

```
_engine_can_do = collections.OrderedDict([
    # engine order establishes default for each disp
    ("libdisp",  ["d1", "d2",
    ("s-dftd3",  [           "d3zero2b", "d3bj2b", "d3mzero2b", "d3mbj2b", "d3zeroatm", "d3bjatm", "d3mzeroatm", "d
    ("dftd3",    [     "d2", "d3zero2b", "d3bj2b", "d3mzero2b", "d3mbj2b",
    ("nl",       [
    ("mp2d",     [
    ("dftd4",    [
    ("mctc-gcp", [
    ("gcp",      [
]) # yapf: disable
```

A DFTD3 STORY

smoothly replacing a much-used external feature

GUIDED PROGRESSION THROUGH ALTERNATE IMPLEMENTATIONS

pre-v1.0 classic dftd3 only

v1.7 s-dftd3 preferred; classic through:

```
energy("b3lyp-d3", engine="dftd3")
```

v1.9 s-dftd3 preferred; classic deprecated but runs: *(Note: apparently I forgot the warning, but this is the usual progression.)*

```
FutureWarning("Using `engine='dftd3'` instead of `engine='s-dftd3'` is deprecated, and as soon as 1.10 it will stop working")
```

v1.10 s-dftd3 only; classic errors:

```
UpgradeHelper("Using `engine='dftd3'` instead of `engine='s-dftd3'` is obsolete as of 1.10. Allow ... installing dftd3-python")
```

someday remove classic `UpgradeHelper` so errors unhelpfully

QCEngine soon remove classic harness

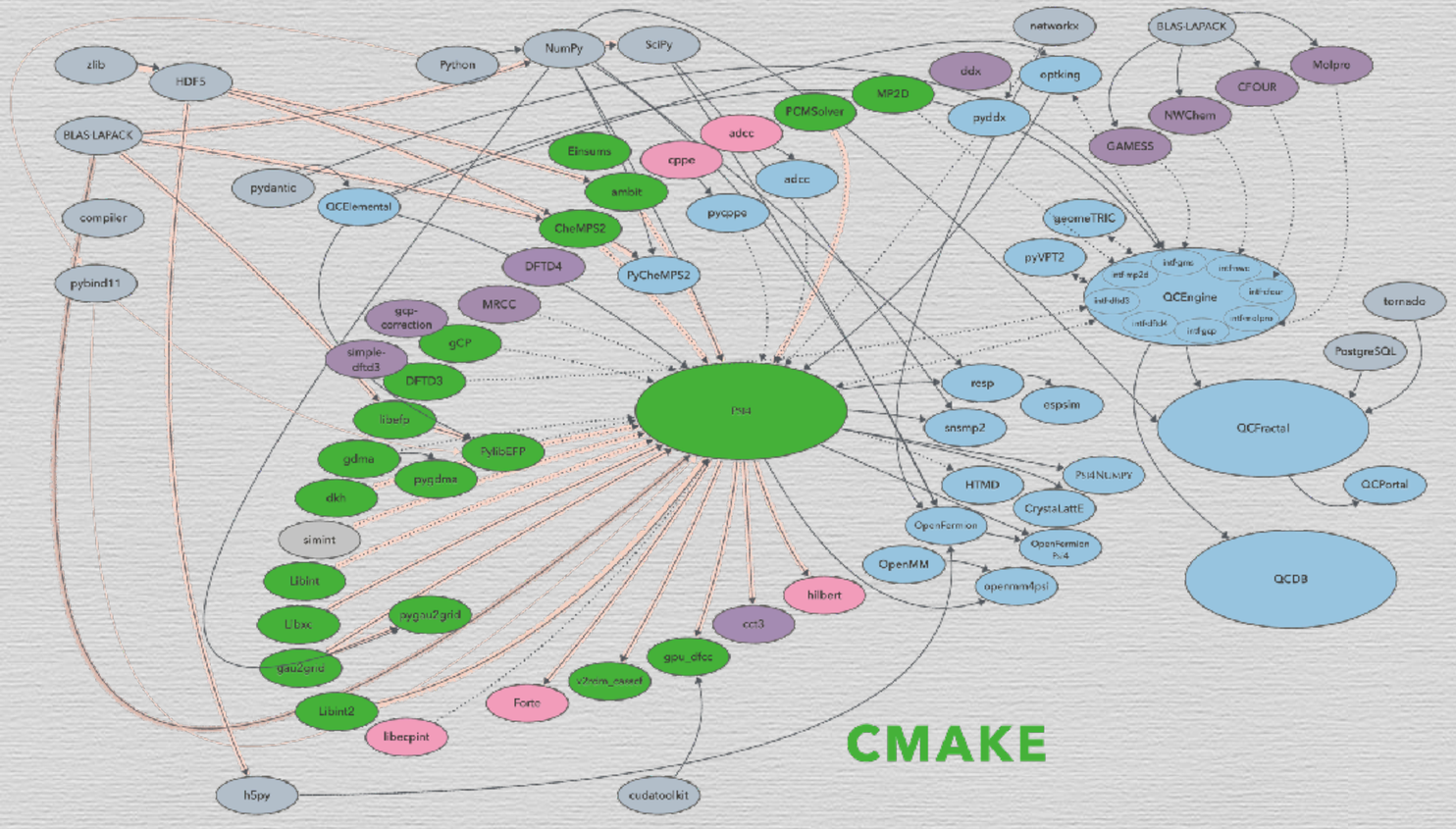
★ new syntax advice is more helpful at runtime than in docs. a grace period is a bonus

- Dispersion switches implementation w/i function call, and optimizers are similar: `optimize("b3lyp", engine="geometric|optking")`
- Alternate impl. of deeper functionality use keywords: `set integrals_package libint2|simint`, `set orbital_optimizer_package openorbitaloptimizer|internal`

```
#ifdef USING_OpenOrbitalOptimizer
        /*- Orbital optimizer package to use for SAD guess. -*/
        options.add_str("ORBITAL_OPTIMIZER_PACKAGE", "OPENORBITALOPTIMIZER", "OPENORBITALOPTIMIZER INTERNAL");
#else
        /*- Orbital optimizer package to use for SAD guess. -*/
        options.add_str("ORBITAL_OPTIMIZER_PACKAGE", "INTERNAL", "INTERNAL");
#endif
```

CONDA-FORGE PACKAGING

https://anaconda.org/



- **CONDA** is a cross-platform, cross-language binary package manager supporting community packaging & focused on the scientific stack
- **MKL** LAPACK, compilers, CUDA runtimes are among the production-quality tools available
- **CMAKE** has been contributed to a number of community projects to facilitate integration with Psi4 (and add Windows)
- **THIRTY** ecosystem projects are or have been packaged by Psi4
- **STARTING** 2015 Psi4 available from a custom `-c psi4` channel. Since 2023, have recc. `conda install psi4 -c conda-forge`
- **CONDA-FORGE** is increasingly (**30K**) the home for downstream & dep. projects, so for robust environment solving, we've transitioned to c-f
- **PRESENTLY** Linux (Intel & ARM), Windows, and Mac (Intel & Silicon) Psi4 ecosystems available from conda

CONDA-FORGE PACKAGING

https://anaconda.org/

ANACONDA CLOUD

PSI4 CONDA INSTALL

recipe libxc-c

- linux-ppc64le v7.0.0
- osx-64 v7.0.0
- osx-arm64 v7.0.0
- linux-64 v7.0.0
- linux-aarch64 v7.0.0
- win-64 v7.0.0

PACKAGING

- **CONDA** is a cross-platform, cross-language binary package manager supporting community packaging & focused on the scientific stack
- **MKL** LAPACK, compilers, CUDA runtimes are among the production-quality tools available
- **CMAKE** has been contributed to a number of community projects to facilitate integration with Psi4 (and add Windows)
- **THIRTY** ecosystem projects are or have been packaged by Psi4
- **STARTING** 2015 Psi4 available from a custom `-c psi4` channel. Since 2023, have recc. `conda install psi4 -c conda-forge`
- **CONDA-FORGE** is increasingly (**30K**) the home for downstream & dep. projects, so for robust environment solving, we've transitioned to c-f
- **PRESENTLY** Linux (Intel & ARM), Windows, and Mac (Intel & Silicon) Psi4 ecosystems available from conda

CONDA-FORGE RECIPE: CPPE

a Py extension module

recipe/meta.yaml

```yaml
package:
  name: cppe
  version: "0.3.2"

source:
  url: GH:maxscheurer/cppe/archive/v0.3.2.tar.gz
  patches:
    - 0001-update-math-call.patch

build:
  number: 3
  run_exports:
    - {{ pin_subpackage('cppe', max_pin='x.x') }}

requirements:
  build:
    - {{ stdlib("c") }}
    - {{ compiler('cxx') }}
    - python  # [build_platform != target_platform]
    - cmake
    - eigen
  host:
    - pybind11
    - python
  run:
    - python

test:
  requires:
    - pytest
  imports:
    - cppe
  commands:
    - test -f $SP_DIR/cppe/pycppe*.so  # [not win]
    - pytest --pyargs cppe

about:
  home: https://github.com/maxscheurer/cppe
  license: LGPL-3.0-or-later
  summary: "C++/Py library for polarizable embedding"

extra:
  recipe-maintainers:
    - robertodr
    - maxscheurer
```

- A conda **RECIPE** contains project meta-data and build directions. It's similar to a Homebrew formula (Linux/Mac) or a Fedora Spec file (Linux).
- To begin, make a PR with these files to **CONDA-FORGE/STAGED-RECIPES** . Packages will build in CI, so iterate until all passing, then request review.
- Once merged, you (and any co-**MAINTAINERS**) will have full permissions on your own project repository.
- **META** defines source, patches, expected API compat., build tools, build-time and run-time deps, anti-cipated files, testing, and licensing.
- **BUILD** scripts build & install for Unix & Windows with many defined variables.
- Multiple **OUTPUTS** (e.g., C, Fortran, Py) possible from one build.
- Everything is done in **GITHUB CI**, so arch are often cross-compiled.

recipe/build.sh

```
cmake \
  -S "${SRC_DIR}" \
  -B build \
  -D CMAKE_INSTALL_PREFIX:PATH="${PREFIX}" \
  -D Python_EXECUTABLE:STRING="${PYTHON}" \
  -D PYMOD_INSTALL_FULLDIR:PATH="${SP_DIR#$PREFIX/}/cppe"

cmake --build build --target install
```

recipe/bld.bat

```
cmake ^
  -S "%SRC_DIR%" ^
  -B build ^
  -D CMAKE_INSTALL_PREFIX:PATH="%PREFIX%" ^
  -D CMAKE_CXX_COMPILER:STRING=clang-cl ^
  -D CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS:BOOL=ON ^
  -D Python_EXECUTABLE:STRING="%PYTHON%" ^
  -D PYMOD_INSTALL_FULLDIR:PATH="Lib\site-packages\cppe"
if errorlevel 1 exit 1

cmake --build build --config Release --target install
if errorlevel 1 exit 1
```

CONDA-FORGE RECIPE: CPPE

a Py extension module

CONDA-FORGE BUILD MATRIX & VERSION PINNINGS

conda-forge-pinning-feedstock / recipe / conda_build_config.yaml

Code | Blame | 983 lines (965 loc) · 16.3 KB ·

```
23    cxx_compiler:
24      - gxx                   # [linux]
25      - clangxx               # [osx]
26      - vs2022                # [win]
27    cxx_compiler_version:     # [unix]
28      - 13                    # [linux]
29      - 18                    # [osx]
30      - 12                    # [linux and ppc64le and
31      - 13                    # [linux and not ppc64le
                  •••
189   blas_impl:
190     - openblas
191     - mkl           # [x86 or x86_64]
192     - blis          # [x86 or x86_64]
                  •••
813   pybind11_abi:
814     - 4
815   python:
816     # part of a zip_keys: python, is_python_min
817     - 3.9.* *_cpython
818     - 3.10.* *_cpython
819     - 3.11.* *_cpython
820     - 3.12.* *_cpython
821   python_impl:
822     - cpython
                  •••
980   zlib_ng:
981     - '2.2'
982   zstd:
983     - '1.5'
```

recipes can override as needed

- A conda **RECIPE** contains project meta-data and build directions. It's similar to a Homebrew formula (Linux/Mac) or a Fedora Spec file (Linux).
- To begin, make a PR with these files to **CONDA-FORGE/STAGED-RECIPES** . Packages will build in CI, so iterate until all passing, then request review.
- Once merged, you (and any co-**MAINTAINERS**) will have full permissions on your own project repository.
- **META** defines source, patches, expected API compat., build tools, build-time and run-time deps, anti-cipated files, testing, and licensing.
- **BUILD** scripts build & install for Unix & Windows with many defined variables.
- Multiple **OUTPUTS** (e.g., C, Fortran, Py) possible from one build.
- Everything is done in **GITHUB CI**, so arch are often cross-compiled.

recipe/build.sh

```
cmake \
  -S "${SRC_DIR}" \
  -B build \
  -D CMAKE_INSTALL_PREFIX:PATH="${PREFIX}" \
  -D Python_EXECUTABLE:STRING="${PYTHON}" \
  -D PYMOD_INSTALL_FULLDIR:PATH="${SP_DIR#$PREFIX/}/cppe"

cmake --build build --target install
```

recipe/bld.bat

```
cmake ^
  -S "%SRC_DIR%" ^
  -B build ^
  -D CMAKE_INSTALL_PREFIX:PATH="%PREFIX%" ^
  -D CMAKE_CXX_COMPILER:STRING=clang-cl ^
  -D CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS:BOOL=ON ^
  -D Python_EXECUTABLE:STRING="%PYTHON%" ^
  -D PYMOD_INSTALL_FULLDIR:PATH="Lib\site-packages\cppe"
if errorlevel 1 exit 1

cmake --build build --config Release --target install
if errorlevel 1 exit 1
```

CONDA-FORGE RECIPE: CPPE

a Py extension module

anaconda.org/conda-forge/cppe

conda-forge / packages / **cppe** 0.3.2

C++ and Python library for Polarizable Embedding calculations

copied from cf-staging / cppe

Conda | Files | Labels | Badges

License: LGPL-3.0-or-later
Home: https://github.com/maxscheurer/cppe
Development: https://github.com/maxscheurer/cppe
Documentation: https://cppe.readthedocs.io
242831 total downloads
Last upload: 2 months and 18 days ago

Installers

linux-64 v0.3.2
osx-64 v0.3.2
osx-arm64 v0.3.2
linux-aarch64 v0.3.2
win-64 v0.3.2
linux-ppc64le v0.3.2

conda install

To install this package run one of the following:

```
conda install conda-forge::cppe
```

- From this setup, you get many built packages with indiv. & cummulative download **COUNTS**.
- With **30K** packages, your deps are probably already present and maintained.
- You can **SHARE** tasks with co-maintainers.
- You get prodded by **BOTS** to update for new Py, HDF5, etc. versions and for new tech: compilers, architectures, GPU.
- When you need a **BUGFIX FAST**, it can be fully published within 30m (+ build time)
- Enter into a **COMMUNITY** that will push along standards and best-practices and bring deficiencies to your attention.
- **WHEN?** IMO, once you have the buildsys working on one arch, you start developing GHA CI for other arch. Once you have GHA CI for all arch and a small test, you start the packaging process.

PACKAGES

Size	Name
2.9 MB	linux-aarch64/cppe-0.3.2-py39hebc598f_3.conda
3.0 MB	linux-aarch64/cppe-0.3.2-py312h279dd22_3.conda
3.0 MB	linux-aarch64/cppe-0.3.2-py311hef841b3_3.conda
3.0 MB	linux-aarch64/cppe-0.3.2-py310h37ef1f8_3.conda
2.9 MB	win-64/cppe-0.3.2-py310h9d296e7_3.conda
3.0 MB	win-64/cppe-0.3.2-py313hf4e7ab2_3.conda
3.0 MB	linux-ppc64le/cppe-0.3.2-py39h59d6d83_3.conda
2.9 MB	win-64/cppe-0.3.2-py312ha5776e6_3.conda
3.0 MB	linux-ppc64le/cppe-0.3.2-py311hcf87a8c_3.conda
3.0 MB	linux-ppc64le/cppe-0.3.2-py312hce74920_3.conda
3.0 MB	linux-ppc64le/cppe-0.3.2-py313h7ffc702_3.conda
3.0 MB	linux-ppc64le/cppe-0.3.2-py310hae48ae8_3.conda
3.0 MB	win-64/cppe-0.3.2-py39hbe34ff7_3.conda
3.0 MB	win-64/cppe-0.3.2-py311h038952a_3.conda
2.9 MB	osx-arm64/cppe-0.3.2-py310h1577cc6_3.conda
3.0 MB	osx-64/cppe-0.3.2-py39h1959004_3.conda
2.8 MB	osx-arm64/cppe-0.3.2-py312h382e28c_3.conda
3.0 MB	linux-aarch64/cppe-0.3.2-py313h2f34184_3.conda
3.0 MB	osx-64/cppe-0.3.2-py312h1701139_3.conda
2.9 MB	osx-arm64/cppe-0.3.2-py311h1b28dfe_3.conda
3.0 MB	osx-64/cppe-0.3.2-py310h7a5d817_3.conda
3.0 MB	osx-64/cppe-0.3.2-py313h1c70d2e_3.conda
2.9 MB	osx-arm64/cppe-0.3.2-py313h403398f_3.conda
2.9 MB	osx-arm64/cppe-0.3.2-py39hf608f95_3.conda
3.0 MB	osx-64/cppe-0.3.2-py311hcdf574e_3.conda

6 architectures x 5 pythons

CONDA-FORGE RECIPE: CPPE

a Py extension module

anaconda.org/conda-forge/cppe

conda-forge / packages / **cppe** 0.3.2

C++ and Python library for Polarizable Embedding calculations

copied from cf-staging / cppe

Conda | Files | Labels | Badges

License: LGPL-3.0-or-later
Home: https://github.com/maxscheurer/cppe
Development: https://github.com/maxscheurer/cppe
Documentation: https://cppe.readthedocs.io
242831 total downloads
Last upload: 2 months and 18 days ago

Installers

linux-64 v0.3.2
osx-64 v0.3.2
osx-arm64 v0.3.2
linux-aarch64 v0.3.2
win-64 v0.3.2
linux-ppc64le v0.3.2

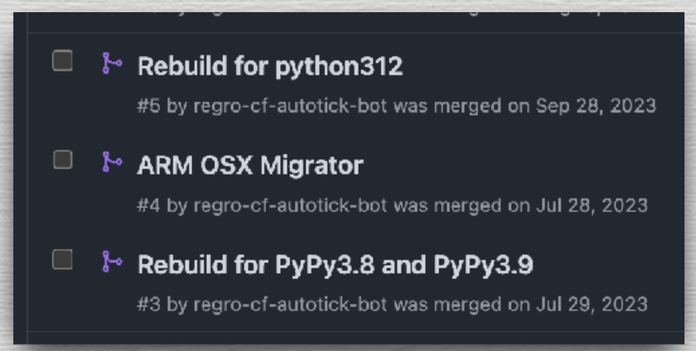
conda install

To install this package run one of the following:

```
conda install conda-forge::cppe
```

- From this setup, you get many built packages with indiv. & cummulative download **COUNTS**.
- With **30K** packages, your deps are probably already present and maintained.
- You can **SHARE** tasks with co-maintainers.
- You get prodded by **BOTS** to update for new Py, HDF5, etc. versions and for new tech: compilers, architectures, GPU.
- When you need a **BUGFIX FAST**, it can be fully published within 30m (+ build time)
- Enter into a **COMMUNITY** that will push along standards and best-practices and bring deficiencies to your attention.
- **WHEN?** IMO, once you have the buildsys working on one arch, you start developing GHA CI for other arch. Once you have GHA CI for all arch and a small test, you start the packaging process.

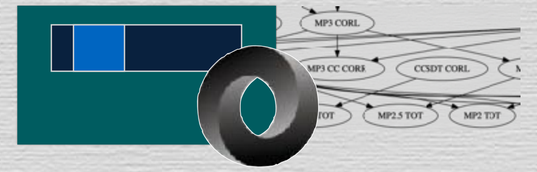
PR HISTORY

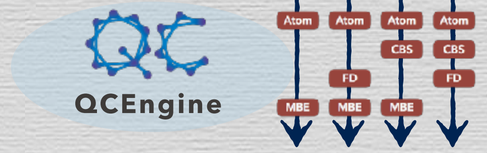


bot prompts for version or c-f updates

★Getting a production-quality binary to users for multiple architectures can be fairly easy with conda-forge.

★Joining a packaging community can prod software engineering aspects.



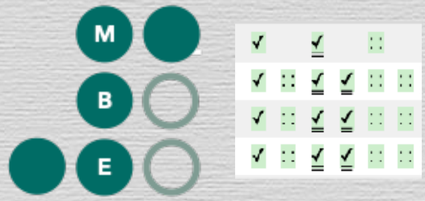


Psi4

OPEN-SOURCE QUANTUM CHEMISTRY

DISTRIBUTION

GDMA



GDMA: DISTRIBUTED MULTIPOLE ANALYSIS

gitlab.com/anthonyjs/gdma

ANALYSIS technique & code publication ~2005

Journal of Chemical Theory and Computation > Vol 1/Issue 6 > Article

ARTICLE | September 13, 2005

Distributed Multipole Analysis: Stability for Large Basis Sets

Anthony J. Stone

Anthony Stone: computer programs

- **GDMA 2.3.** GDMA is a Fortran program for performing Distributed Multipole Analysis of wavefunctions calculated using the Gaussian program system. That is, it calculates electric multipole moments at the atomic positions, or at other specified sites, that can give a very accurate representation of the electrostatic field of the molecule. It uses the formatted checkpoint file produced by the Gaussian program. Version 2.3 handles checkpoint files from any version of Gaussian. It can also use checkpoint files produced by the Psi4 program system. It is also possible to use GDMA with Dalton or NWChem or Psi4 as part of the CamCASP package. The GDMA package also includes the MULFIT program of Ferenczy et al.

CONVERSION to C library in 2016 for Psi4

andysim commented on Mar 24, 2016 — Member — Author

Because the code just calls GDM[illegible] which I turned into a function call instead of an executable, I think we should distributed the GDMA manual with the Psi4. I dumped the PDF into doc/external for now. Any objections?

andysim commented on Mar 25, 2016 — Member — Author

I added full Sphinx documentation for both the FCHK writer and for GDMA, in commit `f2aca76`, **@CDSherrill**. I heard back from Prof. Stone and he has kindly agreed to allow us to use GDMA in the way that we want to, so this PR is ready to go. In

C INTERFACE (+ I/O WORK)

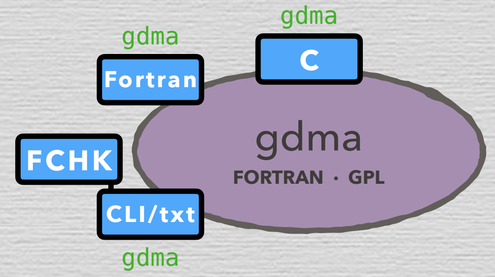
gdma / src / dma.f90

Code | Blame | 3100 lines (2832 loc) · 118 KB

```
328   CONTAINS
329   !-----------------------------------------------------------------
330   ! Some getter functions to access the DMA information from C
331   INTEGER(C_INT) FUNCTION get_nsites() BIND(c, name='get_nsites')
332       get_nsites = ns
333   END FUNCTION get_nsites
334
335   INTEGER(C_INT) FUNCTION get_order(site) BIND(c, name='get_order')
336       INTEGER(C_INT), VALUE, INTENT(IN) :: site
337       get_order = limit(site)
338   END FUNCTION get_order
339
340   REAL(C_DOUBLE) FUNCTION get_dma_value(site, addr) BIND(c, name='get_dma_value')
341       INTEGER(C_INT), VALUE, INTENT(IN) :: site, addr
342       get_dma_value = q(addr,site)
343   END FUNCTION get_dma_value
344
345   REAL(C_DOUBLE) FUNCTION get_tot_value(addr) BIND(c, name='get_tot_value')
346       INTEGER(C_INT), VALUE, INTENT(IN) :: addr
347       get_tot_value = qt(addr)
348   END FUNCTION get_tot_value
349   !-----------------------------------------------------------------  DMA
350
```

- **DEVELOPED** by A. J. Stone (Ret.)
- stand-alone Fortran **EXECUTABLE**
- **FCHK** interface
- GPL v3 **LICENSE**

Anthony Stone
Cambridge



Andy Simmonett
NIH → QC Ware

GDMA: DISTRIBUTED MULTIPOLE ANALYSIS

adapting established code to new use

noticed the **LICENSE** when updating to upstream

CONVERSION to Python module in 2023 for license

Update gdma to v2.3; convert to runtime #2968

Merged loriab merged 4 commits into psi4:master from loriab:pygdma on Aug 11, 2023

Conversation 4 · Commits 4 · Checks 0 · Files changed 14

loriab commented on May 24, 2023 • edited · Member

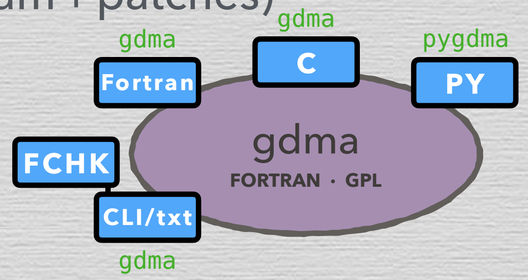
- [x] A perhaps obscure point is that psi hasn't been using exactly upstream GDMA, which is a Fortran executable. Instead, Andy Simmonett did lots of I/O conversion and turned it into a library and bound some functions to C so psi could extract results w/o parsing. That's great, but it's meant we've been fixed at upstream v2.2.06.
- [x] I was startled to realize gdma is GPL. For background, roughly speaking, GPL probably does adhere to psi4 if libgdma statically linked, might adhere if dynamically linked (FSF thinks yes; everyone else thinks no; result is that any GPL makes ppl nervous), doesn't adhere if dlopen'ed or runtime optional. Conda-package-wise, we've always dynamically linked to libgdma, and scouring my memory, there was a plan to have an alternate gdma- and chemps2-free conda build of psi4 to remove doubt; that never happened.

PYBIND11 INTERFACE

```
4   extern "C" int get_nsites();
5   extern "C" int get_order(int site);
6   extern "C" double get_dma_value(int site, int addr);
7   extern "C" double get_tot_value(int addr);
8
9   PYBIND11_MODULE(gdma, m) {
10
11      m.def("run_gdma", &run_gdma,
12            "Execute the GDMA analysis with input directives from the file named in the second argument (FCHK file
13      m.def("get_nsites", &get_nsites, "Return count of sites at which multipoles are placed in DMA analysis.");
14      m.def("get_order", &get_order, "Return multipole order for site in first argument (1-indexed).");
15      m.def("get_dma_value", &get_dma_value, "Return the spherical harmonic DMA for the site in the first argument
16      m.def("get_tot_value", &get_tot_value, "Return the total multipoles translated to the origin for component in
17      m.attr("__version__") = "@pygdma_VERSION@";
18  }
```

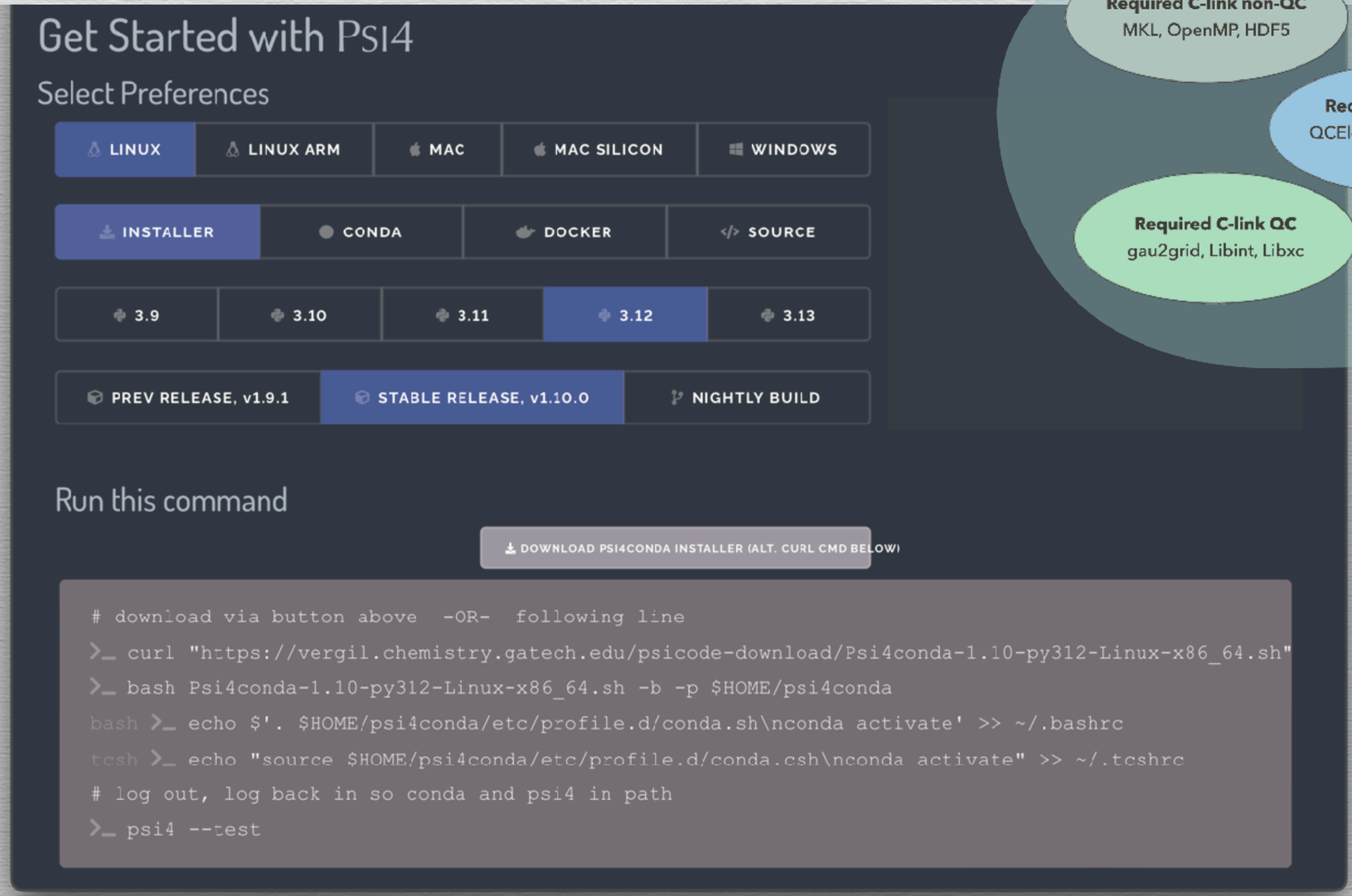
- **PAYOFF** library accesses results w/o parsing & output merged with Psi4's
- **PAYOFF** module defers enabling to runtime, avoiding entanglement of linking Psi4 with GPL (c.f. CheMPS2)
- **PAYOFF** conda-forge packaging (upstream + patches)

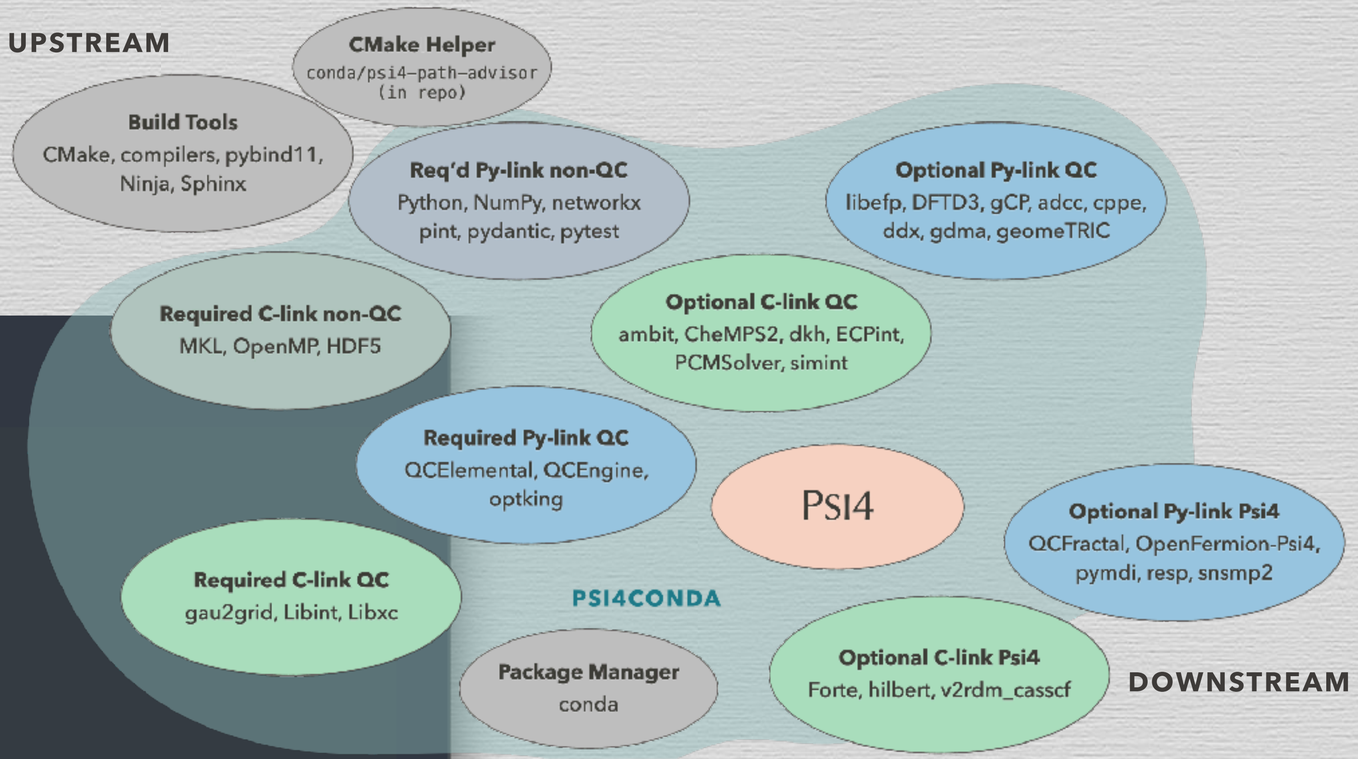
- ★ adapt established modular code with new interfaces
- ★ beware choice of license can limit uptake
- ★ defer loading to runtime for licensing and modularity

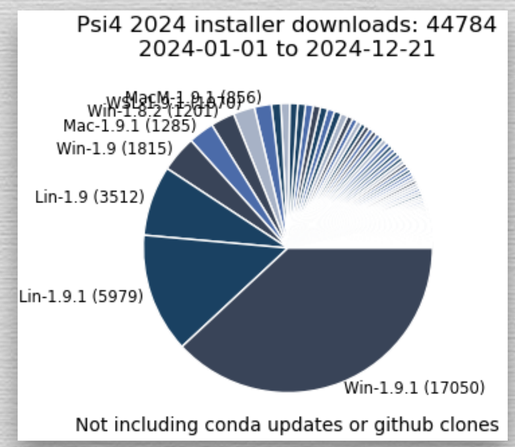


DISTRIBUTION

https://psicode.org/installs/latest







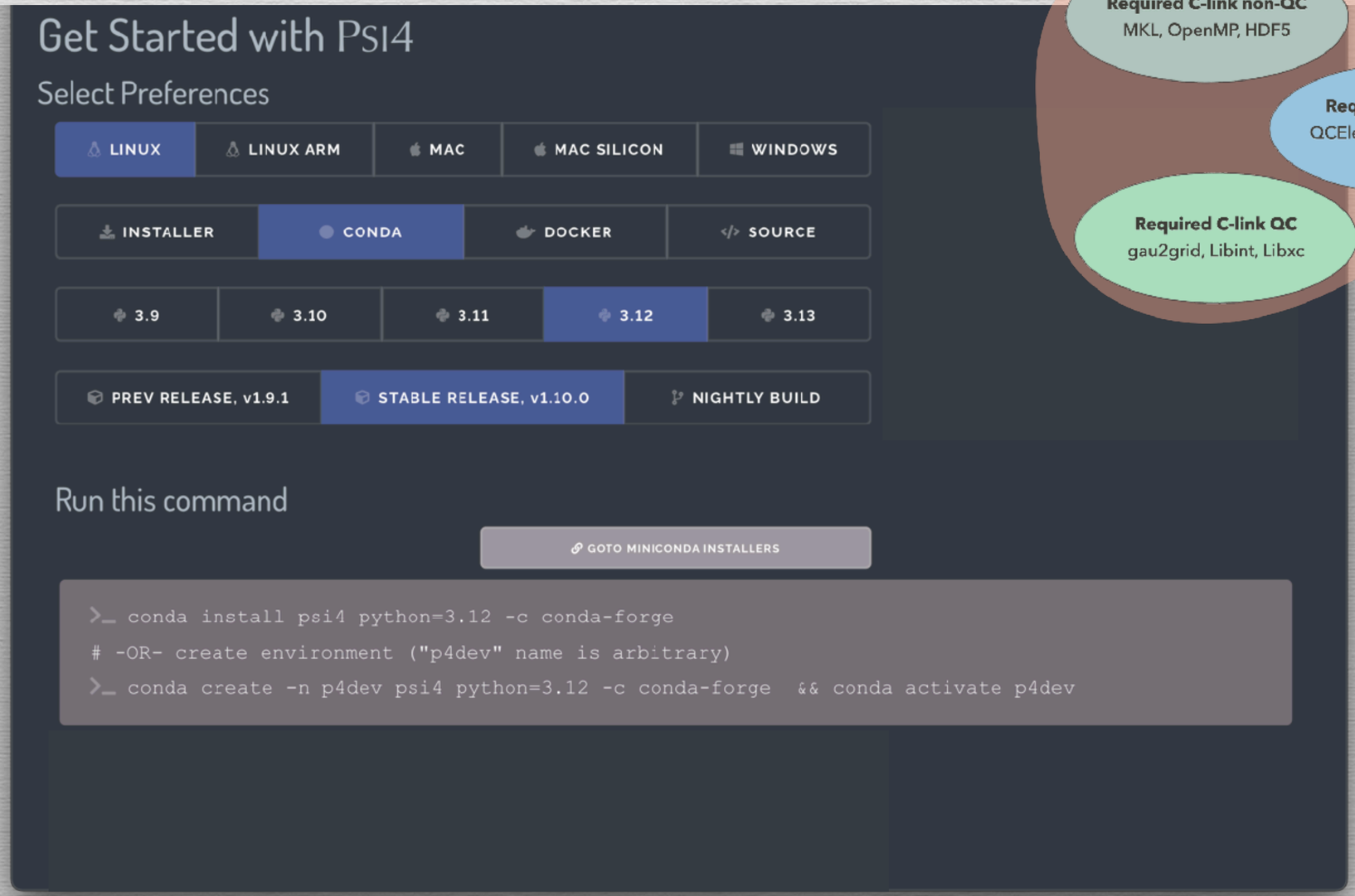
PY-WARY USERS: CONDA INSTALLER

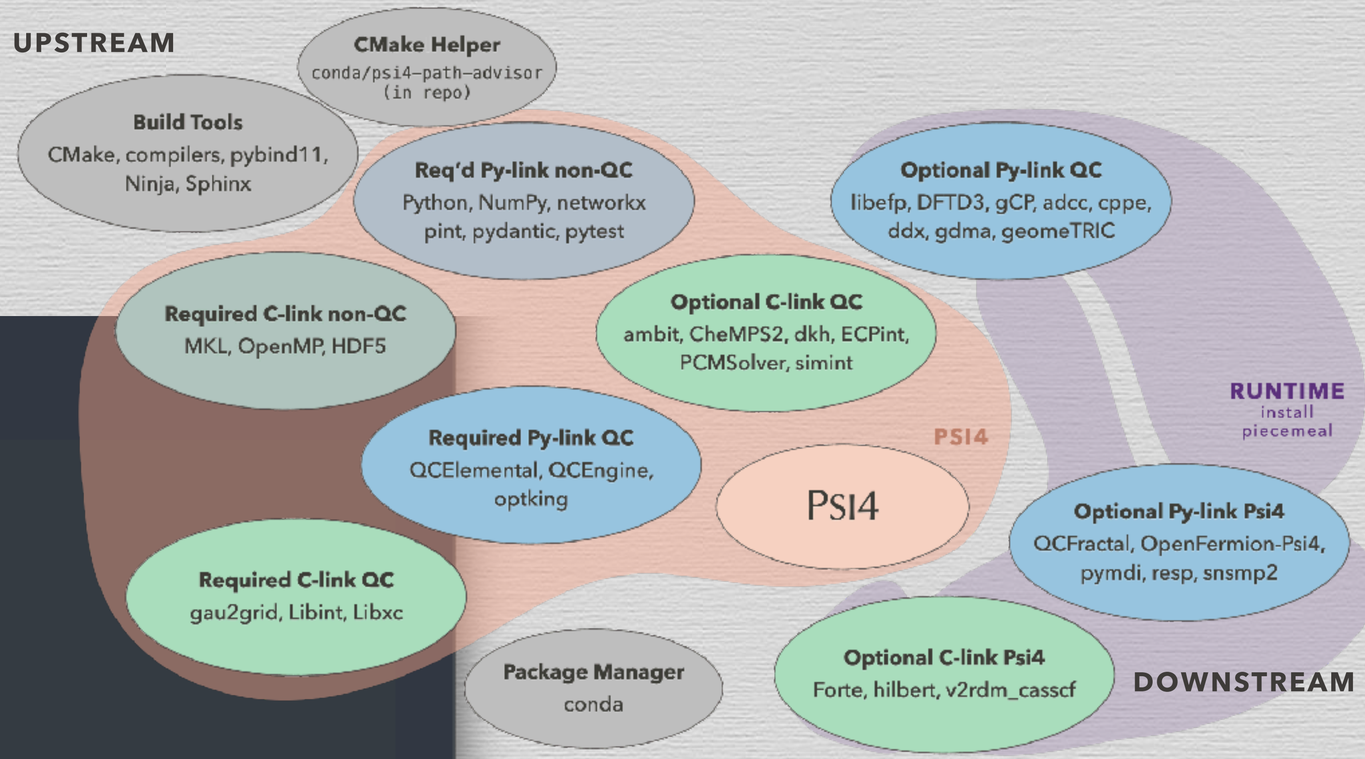
Psi4 · dependencies · add-ons

PSI4CONDA **633K** downloads

DISTRIBUTION

https://psicode.org/installs/latest





PY-FRIENDLY USERS:
CONDA PACKAGE

Psi4 · dependencies · add-ons

628K downloads

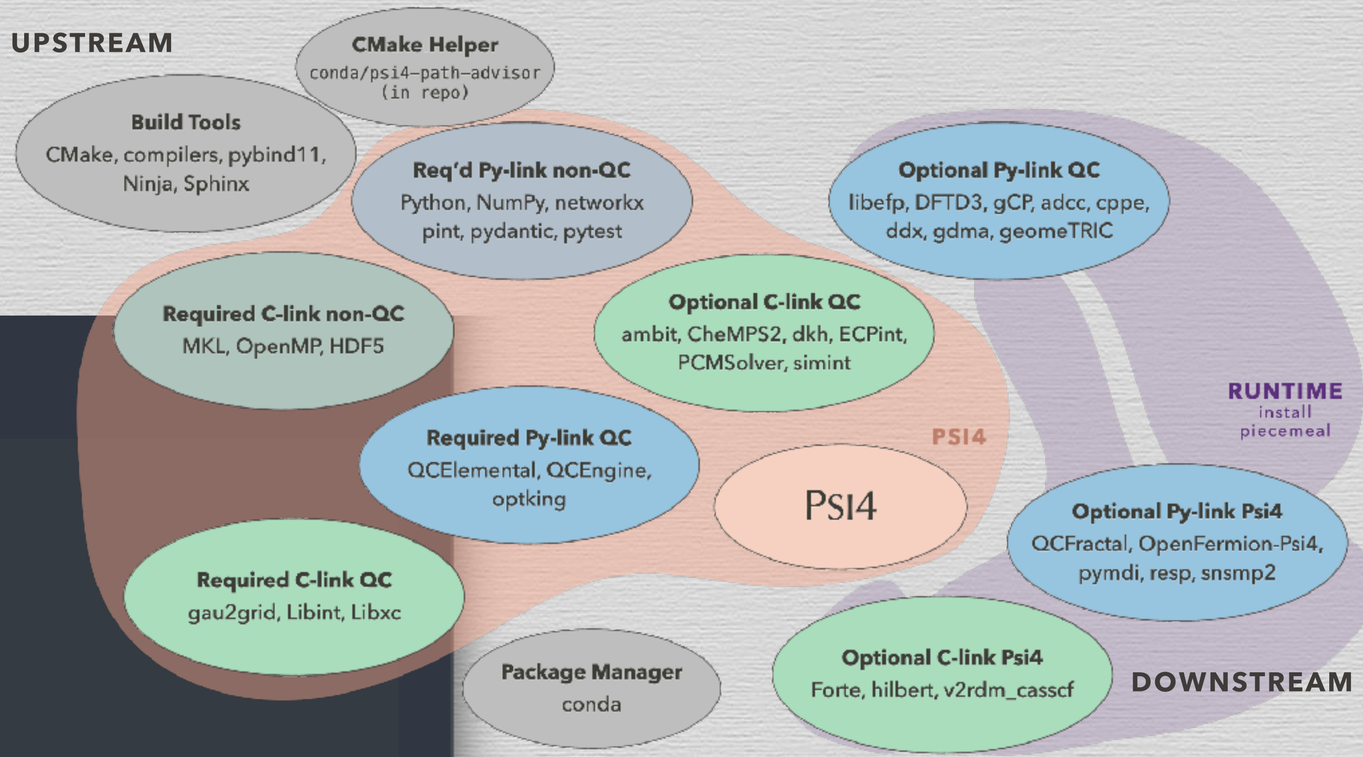
PY-WARY USERS:
CONDA INSTALLER

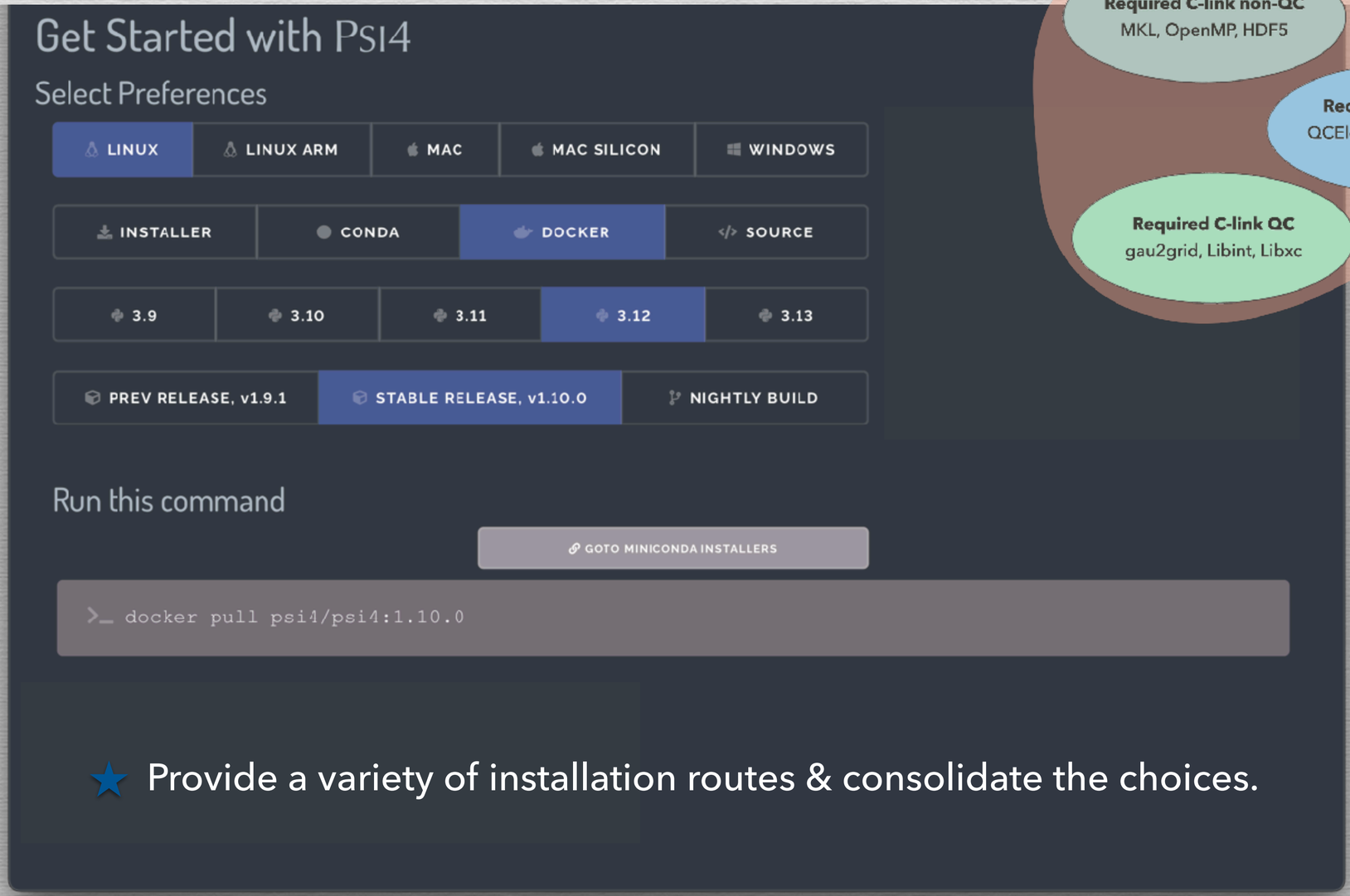
Psi4 · dependencies · add-ons

PSI4CONDA 633K downloads

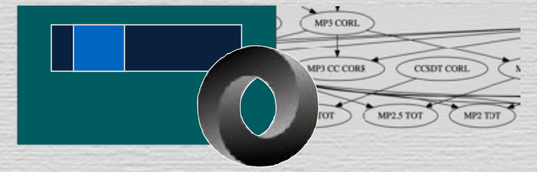
DISTRIBUTION

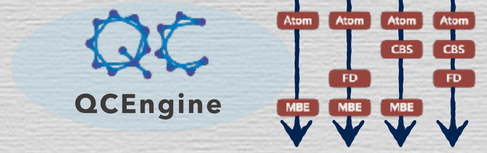
https://psicode.org/installs/latest





★ Provide a variety of installation routes & consolidate the choices.



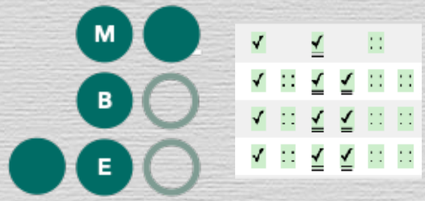


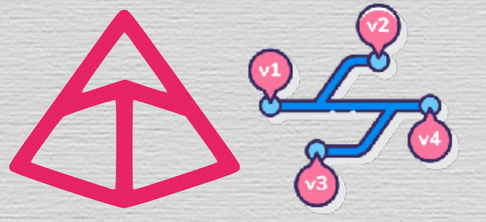
Psi4

OPEN-SOURCE QUANTUM CHEMISTRY

CODEDEPS/CI

GAU2GRID





GAU2GRID: COLLOCATION SPECIALIST

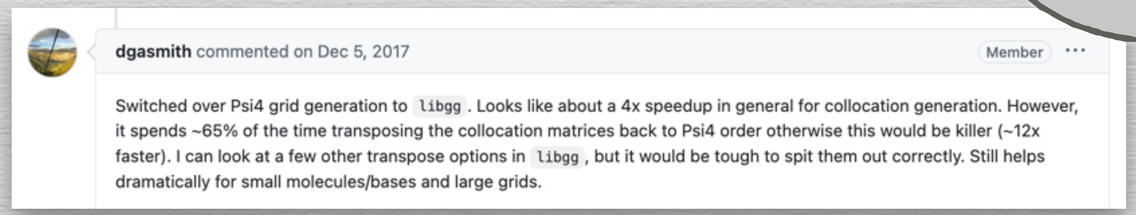
github.com/dgasmith/gau2grid (now psi4/gau2grid)

Identified **BOTTLENECK** in internal code

- **COLLOCATION** matrix btwn basis functions & DFT grid inefficient in Psi4 as simple non-vectorizable loops so extracted ops.

$$\phi_{mp} = Y_\ell^m(\mathbf{r}_p - \mathbf{r}_{\text{center}}) \sum_i c_i e^{-\alpha_i (\mathbf{r}_{\text{center}} - \mathbf{r}_p)^2}$$

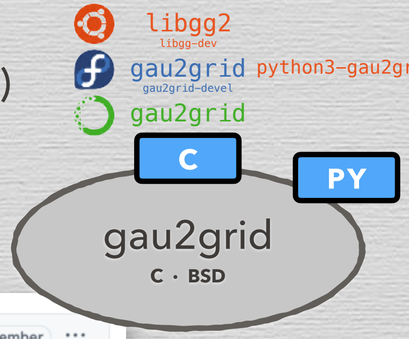
- Daniel Smith, Rob Parrish, Andy Simmonett for Psi4 **v1.2**
- Python generates a standalone **C LIBRARY** of unrolled loops. Algorithm adjusted for AM, deriv, hardware at gen-time
- **ARBITRARY** (specified) AM, up to 3rd derivatives
- C and Python (ref. impl.) **APIs**, no dependencies
- **RUNTIME** ordering switching w/o loss of efficiency (v2)
- Much easier to share **MAINTENANCE** on GitHub orgs
- **PAYOFF** enhanced efficiency for Psi4 DFT



Wrote new **C LIBRARY** in 2017 for Psi4

```c
#include "gau2grid/gau2grid.h"

#if psi4_SHGSHELL_ORDERING == LIBINT_SHGSHELL_ORDERING_STANDARD
const int order = (int)puream_ ? GG_SPHERICAL_CCA : GG_CARTESIAN_CCA;
#elif psi4_SHGSHELL_ORDERING == LIBINT_SHGSHELL_ORDERING_GAUSSIAN
const int order = (int)puream_ ? GG_SPHERICAL_GAUSSIAN : GG_CARTESIAN_CCA;
#endif

// Compute collocation
if (deriv_ == 0) {
    gg_collocation(L, npoints, xyz.data(), 1, nprim, norm, alpha,
                   center.data(), order, phi_start);
} else if (deriv_ == 1) {
    gg_collocation_deriv1(L, npoints, xyz.data(), 1, nprim, norm, alpha,
                          center.data(), order, phi_start,
                          phi_x_start, phi_y_start, phi_z_start);
}
```

Using Gau2grid in Psi4. Note the flexible ordering.



- extract code into a module to concentrate performance optimizations
- ship a Py reference implementation and/or a Py MWE to aid downstream adoption
- defer ordering configuration to runtime & build broadly for consumers

CENTRALIZING ECOSYSTEM PROJECTS

SSOT

- Packaging — providing artifacts compiled under fixed conditions — is easier than providing a flexible buildsys plus enough **GUIDANCE** for users to cope with an ecosystem build.
- Build guidance shouldn't be overly **PRESCRIPTIVE** and add another constraint to a developer's chosen workflow.
- After much floundering, I'm pleased with a single-source-of-truth file, `codedeps.yaml`, plus a mediating script `psi4-path-advisor`.
- **CODEDEPS** accumulates in YAML format all information on all the projects for building, augmenting, testing, documenting, or interfacing to Psi4. It runs to 1500 lines.

```
python -c "import yaml, pathlib; ydict = yaml.safe_load(pathlib.Path('../
codedeps.yaml').read_text());print([itm['project'] for itm in ydict['data']])"
#> ['c', 'cxx', 'fortran', 'python', 'cmake', 'generator', 'cfour', 'chemps2', 'dftd3',
'dkh', 'erd', 'gdma', 'lapack', 'libint', 'libefp', 'mrcc', 'openmp', 'pcmsolver',
'v2rdm_casscf', 'doxygen', 'python-graphviz', 'sphinx', 'sphinx-psi-theme', 'gcp',
'pybind11', 'pytest', 'nbsphinx', 'numpy', 'simint', 'ambit', 'gau2grid',
'memory_profiler', 'libxc', 'openfermionpsi4', 'qcelemental', 'adcc', 'brianqc',
'cppe', 'geometric', 'i-pi', 'mdi', 'mp2d', 'qcengine', 'dftd4', 'ecpint', 'qcfractal',
'scipy', 'ddx', 'optking', 'bse', 'einsums', 'integratorxx', 'gauxc', 'h5py']
```

CENTRALIZING ECOSYSTEM PROJECTS

SSOT

- Packaging — providing artifacts compiled under fixed conditions — is easier than providing a flexible buildsys plus enough **GUIDANCE** for users to cope with an ecosystem build.
- Build guidance shouldn't be overly **PRESCRIPTIVE** and add another constraint to a developer's chosen workflow.
- After much floundering, I'm pleased with a single-source-of-truth file, `codedeps.yaml`, plus a mediating script `psi4-path-advisor`.
- **CODEDEPS** accumulates in YAML format all information on all the projects for building, augmenting, testing, documenting, or interfacing to Psi4. It runs to 1500 lines.
- Two years ago it was experimental because I don't like another invented standard (does have a basic **SCHEMA**), but I'm increasingly pleased with the scheme.

IMPROVISED CODEDEPS SCHEMA

```
#  - project: !!str                 # Field required. Project <dummy> is replaced by the Psi4 internal label used for CMake (e.g., TODO FALSE ENABLE_<dummy>, external/upstream/<dummy>/CM
#    use: !!map                     # Section required. Important Psi4 dependency usage information.
#      added: !!str                   # Field required. Minor version at which <dummy> added to Psi4 ecosystem. v1.0 is earliest value.
#      deprecated: !!str              #                 Minor version at which <dummy> removed from Psi4 ecosystem. Further fields not present.
#      required: !!bool               # Field required. Whether <dummy> is a required Psi4 dependency.
#      buildable: !!bool              # Field required. If <dummy> is required or enabled, `true` if the Psi4 build system can supply it by building from source or `false` if it must be
#      buildtime: !!bool              # Field required. Whether <dummy> is compile-linkage (`true`) or runtume-linkage (`false`); some runtime are marked `true` if required to detect at
#      cms: !!bool                    # Field required. Whether <dummy> is a computational molecular sciences project vs. generic compsci.
#      test_required: !!bool          # Field required if used in Psi4 testing. Whether <dummy> is a required dependency for testing.
#      docs_required: !!bool          # Field required if used in building Psi4 documentation. Whether <dummy> is a required dependency for docs build.
#    repository: !!map              # Section required if use/buildable=true. Where to acquire source code to build.
#      host: !!str                    # Field required. (github | gitlab | url)
#      url: !!str                     # Field required if host=url. Full url to tarball for source.
#      account: !!str                 # Field required if host!=url. Personal or organization account name on host.
#      name: !!str or !!set           # Field required if host!=url. Project name on host/account. Use list for optional dependencies
#      commit: !!str                  # Field required if host!=url. The tag, branch, or hash of the repository at which to pull source. Usually latest compatible release, not earliest c
#    cmake: !!map                   # Section required if use/buildable=true. How/what to detect <dummy> with `find_package(<cmake/name> <cmake/constraint> COMPONENTS <cmake/components>)
#      name: !!str                    # Field required. CMake project name in `find_package` call. Often has capital letters.
#      constraint:                    # Field required. Version of earliest compatible release. May also include ATLEAST or EXACT. Remember CMake has only crude spec for version constrai
#      target: !!str                  # Field required. Target <cmake/name>::<cmake/target>> used by Psi4. Field may be `null` if no target, like for CMake-built python package.
#      components: !!set              # Field required. Required components for pre-built to be accepted. Field may be `null` if no components.
#    conda: !!map                   # Section required if use/required=true.
#      channel: !!str                 # Field required. Preferred channel on anaconda.org suitable for `-c`. Use the broadest if conda/name on multiple channels or subchannels.
#      name: !!str or !!map           # Field required. The conda package name. Use minimal required package (e.g., libxc-c (C intf) rather than libxc (C & Fortran intf)). Occasionally m
#      constraint: !!str or !!map     # Field required. Version or build number restrictions on package. Field should be `null` if most recent settings will do or needed to solve environ
#      aux_run_names: !!set           # Field optional. Use to list add'l packages needed to run dummy and Psi4 together that aren't deps of dummy (e.g., postgresql for qcfractal).
#      aux_build_names: !!set         # Field optional. Use to list add'l packages needed to build Psi4 against dummy that aren't deps of dummy (e.g., mkl-devel for mkl).
#      cmake: !!map                   # Field required. Use to specify CMake variables relevant to this package. Those starting with // aren't usually set. May involve substitutions.
#      skip_win: !!bool               # Field optional. Use to list packages not adapted for Windows. Perhaps should be in 'use' block?
```

CENTRALIZING ECOSYSTEM PROJECTS

SSOT

- Packaging — providing artifacts compiled under fixed conditions — is easier than providing a flexible buildsys plus enough **GUIDANCE** for users to cope with an ecosystem build.
- Build guidance shouldn't be overly **PRESCRIPTIVE** and add another constraint to a developer's chosen workflow.
- After much floundering, I'm pleased with a single-source-of-truth file, `codedeps.yaml`, plus a mediating script `psi4-path-advisor`.
- **CODEDEPS** accumulates in YAML format all information on all the projects for building, augmenting, testing, documenting, or interfacing to Psi4. It runs to 1500 lines.
- Two years ago it was experimental because I don't like another invented standard (does have a basic **SCHEMA**), but I'm increasingly pleased with the scheme.
- Main **GOALS**:
  - Users need to know if an addon is **INTERFACED** to Psi4
  - Packagers need to know what **DEPS** are new/retired/updated
  - Developers need to know how to **BUILD** with mixed binary and from-source dependencies
  - I need to not **SYNC** version/deps changes among multiple CMake, 2 Azure, 3 GHA, docs, and env spec files.
  - I need to know if the ecosystem that worked yesterday **BROKE** overnight.

CODEDEPS.YAML

TOOL

```yaml
- project: cxx
  use:
    added: "1.0"

    required: true
    required_note: "Must support c++20 standard."
    buildable: false
    buildtime: true

    cms: false
  repository: null




  cmake: null




  conda:
    channel: conda-forge
    name: cxx-compiler
    constraint: null
    aux_build_names:
      - //dpcpp_linux-64
    cmake:
      GNU_linux_64:
        CMAKE_CXX_COMPILER:
          "${CONDA_PREFIX}/bin/${HOST}-g++"
        //CMAKE_CXX_FLAGS:    null
      MSVC_win-64:
        CMAKE_CXX_COMPILER:   "clang-cl"
        //CMAKE_CXX_FLAGS:    null
      IntelLLVM_linux-64:
        CMAKE_CXX_COMPILER:
          "\"${CONDA_PREFIX}/bin/icpx;
           --target=${HOST};
           --gcc-toolchain=${CONDA_PREFIX};
           --sysroot=${CONDA_PREFIX}/${HOST}/sysroot\""
        //CMAKE_CXX_FLAGS:    null
```

OPT'L COMPILED

```yaml
- project: gdma
  use:
    added: "1.0"
    added_note: "Lib to pygdma at 1.9. Ext at 1.1."
    required: false
    required_note: "Allow Stone multipole analysis"
    buildable: true
    buildtime: false

    cms: true
  repository:
    host: github
    account: psi4
    name: gdma
    commit: v2.3.3
    commit_note: "GL upstream c2e0b5 + lib, I/O, Py"
  cmake:
    name: gdma
    constraint: 2.3.3
    components:
      - Python
    target: gdma::pygdma
    enable: ENABLE_gdma
  conda:
    channel: conda-forge
    name: pygdma
    constraint: null
    cmake:
      ENABLE_gdma: true
      gdma_DIR:
        unix: ${CONDA_PREFIX}/share/cmake/gdma
        win-64: ${CONDA_PREFIX}/Library/share/cmake/gdma
      //CMAKE_INSIST_FIND_PACKAGE_gdma: true
      //CMAKE_DISABLE_FIND_PACKAGE_gdma: true
    cmake_note: "Primarily OTF runtime detected.
           CMake enabling only relevant for CTest."
```

REQ'D PY

```yaml
- project: qcelemental
  use:
    added: 1.3

    required: true

    buildable: true
    buildtime: true
    buildtime_note: "BT for cmake checks; use is RT"
    cms: true
  repository:
    host: github
    account: MolSSI
    name: QCElemental
    commit: v0.29.0

  cmake:
    name: qcelemental
    constraint: "ATLEAST 0.28.0"


    target: null

  conda:
    channel: conda-forge
    name: qcelemental
    constraint: null
    aux_run_names:
      - msgpack-python
      - networkx
      - setuptools
      - pydantic>=1.10.17
    aux_run_names_note:
      "setuptools": "Needed for cmake detection"
      "pydantic>=1.10.17": "Uniform Pydantic v1 API"
    cmake:
      //CMAKE_INSIST_FIND_PACKAGE_qcelemental: true
      //CMAKE_DISABLE_FIND_PACKAGE_qcelemental: true
    cmake_note: "Primarily OTF runtime detected."
```

- **USE**: info like when added to Psi4, req'd/opt'l, and buildable, along with formal notes.

CODEDEPS.YAML

TOOL

```yaml
- project: cxx
  use:
    added: "1.0"

    required: true
    required_note: "Must support c++20 standard."
    buildable: false
    buildtime: true

    cms: false
  repository: null




  cmake: null




  conda:
    channel: conda-forge
    name: cxx-compiler
    constraint: null
    aux_build_names:
      - //dpcpp_linux-64
    cmake:
      GNU_linux_64:
        CMAKE_CXX_COMPILER:
          "${CONDA_PREFIX}/bin/${HOST}-g++"
        //CMAKE_CXX_FLAGS:    null
      MSVC_win-64:
        CMAKE_CXX_COMPILER:   "clang-cl"
        //CMAKE_CXX_FLAGS:    null
      IntelLLVM_linux-64:
        CMAKE_CXX_COMPILER:
          "\"${CONDA_PREFIX}/bin/icpx;
           --target=${HOST};
           --gcc-toolchain=${CONDA_PREFIX};
           --sysroot=${CONDA_PREFIX}/${HOST}/sysroot\""
        //CMAKE_CXX_FLAGS:    null
```

OPT'L COMPILED

```yaml
- project: gdma
  use:
    added: "1.0"
    added_note: "Lib to pygdma at 1.9. Ext at 1.1."
    required: false
    required_note: "Allow Stone multipole analysis"
    buildable: true
    buildtime: false

    cms: true
  repository:
    host: github
    account: psi4
    name: gdma
    commit: v2.3.3
    commit_note: "GL upstream c2e0b5 + lib, I/O, Py"
  cmake:
    name: gdma
    constraint: 2.3.3
    components:
      - Python
    target: gdma::pygdma
    enable: ENABLE_gdma
  conda:
    channel: conda-forge
    name: pygdma
    constraint: null
    cmake:
      ENABLE_gdma: true
      gdma_DIR:
        unix: ${CONDA_PREFIX}/share/cmake/gdma
        win-64: ${CONDA_PREFIX}/Library/share/cmake/gdma
      //CMAKE_INSIST_FIND_PACKAGE_gdma: true
      //CMAKE_DISABLE_FIND_PACKAGE_gdma: true
    cmake_note: "Primarily OTF runtime detected.
           CMake enabling only relevant for CTest."
```

REQ'D PY

```yaml
- project: qcelemental
  use:
    added: 1.3

    required: true

    buildable: true
    buildtime: true
    buildtime_note: "BT for cmake checks; use is RT"
    cms: true
  repository:
    host: github
    account: MolSSI
    name: QCElemental
    commit: v0.29.0

  cmake:
    name: qcelemental
    constraint: "ATLEAST 0.28.0"


    target: null

  conda:
    channel: conda-forge
    name: qcelemental
    constraint: null
    aux_run_names:
      - msgpack-python
      - networkx
      - setuptools
      - pydantic>=1.10.17
    aux_run_names_note:
      "setuptools": "Needed for cmake detection"
      "pydantic>=1.10.17": "Uniform Pydantic v1 API"
    cmake:
      //CMAKE_INSIST_FIND_PACKAGE_qcelemental: true
      //CMAKE_DISABLE_FIND_PACKAGE_qcelemental: true
    cmake_note: "Primarily OTF runtime detected."
```

- **USE**: info like when added to Psi4, req'd/opt'l, and buildable, along with formal notes.
- **REPOSITORY**: info to build from source: repository and find_package call.

CODEDEPS.YAML

TOOL

```yaml
- project: cxx
  use:
    added: "1.0"

    required: true
    required_note: "Must support c++20 standard."
    buildable: false
    buildtime: true

    cms: false
  repository: null




  cmake: null




  conda:
    channel: conda-forge
    name: cxx-compiler
    constraint: null
    aux_build_names:
      - //dpcpp_linux-64
    cmake:
      GNU_linux_64:
        CMAKE_CXX_COMPILER:
          "${CONDA_PREFIX}/bin/${HOST}-g++"
        //CMAKE_CXX_FLAGS:    null
      MSVC_win-64:
        CMAKE_CXX_COMPILER:   "clang-cl"
        //CMAKE_CXX_FLAGS:    null
      IntelLLVM_linux-64:
        CMAKE_CXX_COMPILER:
          "\"${CONDA_PREFIX}/bin/icpx;
           --target=${HOST};
           --gcc-toolchain=${CONDA_PREFIX};
           --sysroot=${CONDA_PREFIX}/${HOST}/sysroot\""
        //CMAKE_CXX_FLAGS:    null
```

OPT'L COMPILED

```yaml
- project: gdma
  use:
    added: "1.0"
    added_note: "Lib to pygdma at 1.9. Ext at 1.1."
    required: false
    required_note: "Allow Stone multipole analysis"
    buildable: true
    buildtime: false

    cms: true
  repository:
    host: github
    account: psi4
    name: gdma
    commit: v2.3.3
    commit_note: "GL upstream c2e0b5 + lib, I/O, Py"
  cmake:
    name: gdma
    constraint: 2.3.3
    components:
      - Python
    target: gdma::pygdma
    enable: ENABLE_gdma
  conda:
    channel: conda-forge
    name: pygdma
    constraint: null
    cmake:
      ENABLE_gdma: true
      gdma_DIR:
        unix: ${CONDA_PREFIX}/share/cmake/gdma
        win-64: ${CONDA_PREFIX}/Library/share/cmake/gdma
      //CMAKE_INSIST_FIND_PACKAGE_gdma: true
      //CMAKE_DISABLE_FIND_PACKAGE_gdma: true
    cmake_note: "Primarily OTF runtime detected.
           CMake enabling only relevant for CTest."
```

REQ'D PY

```yaml
- project: qcelemental
  use:
    added: 1.3

    required: true

    buildable: true
    buildtime: true
    buildtime_note: "BT for cmake checks; use is RT"
    cms: true
  repository:
    host: github
    account: MolSSI
    name: QCElemental
    commit: v0.29.0

  cmake:
    name: qcelemental
    constraint: "ATLEAST 0.28.0"


    target: null

  conda:
    channel: conda-forge
    name: qcelemental
    constraint: null
    aux_run_names:
      - msgpack-python
      - networkx
      - setuptools
      - pydantic>=1.10.17
    aux_run_names_note:
      "setuptools": "Needed for cmake detection"
      "pydantic>=1.10.17": "Uniform Pydantic v1 API"
    cmake:
      //CMAKE_INSIST_FIND_PACKAGE_qcelemental: true
      //CMAKE_DISABLE_FIND_PACKAGE_qcelemental: true
    cmake_note: "Primarily OTF runtime detected."
```

- **USE**: info like when added to Psi4, req'd/opt'l, and buildable, along with formal notes.
- **REPOSITORY**: info to build from source: repository and find_package call.
- **CONDA**: info to use pre-built, including any constraints and additional BT dependencies.

CODEDEPS → CONDA

GENERATED env_p4dev.yaml

```yaml
name: p4dev
channels:
  - conda-forge
  - nodefaults
dependencies:
    # build
  - c-compiler
  - cmake
  - cxx-compiler
  #- dpcpp_linux-64            # opt'l with cxx-compiler
  - ninja
    # non-qc buildtime required
  - blas-devel                # req'd with libblas
  - eigen                     # req'd with libint
  - libblas=*=*mkl
  - libboost-headers          # req'd with libint
  - llvm-openmp
  - numpy
  - pip                       # req'd with python; Tie pkg installer to this py if used.
  - pybind11
  - python
    # qc buildtime required
  - gau2grid
  #- psi4::libint              # opt'l with libint; Optionally for linux-64, an AM=7 build here.
  - libint
  - libxc-c
  - optking
  - qcelemental
  - qcengine
    # buildtime optional
  - dkh
  - einsums>=1.0.3
  - fortran-compiler
  - hdf5                      # req'd with einsums; Use 1.14.3 or 1.14.6 (so far) for HDF5_VERSION.
  - integratorxx
  - libecpint
  - pcmsolver
  - range-v3                  # req'd with einsums
  - zlib                      # req'd with einsums
    # runtime required
  - msgpack-python            # req'd with qcelemental
  - networkx                  # req'd with qcelemental
  - pydantic>=1.10.17         # req'd with qcelemental; Uniform Pydantic v1 API
  - scipy
  - setuptools                # req'd with qcelemental; Needed for cmake detection
    # runtime optional
  - adcc
  - basis_set_exchange
  - cppe
  - dftd3-python>=1.0
  - dftd4-python
  - gcp-correction
  - geometric
  - i-pi==3.1.0
  - postgresql                # req'd with qcfractal; Easy DB or supply your own.
  - pyddx
  - pygdma
  - pylibefp
  - pymdi
  - qcfractal>=0.60
  - toml                      # req'd with dftd4-python
    # test
  - pytest
  - pytest-xdist              # req'd with pytest; Parallel runner handy, !req'd.
```

psi4-path-advisor.py env

analyze codedeps and generate a Conda environment specification file w/ build reqs & optional full eco

```
conda/psi4-path-advisor.py env --help
#> usage: psi4-path-advisor env
#>      [-h] [-n NAME] [--python PYTHON]
#>      [--lapack {mkl,openblas,accelerate,blis,netlib}]
#>      [--disable {compilers,lapack,addons,test,docs}]
#>      [--platform {linux-64,osx-64,osx-arm64,win-64}]
#>      [--offline-conda]
```

REQ'D PY

```yaml
- project: qcelemental
  use:
    added: 1.3

    required: true

    buildable: true
    buildtime: true
    buildtime_note: "BT for cmake checks; use is RT"
    cms: true
  repository:
    host: github
    account: MolSSI
    name: QCElemental
    commit: v0.29.0

  cmake:
    name: qcelemental
    constraint: "ATLEAST 0.28.0"


    target: null

  conda:
    channel: conda-forge
    name: qcelemental
    constraint: null
    aux_run_names:
      - msgpack-python
      - networkx
      - setuptools
      - pydantic>=1.10.17
    aux_run_names_note:
      "setuptools": "Needed for cmake detection"
      "pydantic>=1.10.17": "Uniform Pydantic v1 API"
    cmake:
      //CMAKE_INSIST_FIND_PACKAGE_qcelemental: true
      //CMAKE_DISABLE_FIND_PACKAGE_qcelemental: true
  cmake_note: "Primarily OTF runtime detected."
```

CODEDEPS → CMAKE

TOOL

```
- project: cxx
  use:
    added: "1.0"

    required: true
    required_note: "Must support c++20 standard."
    buildable: false
    buildtime: true

    cms: false
  repository: null


  cmake:


  conda:
    channel: conda-forge
    name: cxx-compiler
    constraint: null
    aux_build_names:
      - //dpcpp_linux-64
    cmake:
      GNU_linux_64:
        CMAKE_CXX_COMPILER:
          "${CONDA_PREFIX}/bin/${HOST}-g++"
        //CMAKE_CXX_FLAGS:    null
      MSVC_win-64:
        CMAKE_CXX_COMPILER:   "clang-cl"
        //CMAKE_CXX_FLAGS:    null
      IntelLLVM_linux-64:
        CMAKE_CXX_COMPILER:
          "\"${CONDA_PREFIX}/bin/icpx;
           --target=${HOST};
           --gcc-toolchain=${CONDA_PREFIX};
           --sysroot=${CONDA_PREFIX}/${HOST}/sysroot\""
        //CMAKE_CXX_FLAGS:    null
```

psi4-path-advisor.py cache

analyze current conda env and generate likely & optional cache variables to seed CMake configuration

GENERATED
cache_p4dev.cmake

```
# <<<  cxx-compiler
set(CMAKE_CXX_COMPILER        /home/miniconda/envs/p4dev/bin/x86_64-conda-linux-gnu-g++ CACHE STRING "")
# set(CMAKE_CXX_FLAGS         <placeholder> CACHE STRING "")

# <<<  pygdma
# Primarily OTF runtime detected. With package present, CMake enabling only relevant for CTest.
set(ENABLE_gdma               ON CACHE BOOL "")
set(gdma_DIR                  /home/miniconda/envs/p4dev/share/cmake/gdma CACHE PATH "")
# set(CMAKE_INSIST_FIND_PACKAGE_gdma ON CACHE BOOL "")
# set(CMAKE_DISABLE_FIND_PACKAGE_gdma ON CACHE BOOL "")

# <<<  libblas                  blas-devel
set(LAPACK_LIBRARIES          /home/miniconda/envs/p4dev/lib/libmkl_rt.so CACHE STRING "")
set(LAPACK_INCLUDE_DIRS       /home/miniconda/envs/p4dev/include CACHE STRING "")

# <<<  libint                   libboost-headers, eigen
set(BOOST_ROOT                /home/miniconda/envs/p4dev CACHE STRING "")
set(Eigen3_ROOT               /home/miniconda/envs/p4dev CACHE STRING "")
set(Libint2_DIR               /home/miniconda/envs/p4dev/lib/cmake/libint2 CACHE PATH "")
# set(MAX_AM_ERI              5 CACHE STRING "")
# set(CMAKE_INSIST_FIND_PACKAGE_Libint2 ON CACHE BOOL "")
# set(CMAKE_DISABLE_FIND_PACKAGE_Libint2 ON CACHE BOOL "")

# <<<  qcelemental            msgpack-python, networkx, setuptools
# Primarily OTF runtime detected.
# set(CMAKE_INSIST_FIND_PACKAGE_qcelemental ON CACHE BOOL "")
# set(CMAKE_DISABLE_FIND_PACKAGE_qcelemental ON CACHE BOOL "")

...

# Sections skipped because packages detected on-the-fly, not through CMake
# <<<  pytest                   Primarily OTF runtime detected.
# <<<  geometric                Primarily OTF runtime detected.

# Sections skipped because packages absent from current environment
# <<<  ~pylibefp~
# <<<  ~integratorxx~
```

```
conda/psi4-path-advisor.py cache --help
#> usage: psi4-path-advisor cache
#>      [-h] [--objdir OBJDIR]
#>      [--compiler {byo,conda,GNU,~~IntelLLVM~~,~~Intel~~}]
#>      [--lapack {byo,conda,mkl}] [--insist]
```

CODEDEPS → REPO

psi4-path-advisor.py deploy

copy repository source info into many CMakeLists.txt. Also dry-run solve for all arch and write static conda env spec files.

external/gdma/CMakeLists.txt

```
if(${ENABLE_gdma})
    find_package(gdma 2.3.3 COMPONENTS Python
                 CONFIG QUIET)  # edit in codedeps

    if(${gdma_FOUND})
        get_property(_loc TARGET gdma::pygdma PROPERTY LOC
        message(STATUS "${Cyan}Found gdma${ColourReset}: $
        add_library(gdma_external INTERFACE)  # dummy

    else()
        if(${CMAKE_INSIST_FIND_PACKAGE_gdma})
            message(FATAL_ERROR "Suitable gdma could not b
        endif()
        message(STATUS "Suitable gdma could not be located

        ExternalProject_Add(gdma_external
            DEPENDS pybind11_external
            URL https://github.com/psi4/gdma/archive/v2.3.3.tar.gz  # Stone/upstream c2e0b5 + lib, I/O, Py  # edit in codedeps
            CMAKE_ARGS -DCMAKE_INSTALL_PREFIX=${STAGED_INSTALL_PREFIX}
                       -DCMAKE_CXX_COMPILER=${CMAKE_CXX_COMPILER}
                       -Dgdma_ENABLE_PYTHON=ON
                       -DPython_EXECUTABLE=${Python_EXECUTABLE}
                       -Dpybind11_DIR=${pybind11_DIR}
                       ...
```

OPT'L COMPILED

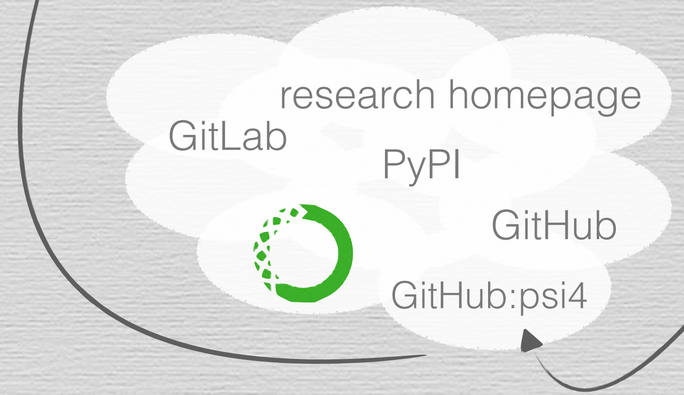
```
- project: gdma
  use:
    added: "1.0"
    added_note: "Lib to pygdma at 1.9. Ext at 1.1."
    required: false
    required_note: "Allow Stone multipole analysis"
    buildable: true
    buildtime: false

    cms: true
  repository:
    host: github
    account: psi4
    name: gdma
    commit: v2.3.3
    commit_note: "GL upstream c2e0b5 + lib, I/O, Py"
  cmake:
    name: gdma
    constraint: 2.3.3
    components:
      - Python
    target: gdma::pygdma
    enable: ENABLE_gdma
  conda:
    channel: conda-forge
    name: pygdma
    constraint: null
    cmake:
      ENABLE_gdma: true
      gdma_DIR:
        unix: ${CONDA_PREFIX}/share/cmake/gdma
        win-64: ${CONDA_PREFIX}/Library/share/cmake/gdma
      //CMAKE_INSIST_FIND_PACKAGE_gdma: true
      //CMAKE_DISABLE_FIND_PACKAGE_gdma: true
    cmake_note: "Primarily OTF runtime detected.
            CMake enabling only relevant for CTest."
```

psi4 / devtools / conda-envs /

- docs-cf.yaml
- linux-64-buildrun.yaml
- osx-64-buildrun.yaml
- osx-arm64-buildrun.yaml
- win-64-buildrun.yaml

PUTTING CODEDEPS TO USE FOR DEVS

conda-enabled superbuild




```
=========================================
  (B) flexible usage
=========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4

# (2.1) generate env spec file from codedeps.yaml.
conda/psi4-path-advisor.py env -n p4dev --python 3.12 --disable addons
#> conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev
# (2.2) edit env_p4dev.yaml to customize software packages.
# (2.3) issue suggested or customized command to create and activate conda env.
conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev

# (3.1) generate cmake cache file from conda env.
conda/psi4-path-advisor.py cmake
#> cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev && \
   cmake --build objdir_p4dev
# (3.2) edit cache_p4dev.cmake to customize build configuration.
# (3.3) issue suggested or customized command to configure and build with cmake.
cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev \
  -DCMAKE_INSTALL_PREFIX=/path/to/install-psi4 && cmake --build objdir_p4dev
```

- Sounds overly prescriptive, but the idea is to meet known tools at known configuration points.
- Above is the recc. build procedure for devs. Note steps (2.2) & (3.2) where you customize.
- Nowadays, use `fetch_content` instead of superbuild, as latter poor for non-top-level or shared deps.

PUTTING CODEDEPS TO USE FOR DEVS

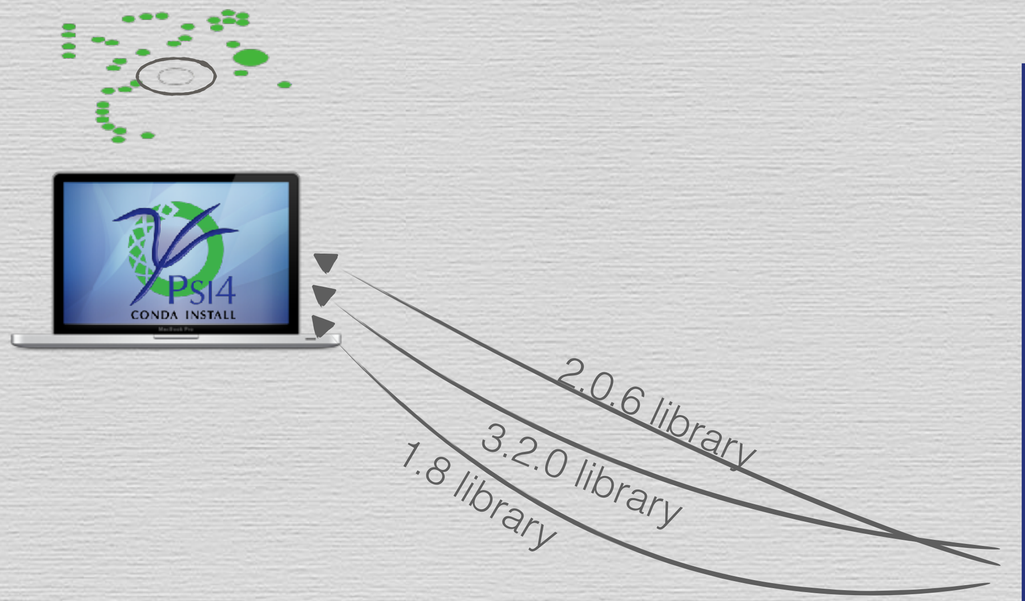
conda-enabled superbuild


```
=========================================
  (B) flexible usage
=========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4

# (2.1) generate env spec file from codedeps.yaml.
conda/psi4-path-advisor.py env -n p4dev --python 3.12 --disable addons
#> conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev
# (2.2) edit env_p4dev.yaml to customize software packages.
# (2.3) issue suggested or customized command to create and activate conda env.
conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev

# (3.1) generate cmake cache file from conda env.
conda/psi4-path-advisor.py cmake
#> cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev && \
   cmake --build objdir_p4dev
# (3.2) edit cache_p4dev.cmake to customize build configuration.
# (3.3) issue suggested or customized command to configure and build with cmake.
cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev \
  -DCMAKE_INSTALL_PREFIX=/path/to/install-psi4 && cmake --build objdir_p4dev
```

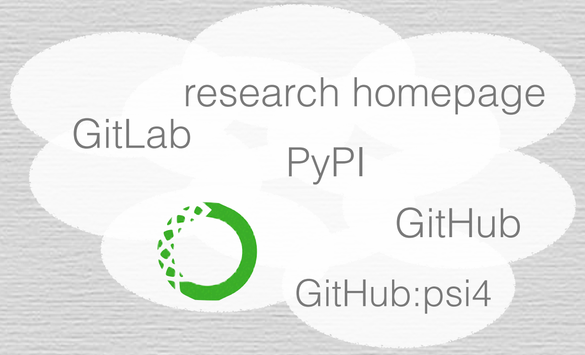
- Sounds overly prescriptive, but the idea is to meet known tools at known configuration points.
- Above is the recc. build procedure for devs. Note steps (2.2) & (3.2) where you customize.
- Nowadays, use `fetch_content` instead of superbuild, as latter poor for non-top-level or shared deps.

PUTTING CODEDEPS TO USE FOR DEVS

conda-enabled superbuild




```
=========================================
  (B) flexible usage
=========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4

# (2.1) generate env spec file from codedeps.yaml.
conda/psi4-path-advisor.py env -n p4dev --python 3.12 --disable addons
#> conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev
# (2.2) edit env_p4dev.yaml to customize software packages.
# (2.3) issue suggested or customized command to create and activate conda env.
conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev

# (3.1) generate cmake cache file from conda env.
conda/psi4-path-advisor.py cmake
#> cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev && \
   cmake --build objdir_p4dev
# (3.2) edit cache_p4dev.cmake to customize build configuration.
# (3.3) issue suggested or customized command to configure and build with cmake.
cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev \
  -DCMAKE_INSTALL_PREFIX=/path/to/install-psi4 && cmake --build objdir_p4dev
```



- Sounds overly prescriptive, but the idea is to meet known tools at known configuration points.
- Above is the recc. build procedure for devs. Note steps (2.2) & (3.2) where you customize.
- Nowadays, use `fetch_content` instead of superbuild, as latter poor for non-top-level or shared deps.

PUTTING CODEDEPS TO USE FOR DEVS

conda-enabled superbuild

compile

```
==========================================
  (B) flexible usage
==========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4

# (2.1) generate env spec file from codedeps.yaml.
conda/psi4-path-advisor.py env -n p4dev --python 3.12 --disable addons
#> conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev
# (2.2) edit env_p4dev.yaml to customize software packages.
# (2.3) issue suggested or customized command to create and activate conda env.
conda env create -n p4dev -f /home/psi4/env_p4dev.yaml && conda activate p4dev

# (3.1) generate cmake cache file from conda env.
conda/psi4-path-advisor.py cmake
#> cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev && \
   cmake --build objdir_p4dev
# (3.2) edit cache_p4dev.cmake to customize build configuration.
# (3.3) issue suggested or customized command to configure and build with cmake.
cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev \
  -DCMAKE_INSTALL_PREFIX=/path/to/install-psi4 && cmake --build objdir_p4dev
```

research homepage

GitLab

PyPI

GitHub

GitHub:psi4

- Sounds overly prescriptive, but th…on points.
- Above is the recc. build procedure…mize.
- Nowadays, use `fetch_content` inste…shared deps.

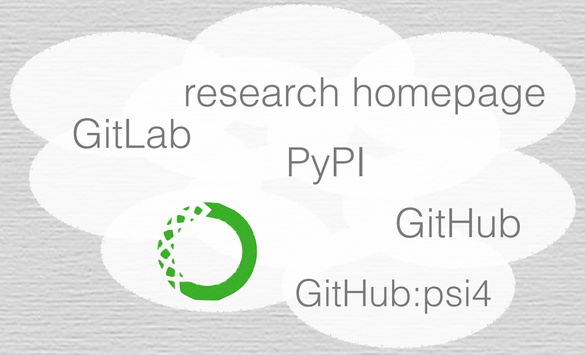
```
-- Found pybind11: /usr/share/miniconda/envs/p4build/include (found version 2.13.6)
-- Found ambit: /usr/share/miniconda/envs/p4build/lib/libambit.so (found version 0.6.0)
-- Found bse: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/basis_set_exchange (found version 0.11)
-- Found dkh: /usr/share/miniconda/envs/p4build/lib/libdkh.so (found version 1.2)
-- Found ecpint: /usr/share/miniconda/envs/p4build/lib/libecpint.so.1 (found version 1.0.7)
-- Found libefp: /usr/share/miniconda/envs/p4build/lib/libefp.so.1.5 (found version 1.5.0)
-- Found Einsums: /usr/share/miniconda/envs/p4build/lib/libEinsums.so.1.0.4 (found version 1.0.4)
-- Found gau2grid: /usr/share/miniconda/envs/p4build/lib/libg2g.so.2 (found version 2.0.7)
-- Found gdma: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/gdma.cpython-313-x86_64-linux-gnu.so (found version 2.3.3)
-- Found Boost: /usr/share/miniconda/envs/p4build/include (found suitable version "1.88.0", minimum required is "1.29")
-- Found Libint2 6: /usr/share/miniconda/envs/p4build/lib/libint2.so.2.9.0 (found version 2.9.0)
-- Found PCMSolver: /usr/share/miniconda/envs/p4build/lib/libpcm.so.1 (found version 1.2.3.999)
-- Found pylibefp: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/pylibefp (found version undefined)
-- Found qcelemental: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/qcelemental (found version 0.29.0)
-- Found qcengine: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/qcengine (found version v0.32.0)
-- Found optking: /usr/share/miniconda/envs/p4build/lib/python3.13/site-packages/optking (found version 0.3.0)
-- Found Libxc: /usr/share/miniconda/envs/p4build/lib/libxc.so.15 (found version 7.0.0)
-- Suitable BrianQC could not be located, please contact info@brianqc.com to obtain BrianQC
-- Found IntegratorXX: /usr/share/miniconda/envs/p4build/lib/libintegratorxx.so (found version 1.2.0)
-- Suitable v2rdm_casscf could not be located, Building v2rdm_casscf instead.
-- No Doxygen, no docs.
-- No Sphinx, no docs. Pre-built documentation at http://psicode.org/psi4manual/master/index.html
```

PUTTING CODEDEPS TO USE FOR DEVS

conda-enabled superbuild


```
==========================================
  (B) flexible usage
==========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4
# (2) generate env spec file from codedeps.yaml. "eval $(...)" creates and activates conda env.
eval $(conda/psi4-path-advisor.py env)
# (3) generate cmake cache file from conda env. "eval $(...)" configures and builds with cmake.
eval $(conda/psi4-path-advisor.py cmake)

shows up in p4dev
# (3.2) edit cache_p4dev.cmake to customize build configuration.
# (3.3) issue suggested or customized command to configure and build with cmake.
cmake -S. -C/home/psi4/cache_p4dev.cmake -Bobjdir_p4dev \
  -DCMAKE_INSTALL_PREFIX=/path/to/install-psi4 && cmake --build objdir_p4dev
```

```
==========================================
  (A) black-box usage (copy/paste-able)
==========================================
# (1) get code from GitHub
git clone https://github.com/psi4/psi4.git && cd psi4
# (2) generate env spec file from codedeps.yaml. "eval $(...)" creates and activates conda env.
eval $(conda/psi4-path-advisor.py env)
# (3) generate cmake cache file from conda env. "eval $(...)" configures and builds with cmake.
eval $(conda/psi4-path-advisor.py cmake)

shows up in p4dev
```



- Sounds overly prescriptive, but the idea is to meet known tools at known configuration points.
- Above is the recc. build procedure for devs. Note steps (2.2) & (3.2) where you customize.
- Nowadays, use `fetch_content` instead of superbuild, as latter poor for non-top-level or shared deps.
- In practice, though, the procedure is so steady, one can do the short route:

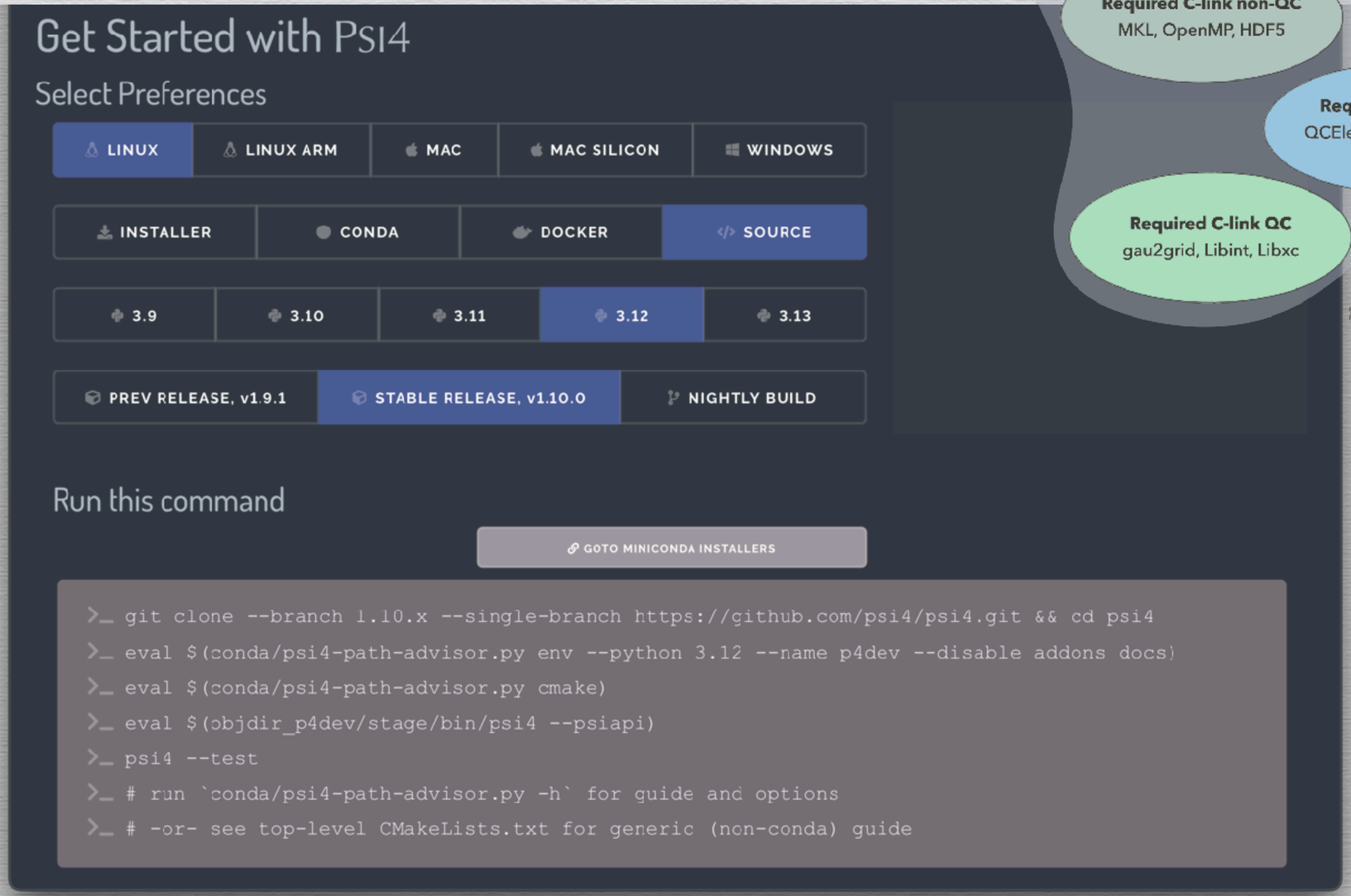
★At the ecosys level, even simple dependencies logic (pytest is for testing) can get overwhelming, so consolidate & encode.

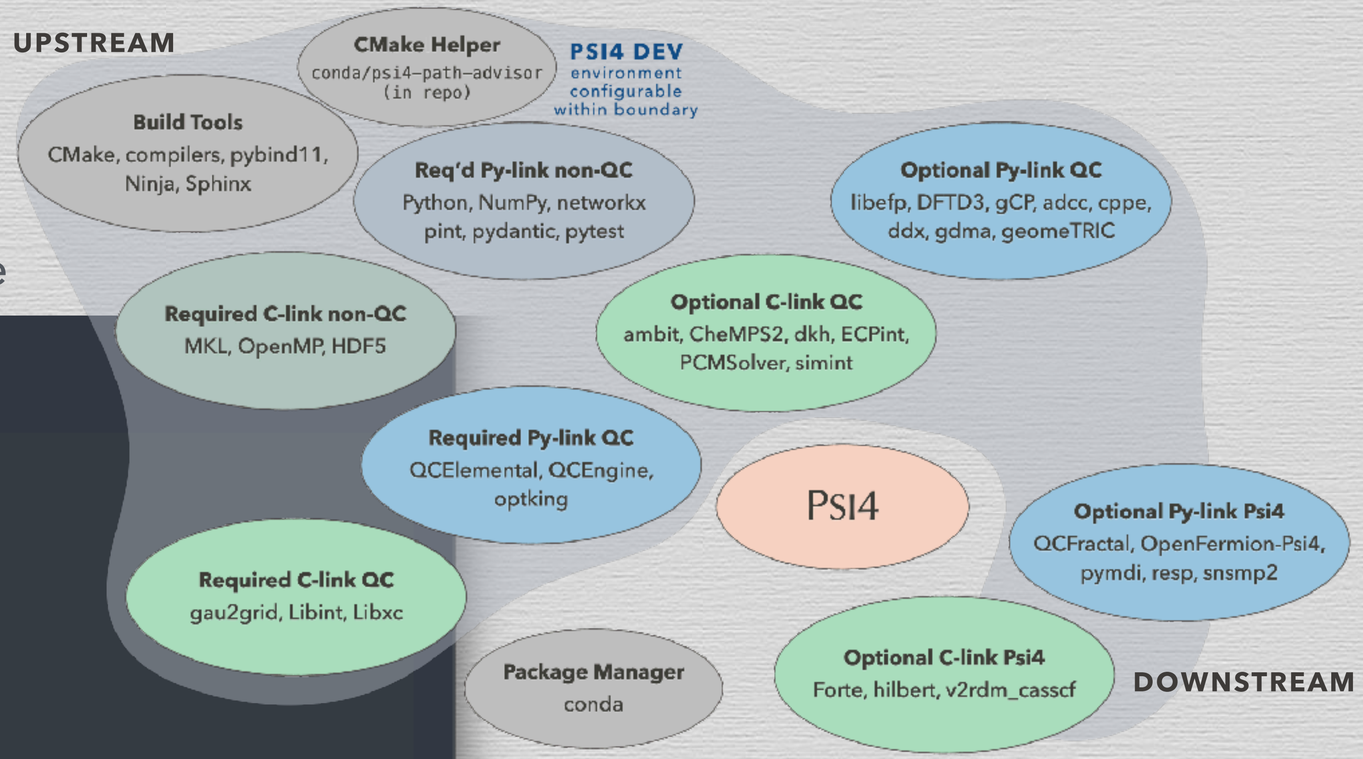
★A SSOT deps file can be a path to recovery when the build breaks by broadcast fixes

DISTRIBUTION

https://psicode.org/installs/latest

★Aim for copy/paste install even for compiled software





CORE DEVELOPERS: CONDA-ENABLED SUPERBUILD

deps · dev tools · add-ons · cmake cmd

PY-FRIENDLY USERS: CONDA PACKAGE

628K

Psi4 · dependencies · add-ons

PY-WARY USERS: CONDA INSTALLER

PSI4CONDA 633K

Psi4 · dependencies · add-ons

CONTINUOUS INTEGRATION TESTING

build passing

a repository's lifeline and anchor

Strongly recommended characteristics of a **TEST RUNNER**

- **RUN-TIME DETECTION**: optional dependencies bringing new capabilities may be runtime-loaded, so a runner should detect current environment dependencies without reconfiguration (like CTest).
  - QCElemental & QCEngine provide detection utilities
  - Pytest marks can run, skip, & select: **`pytest -m dftd3`**

MARK TESTS TO RUN OR SKIP

```
@pytest.mark.scf
@pytest.mark.dft
@uusing("libxc")
@pytest.mark.parametrize("func,expected", [
  pytest.param('wM06-D3', 0.4563, marks=using("dftd3")), # Libxc 180
  pytest.param(  'X3LYP', 0.4549),  # Q-Chem
  pytest.param(  'KMLYP', 0.4665),  # ERKALE
], ids=name_dft_test)
def test_dft_bench_ionization(func, expected, dft_bench_systems):
    """functionals ionization energies vs. Q-Chem"""
    ...
```

DETECT ADDONS

```
{
    "mrcc": which("dmrcc", return_bool=True),
    "qcfractal": is_qcfractal_new_enough("0.60"),
    "bse": which_import("basis_set_exchange", return_bool=True),
    "gauxc": psi4.addons("gauxc"),
}
```

CONTINUOUS INTEGRATION TESTING

build passing

a repository's lifeline and anchor

Strongly recommended characteristics of a **TEST RUNNER**

- **RUN-TIME DETECTION**: optional dependencies bringing new capabilities may be runtime-loaded, so a runner should detect current environment dependencies without reconfiguration (like CTest).
  - QCElemental & QCEngine provide detection utilities
  - Pytest marks can run, skip, & select: **`pytest -m dftd3`**
- **INSTALL-TIME TESTABLE**: tests are most often run during development and packaging, but in an ecosystem env., it's useful to test component installed packages.
  - even if you're happy with an in-repo-only runner like CTest, pytest can often run those, too. PSI4 has a CTest suite & a Pytest suite, but the latter can run the former.
  - easy testing is another use for Py intf to a compiled lib

INSTALLED TESTING

```
>>> psi4 --test
test_ddx.py::test_ddx_fock_build[h2lpb] SKIPPED (ddx !found)
test_mdi.py::test_mdi_water SKIPPED (mdi !found)
test_option.py::test_spacious_option PASSED
test_psi4.py::test_psi4_basic PASSED
test_psi4.py::test_psi4_cc PASSED
test_psi4.py::test_psi4_cas PASSED
test_psi4.py::test_psi4_dfmp2 PASSED
test_psi4.py::test_psi4_sapt PASSED
test_psi4.py::test_psi4_scfproperty PASSED
test_psi4_qcschema.py::test_psi4_basic PASSED
test_psi4_qcschema.py::test_psi4_cc XFAIL (no AtomicInput opt)
test_psi4_qcschema.py::test_psi4_cas PASSED
test_psi4_qcschema.py::test_psi4_dfmp2 PASSED
test_psi4_qcschema.py::test_psi4_sapt PASSED
test_qcfractal.py::test_qcf_cbs_mbe[internal] PASSED
test_qcfractal.py::test_qcf_cbs_mbe[snowflake] PASSED
test_thc_eri.py::test_ls_thc_exact PASSED
tu1-h2o-energy/test_input.py::test_tu1_h2o_energy PASSED
===== 49 passed, 60 skipped, 7934 deselected, 1 xfailed in 4m
====
```

CONTINUOUS INTEGRATION TESTING

build passing

a repository's lifeline and anchor

Strongly recommended characteristics of a **TEST RUNNER**

- **RUN-TIME DETECTION**: optional dependencies bringing new capabilities may be runtime-loaded, so a runner should detect current environment dependencies without reconfiguration (like CTest).
  - QCElemental & QCEngine provide detection utilities
  - Pytest marks can run, skip, & select: `pytest -m dftd3`
- **INSTALL-TIME TESTABLE**: tests are most often run during development and packaging, but in an ecosystem env., it's useful to test component installed packages.
  - even if you're happy with an in-repo-only runner like CTest, pytest can often run those, too. Psi4 has a CTest suite & a Pytest suite, but the latter can run the former.
  - easy testing is another use for Py intf to a compiled lib
- **ERROR TESTING**: error & warning pathways & deprecated features should be as easy to test as numerical results. QC isn't run only by specialists anymore.
  - fend off nonsense jobs before resources run out
  - nearly as easy to enforce advice in code as via docs
  - check code paths that helpful error is being raised

CHECK ERROR PATHWAYS FUNCTIONING

```
def test_jumbledzmat_error():
    subject = """He
                 He 1 2. 2 100. 3 35.
                 He 1 2.
              """

    with pytest.raises(qcelemental.ValidationError) as e:
        qcelemental.molparse.from_string(subject)

    assert 'aim for lower triangular' in str(e)
```

CONTINUOUS INTEGRATION TESTING

build passing

a repository's lifeline and anchor

Strongly recommended characteristics of a **TEST RUNNER**

- **RUN-TIME DETECTION**: optional dependencies bringing new capabilities may be runtime-loaded, so a runner should detect current environment dependencies without reconfiguration (like CTest).
  - QCElemental & QCEngine provide detection utilities
  - Pytest marks can run, skip, & select: **`pytest -m dftd3`**
- **INSTALL-TIME TESTABLE**: tests are most often run during development and packaging, but in an ecosystem env., it's useful to test component installed packages.
  - even if you're happy with an in-repo-only runner like CTest, pytest can often run those, too. Psi4 has a CTest suite & a Pytest suite, but the latter can run the former.
  - easy testing is another use for Py intf to a compiled lib
- **ERROR TESTING**: error & warning pathways & deprecated features should be as easy to test as numerical results. QC isn't run only by specialists anymore.
  - fend off nonsense jobs before resources run out
  - nearly as easy to enforce advice in code as via docs
  - check code paths that helpful error is being raised

GOBBLEDYGOOK OF EXPECTED ERRORS

```
_p42 = (psi4.MissingMethodError, r"Method=(mp4\(sdq\)|mp4|qcisd|
qcisd\(t\)) is not available for any derivative level under
conditions (CI|MP)_TYPE=CONV, REFERENCE=(U|RO)HF, FREEZE_CORE=(TRUE|
FALSE), QC_MODULE=\(auto\)", "no open-shell mp4(sdq)/mp4/qcisd/
qcisd(t) by fnocc")
_p43 = (psi4.ValidationError, r"(CD|DF) is not a valid choice", "no
df/cd ci by psi4")
_p44 = (psi4.ValidationError, r"Invalid type (DF|CD) for FNOCC energy
through `run_(fnocc|cepa)`.", "no df/cd except ccsd/ccsd(t) by
fnocc")
_p45 = (psi4.ValidationError, r"Reference UHF for DETCI is not
available", "no uhf by detci")
_p46 = (psi4.ValidationError, r"Invalid type (DF|CD) for DETCI energy
through `run_detci`", "no df/cd by detci")
_p47 = (psi4.UpgradeHelper, r"Replace method ZAPT with method MP for
RHF reference", "retire rhf zapt by detci")
_p48 = (psi4.MissingMethodError, r"Method=cisd is not available for
any derivative level under conditions CI_TYPE=CONV, REFERENCE=UHF,
FREEZE_CORE=(TRUE|FALSE), QC_MODULE=\(auto\)", "no uhf cisd by psi4")
_p98 = (psi4.MissingMethodError, r"Method=ccsd\(t\) is not available
for requested derivative level \(reqd=1 > avail=0\) under conditions
CC_TYPE=CONV, REFERENCE=(R|U)HF, FREEZE_CORE=(TRUE|FALSE),
QC_MODULE=\(auto\)", "temporary: cc(t) disabled w/o
qc_module=ccenergy for conv rhf/uhf gradients for ccsd(t) by ccenergy
until scaling reworked")
_p99 = (psi4.MissingMethodError, r"Method=(ccsd\(t\)|a-ccsd\(t\)) is
not available for any derivative level under conditions CC_TYPE=(DF|
CD), REFERENCE=UHF, FREEZE_CORE=(TRUE|FALSE), QC_MODULE=\(auto\)",
"temporary: cc(t) disabled w/o qc_module=occ in dfocc until further
optimization ")
_w1 = ("CCSD TOTAL GRADIENT",  "wrong semicanonical ae rohf ccsd
gradients by ccenergy")
```

7713 total tests in suite

build passing

CONTINUOUS INTEGRATION TESTING

a repository's lifeline and anchor

INTERNAL

```
conda/psi4-path-advisor.py \
  env \
  --name p4env \
  --python $PYTHON_VER \
  --disable compilers addons docs

# create conda env from yaml spec

conda/psi4-path-advisor.py \
  cmake \
  --objdir build \
  --insist

# configure cmake from cache
```

DOCS

```
code/conda/psi4-path-advisor.py \
  env \
  --name p4docs \
  --python ${{ matrix.python-ver }} \
  --disable addons \
  --offline-conda

# env, cache, configure as above
```

ECO

```
conda/psi4-path-advisor.py \
  env \
  --name p4build \
  --python ${{ matrix.python-ver }} \
  --disable docs \
  --offline-conda

# env, cache, configure as above
```

PSI4 INTERNAL CI, AZURE

psi4 / psi4 / Pipelines / psi4.psi4 / 20250618.3

Name	Status	Duration
Windows VS2022	Success	1h 43m 18s
Linux Builds gcc_11	Success	1h 5m 50s
Linux Builds gcc_latest	Success	1h 57m 12s
Linux Builds clang_latest	Success	1h 19m 26s

2296 tests of internal Psi4 capabilities, Azure is cmdline-like for users, non-conda compilers

- quick
- quick + 1/3 full, earliest GCC, Py
- quick + 1/3 full, latest GCC, Py
- quick + 1/3 full

PSI4 ECOSYSTEM CI, GHA

- ✓ Docs / Propose Docs (ubuntu-latest, 3.10) (pull_request) Successful in 28m
- ✓ Eco / Eco • 3.10 • Conda Clang (M-Silicon) (pull_request) Successful in 37m
- ✓ Eco / Eco • 3.11 • Conda Clang (M-Intel) (pull_request) Successful in 103m
- ✓ Eco / Eco • 3.12 • LLVM clang-cl (W) (pull_request) Successful in 60m
- ✓ Eco / Eco • 3.13 • Conda GNU (L) (pull_request) Successful in 45m
- ✓ psi4.psi4 — #20250618.3 succeeded Required

<386 tests integrating with external software, architecture variety, all-conda

- addon + quick
- addon + quick
- addon
- addon

★ Bring the changing env of diff arch, compilers, dep ver into freq. contact w/code.

★ Use CI to assure developers always getting a full & proved working environment.

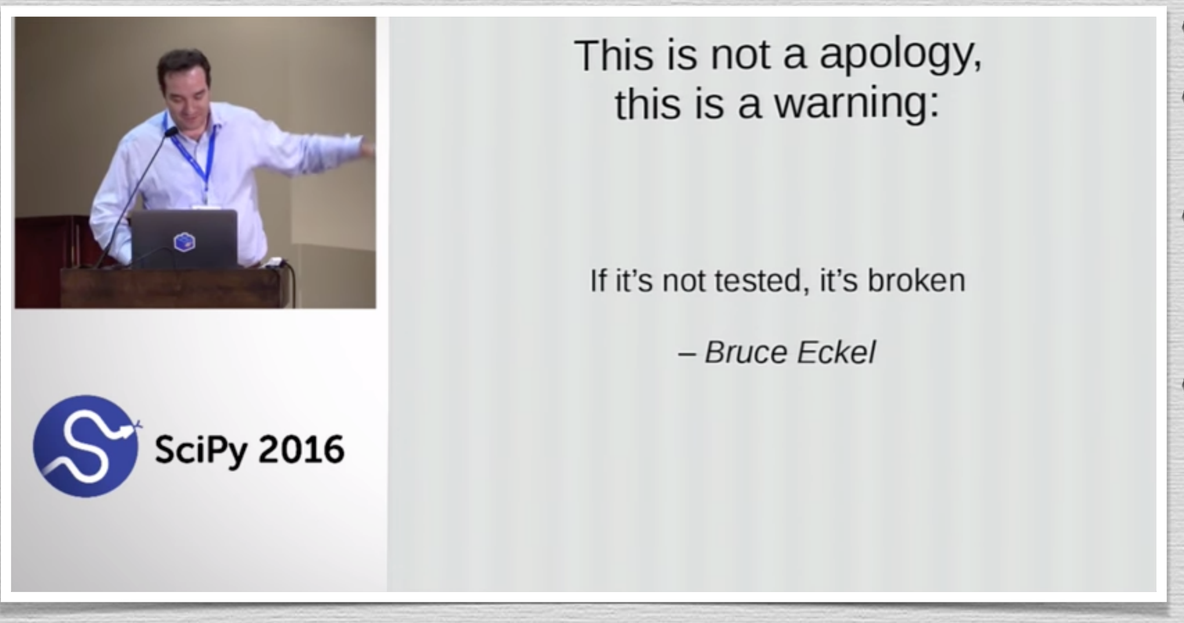
CONTINUOUS INTEGRATION TESTING

build passing

★ a repository's lifeline and anchor

LIFELINE

- **CONSOLIDATE** and define gains
- **OUTSOURCE** fixing problems or discussing changes to PRs that introduce conflicts
- **FACILITATE** refactoring and improving codebase
- **ATTUNED** to version & packaging changes in your own deps

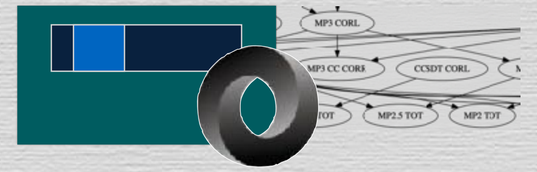


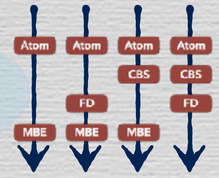
ANCHOR

- **CODE** doesn't rot but envs sure do
- **ECO BUFFETTED** by every problem of upstream, most of them not QC
- As other packages are better maintained, their SE virtues (upgrades & deprecations) hit you.
- **FEATURE** Using CI, you're arranging trouble to hit the manager, not users & arranging for tweaks to reach users (pathadvisor). Try to consider it a feature, not a bug.
- **EXAMPLES** from the past year
  - stray pkg_resources usage in an addon triggered a printed setuptools warning when importing Psi4, which printing interfered with qcng QC detection, breaking 1/3 of Psi4
  - syntax warning in Clang v18 collided with MSVC header Libint includes, repositioned in unreleased v19, so fallback to v17
  - GHA using a Windows image with different compiler & STL version; add define
  - NumPy v2 needs to be used with latest Pybind11; seen as "atoms too close" error
  - new compilers insert a breakpoint for oob memory access: sudden "Illegal Instruction"

WHAT'S MISSING?

- testing against **DEV** (not release) versions of eco projects
- **PERFORMANCE** regression testing
- new arch or **HARDWARE** test suite

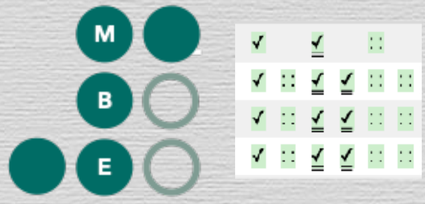




Psi4

OPEN-SOURCE QUANTUM CHEMISTRY

STDSUITE



QCMANYBODY

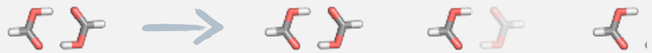
EXTRACTING THE QCMANYBODY PROCEDURE

multiple entry points for flexibility

Psi4

```
class ManyBodyComputer ():
```

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

QCManyBody

- noCP, SSFC, VMFC
- energy, gradient, Hess.
- IE or MBE
- indep. or schema intf.
- intermolecular only
- multiple model chemistry MBE

Ben Pritchard
MolSSI

Asem Alenaizan
GaTech → KFUPM

Daniel Smith
MolSSI → Abiologics

QCManyBody direct

QCFractal service

MBE backend

QCEngine procedure

Psi4 distributed driver

Optimizer backend

TWO INTERFACES FOR QCMANYBODY

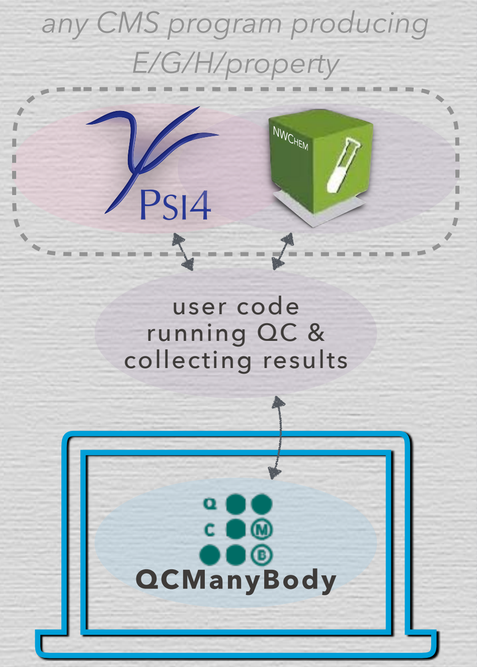
independent or integrated with QCArchive

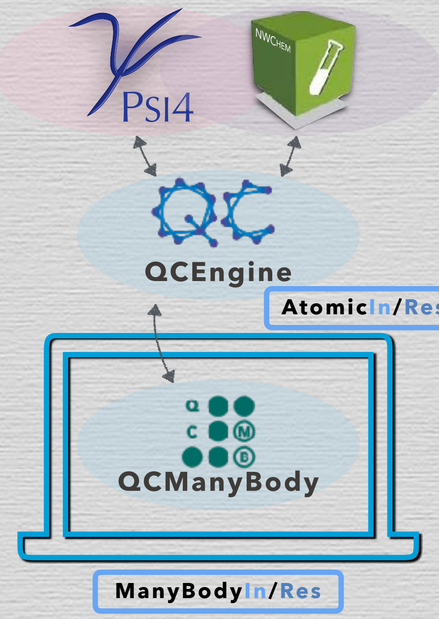
CORE INTERFACE

```python
from qcmanybody import ManyBodyCore

mbc = ManyBodyCore(
        molecule= Ne – Ne – Ne,
        bsse_type=["cp"],
        levels={1: "ccsd/cc-pvtz",
                2: "mp2/cc-pvdz",
                3: "mp2/cc-pvdz"},
        return_total_data=True,
        supersystem_ie_only=False,
        embedding_charges=None)

calculation_results = {}
for chem, lbl, imol in mbc.iterate_molecules():
    # use chem, lbl, imol to construct inputs

    # run them & fill in results
    calculation_results[lbl] = {"energy": None}
    #> calculation_results = {
    #>   '["mp2/cc-pvdz", [1], [1, 2]]': {'energy': -128.6814},}

results = mbc.analyze(calculation_results)

print(results["ret_energy"])
#> -384.1286
```

HIGH-LEVEL INTERFACE

```python
from qcmanybody import ManyBodyComputer
from qcmanybody.models import ManyBodyInput

mbin = ManyBodyInput(**{
  "molecule": Ne – Ne – Ne,
  "specification": {
    "driver": "energy",
    "keywords": {
      "bsse_type": ["cp"],
      "levels": {
        1: "nwc-ccsd/tz",
        3: "p4-mp2/dz"}},
    "specification": {
      "nwc-ccsd/tz": {
        "program": "nwchem",
        "model": {
          "method": "ccsd",
          "basis": "cc-pvtz"}},
      "p4-mp2/dz": {
        "program": "psi4",
        "model": {
          "method": "mp2",
          "basis": "cc-pvdz"}},
}})

# runs with QCEngine
ret = ManyBodyComputer.from_manybodyinput(mbin)

print(ret.return_result)
#> -384.1286
```





QCManyBody core interface

- user-prov. serial/parallel
- user runs CMS backends

QCManyBody high-level interface

- serial
- multi CMS backends

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyKeywords

bsse_type: any combination of
- nocp/mbe
- cp/ssfc
- vmfc

embedding_charges
return_total_data: IE or total
levels:
- {2: hi, 4: lo}
- {1: hi, 2: md, supersystem: lo}

max_nbody: truncate MBE early
supersystem_ie_only: no MBE analysis

ManyBodySpecification

schema_version: 1
program: any MBE speaking QCSchema
protocols: customize MBE return layout

ManyBodyProtocols
component_results: save cluster data

extras: free-form storage
driver: MBE target derivative
- energy
- gradient
- Hessian

E *G* *H*

keywords: knobs in the MBE program

ManyBodyKeywords

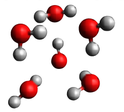
specification: how to run MBE *subsystems*
- single

AtomicSpecification

- mapping

{hi: **AtomicSpecification**,
lo: **AtomicSpecification** }

ManyBodyInput

schema_version: 1
provenance: creator name and version info
`QCManyBody, 0.3.0`
id: tracker for databases
extras: free-form storage
molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

AtomicSpecification

schema_version: 1

program: any SP in QCEngine

protocols: customize return layout

AtomicProtocols

stdout: save text output

wavefunction: save orbital info

native_files: save raw prog. files

error_correction: auto-edit&rerun

extras: free-form storage

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

ManyBodySpecification

schema_version: 1

program: any MBE speaking QCSchema

protocols: customize MBE return layout

ManyBodyProtocols

component_results: save cluster data

extras: free-form storage

driver: MBE target derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*

keywords: knobs in the MBE program

ManyBodyKeywords

specification: how to run MBE *subsystems*

- single

AtomicSpecification

- mapping

{hi: **AtomicSpecification**,

lo: **AtomicSpecification** }

ManyBodyInput

schema_version: 1

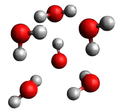
provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:



Molecule

specification: *how* to run MBE single-point

ManyBodySpecification

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyProperties (abr.)

calcinfo_nfr: M

return_energy: E

return_gradient: G

***bsse*_corrected_interaction_*egh*_through_*m*_body:**
- defined as E^m_{IE}
- defined for all *bsse* as **bsse_type** allows
- defined for all *egh* as **driver** allows
- def. for all *m* as **max_nbody/levels** allows

***bsse*_corrected_total_*egh*_through_*m*_body:**
- defined as E^m

***bsse*_corrected_*m*_body_contribution_to_*egh*:**
- defined as $E^{(m)}$

ManyBodyResult

schema_version: 1

input_data: record of starting conditions

ManyBodyInput

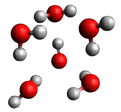
schema_version: 1

provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

success: True

provenance: creator, version, & RT info

`QCManyBody, 0.4.0, psinet, CPU E5-2630`

id: tracker for databases

stdout: program's text output for humans

stderr: anything in error channel

native_files: requested raw program files

extras: free-form storage

molecule: orientation & frame for results

return_result: MBE derivative

E G H

properties: key results for MBE

ManyBodyProperties

cluster_properties: key results for subsystems

{cluster id: **AtomicProperties** }

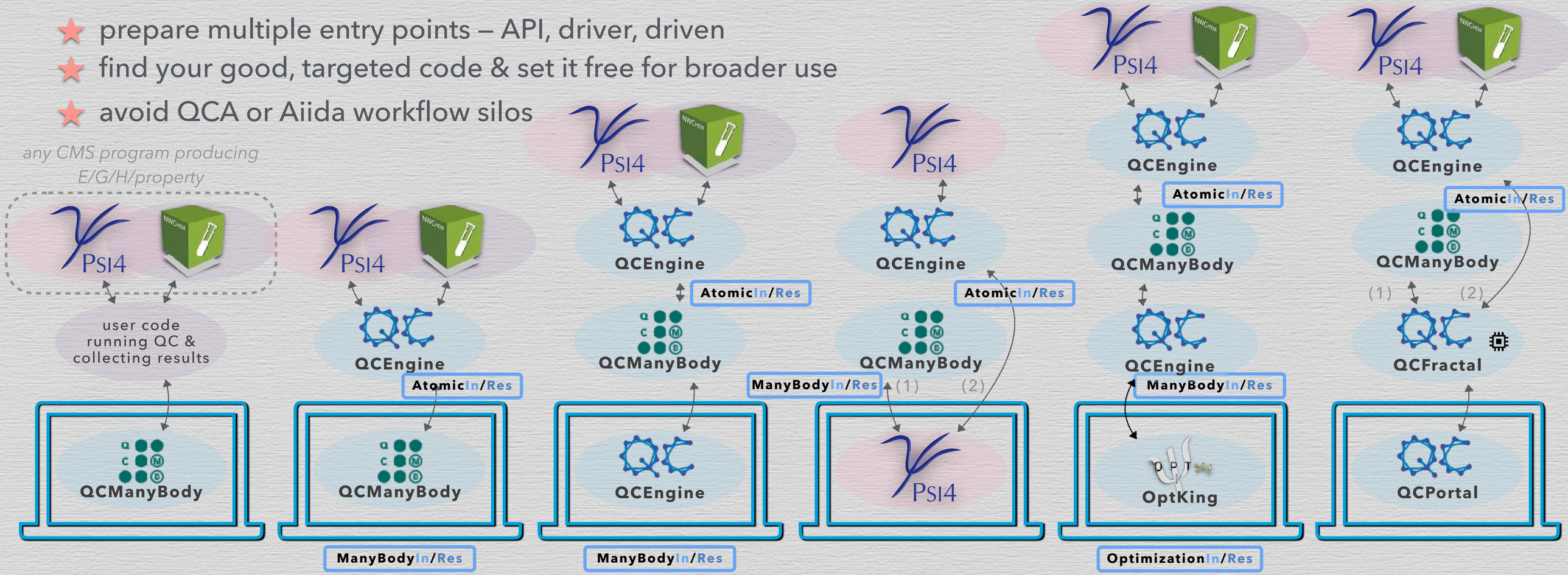
cluster_results: full results for subsystems

{cluster id: **AtomicResult** }

INTEGRATING QCMANYBODY

multiple entry points for flexibility

- prepare multiple entry points – API, driver, driven
- find your good, targeted code & set it free for broader use
- avoid QCA or Aiida workflow silos



QCManyBody core interface

- user-prov. serial/parallel
- user runs CMS backends

QCManyBody high-level interface

- serial
- multi CMS backends

QCEngine procedure

- serial
- multi CMS backends

Psi4 distributed driver

- serial or parallel
- only Psi4 backend

Optimizer backend

- serial
- multi CMS backends

QCFractal service

- parallel
- multi CMS backends

STANDARD SUITE

a framework for systematic testing

SOMEONE CONTRIBUTES A MODULE ADDING OR AFFECTING A METHOD

- Does it adhere to **QCSCHEMA** inputs & returns?
- Is it returning as many **PROPERTIES** as possible?
- Does it **MATCH** other modules or programs?
- Does it work with all **REFERENCES** or give a sensible error message?
- Does it work for all **DERIVATIVES** and match finite difference or give a sensible error message?
- Is the intended module getting called by **DEFAULT**?

MOST QC TESTS ARE FOR CORRECTNESS OF SENSIBLE INPUTS

- Many QC calcs aren't run by specialists anymore. Could be naive users, workflows, or AI. Best to detect **UNFEASIBLE** calcs & exit ASAP.
- Role for tests with **SYSTEMATIC** coverage of many QC methods
- Help new contributions check correctness & **INTEGRATION** with Psi4 plumbing. Currently 3 mols; could add more to handle edge cases like ghost atoms or no beta e^- or new modules like ECP.
- **REFERENCE** values & QCSchema properties contracts shared by Psi4, QCEngine, QCDB

STANDARD SUITE

★ automating systematic coverage & test-driven dev for correctness, schema compliance, completeness, & resiliance

PROGRAM

Psi4

5132 + 671

⟨CC|CC⟩

1158

NWChem

642

GAMESS

496

MRCC

66 tests

METHOD & DERIV

```
#  <<<  HF Energy
#  <<<  HF Gradient
#  <<<  HF Hessian
#  <<<  MP2 Energy
#  <<<  MP2 Gradient
#  <<<  MP2 Hessian
#  <<<  MP2.5 Energy
#  <<<  MP2.5 Gradient
#  <<<  MP3 Energy
#  <<<  MP3 Gradient
#  <<<  MP4(SDQ) Energy
#  <<<  MP4 Energy
#  <<<  ZAPT2 Energy
#  <<<  CISD Energy
#  <<<  QCISD Energy
#  <<<  QCISD(T) Energy
#  <<<  FCI Energy
#  <<<  REMP2 Energy
#  <<<  LCCD Energy
#  <<<  LCCD Gradient
#  <<<  LCCSD Energy
#  <<<  CEPA(1) Energy
#  <<<  CEPA(3) Energy
#  <<<  ACPF Energy
#  <<<  AQCC Energy
#  <<<  CCD Energy
#  <<<  CCD Gradient
#  <<<  BCCD Energy
#  <<<  CC2 Energy
#  <<<  CC2 Gradient
#  <<<  CCSD Energy
#  <<<  CCSD Gradient
#  <<<  CCSD(T) Energy
#  <<<  CCSD(T) Gradient
#  <<<  a-CCSD(T) Energy
#  <<<  BCCD(T) Energy
#  <<<  CC3 Energy
#  <<<  OMP3 Energy
#  <<<  OMP3 Gradient
#  <<<  OREMP2 Energy
#  <<<  OREMP2 Gradient
#  <<<  OLCCD Energy
#  <<<  OLCCD Gradient
#  <<<  MP2-F12 Energy
#  <<<  SVWN Energy
#  <<<  SVWN Gradient
#  <<<  SVWN Hessian
#  <<<  PBE Energy
#  <<<  PBE Gradient
#  <<<  B3LYP Energy
#  <<<  B3LYP Gradient
#  <<<  wB97X Energy
#  <<<  wB97X Gradient
#  <<<  B2PLYP Energy
#  <<<  B2PLYP Gradient
```

58 mtd+deriv

REFERENCE, INTEGRALS, MODULE

```
[ccsd  rhf  conv fc   : * ccenergy-adz-ene0] PASSED            [  2%]
[ccsd  uhf  conv fc   : * ccenergy-adz-ene0] PASSED            [  4%]
[ccsd rohf  conv fc sd: * ccenergy-adz-ene0] PASSED            [  6%]
[ccsd rohf  conv fc sc:   ccenergy-adz-ene0] PASSED            [  8%]
[ccsd  rhf  conv ae   : * ccenergy-adz-ene0] PASSED            [ 10%]
[ccsd  uhf  conv ae   : * ccenergy-adz-ene0] PASSED            [ 13%]
[ccsd rohf  conv ae sd: * ccenergy-adz-ene0] PASSED            [ 15%]
[ccsd rohf  conv ae sc:   ccenergy-adz-ene0] PASSED            [ 17%]
[ccsd  rhf  conv fc   :   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 19%]
[ccsd  uhf  conv fc   :   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 21%]
[ccsd rohf  conv fc sc:   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 23%]
[ccsd  rhf  conv ae   :   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 26%]
[ccsd  uhf  conv ae   :   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 28%]
[ccsd rohf  conv ae sc:   mrcc    -adz-ene0] SKIPPED (No mrcc) [ 30%]
[ccsd  rhf  conv fc   :   fnocc   -adz-ene0] PASSED            [ 32%]
[ccsd  rhf  conv ae   :   fnocc   -adz-ene0] PASSED            [ 34%]
[ccsd  rhf    df fc   : * fnocc   -adz-ene0] PASSED            [ 36%]
[ccsd  rhf    df ae   : * fnocc   -adz-ene0] PASSED            [ 39%]
[ccsd  rhf pk/df fc   :   fnocc   -adz-ene0] PASSED            [ 41%]
[ccsd  rhf pk/df ae   :   fnocc   -adz-ene0] PASSED            [ 43%]
[ccsd  rhf cd/df fc   :   fnocc   -adz-ene0] PASSED            [ 45%]
[ccsd  rhf cd/df ae   :   fnocc   -adz-ene0] PASSED            [ 47%]
[ccsd  rhf    cd fc   : * fnocc   -adz-ene0] PASSED            [ 50%]
[ccsd  rhf    cd ae   : * fnocc   -adz-ene0] PASSED            [ 52%]
[ccsd  rhf pk/cd fc   :   fnocc   -adz-ene0] PASSED            [ 54%]
[ccsd  rhf pk/cd ae   :   fnocc   -adz-ene0] PASSED            [ 56%]
[ccsd  rhf    df fc   :   dfocc   -adz-ene0] PASSED            [ 58%]
[ccsd  uhf    df fc   :   dfocc   -adz-ene0] PASSED            [ 60%]
[ccsd  rhf    df ae   :   dfocc   -adz-ene0] PASSED            [ 63%]
[ccsd  uhf    df ae   :   dfocc   -adz-ene0] PASSED            [ 65%]
[ccsd  rhf pk/df fc   :   dfocc   -adz-ene0] PASSED            [ 67%]
[ccsd  uhf pk/df fc   :   dfocc   -adz-ene0] PASSED            [ 69%]
[ccsd  rhf pk/df ae   :   dfocc   -adz-ene0] PASSED            [ 71%]
[ccsd  uhf pk/df ae   :   dfocc   -adz-ene0] PASSED            [ 73%]
[ccsd  rhf cd/df fc   :   dfocc   -adz-ene0] PASSED            [ 76%]
[ccsd  uhf cd/df fc   :   dfocc   -adz-ene0] PASSED            [ 78%]
[ccsd  rhf cd/df ae   :   dfocc   -adz-ene0] PASSED            [ 80%]
[ccsd  uhf cd/df ae   :   dfocc   -adz-ene0] PASSED            [ 82%]
[ccsd  rhf    cd fc   :   dfocc   -adz-ene0] PASSED            [ 84%]
[ccsd  uhf    cd fc   :   dfocc   -adz-ene0] PASSED            [ 86%]
[ccsd  rhf    cd ae   :   dfocc   -adz-ene0] PASSED            [ 89%]
[ccsd  uhf    cd ae   :   dfocc   -adz-ene0] PASSED            [ 91%]
[ccsd  rhf pk/cd fc   :   dfocc   -adz-ene0] PASSED            [ 93%]
[ccsd  uhf pk/cd fc   :   dfocc   -adz-ene0] PASSED            [ 95%]
[ccsd  rhf pk/cd ae   :   dfocc   -adz-ene0] PASSED            [ 97%]
[ccsd  uhf pk/cd ae   :   dfocc   -adz-ene0] PASSED            [100%]
```

5132 pytests

RETURN, CALCINFO, PROPERTIES

```
[ccsd  rhf pk/cd ae] wfn.............. PASSED
[ccsd  rhf pk/cd ae] return............ PASSED
[ccsd  rhf pk/cd ae] nbasis wfn........ PASSED
[ccsd  rhf pk/cd ae] nmo wfn........... PASSED
[ccsd  rhf pk/cd ae] nalpha wfn........ PASSED
[ccsd  rhf pk/cd ae] nbeta wfn......... PASSED
[ccsd  rhf pk/cd ae] wfn HF TOTAL ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn HF TOTAL ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 CORRELATION ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 CORRELATION ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 TOTAL ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 TOTAL ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 SAME-SPIN CORRELATION ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 SAME-SPIN CORRELATION ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 SINGLES ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 SINGLES ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 DOUBLES ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 DOUBLES ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn MP2 OPPOSITE-SPIN CORRELATION ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn MP2 OPPOSITE-SPIN CORRELATION ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn HF TOTAL ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn HF TOTAL ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CCSD CORRELATION ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CCSD CORRELATION ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CCSD TOTAL ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CCSD TOTAL ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CCSD SAME-SPIN CORRELATION ENERGY SKIP
[ccsd  rhf pk/cd ae] wfn CCSD SINGLES ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CCSD SINGLES ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CCSD DOUBLES ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CCSD DOUBLES ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CCSD OPPOSITE-SPIN CORRELATION ENERGY SKIP
[ccsd  rhf pk/cd ae] wfn CURRENT CORRELATION ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CURRENT CORRELATION ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CURRENT ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CURRENT ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn SCF TOTAL ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn SCF TOTAL ENERGY v. CONV PASSED
[ccsd  rhf pk/cd ae] wfn CURRENT REFERENCE ENERGY PASSED
[ccsd  rhf pk/cd ae] wfn CURRENT REFERENCE ENERGY v. CONV PASSED
```

7700 reference values

COMMUNICATING QC CAPABILITIES

a modular framework complicates the question and the answer

USER: IS A METHOD AVAILABLE?

MP2?
yes

open-shell?
sometimes

density-fitted?
most always

frozen-core?
usually

gradients?
often

by fast-coder?
depends

ECPs ok?
usually

w/ solvation?
possibly

any special keyword to access?

LORI: ANSWERS BASED ON PSI4 AS A WHOLE

COMMUNICATING QC CAPABILITIES

a modular framework complicates the question and the answer

USER: IS A METHOD AVAILABLE?

LORI: ANSWERS BASED ON PSI4 AS A WHOLE

USER: NO, THE INTERSECTION OF MY REQUIREMENTS

- Answers to this usually belong in documentation, but local code PRs rarely revise broad documentation.
- Since the Standard Suite loops over many of these QC characteristics, as well as analytic vs. findif derivatives and default vs. directed modules, Psi4 leverages the large quantity of runs to robustly report capabilities.

COMMUNICATING QC CAPABILITIES

auto-generate documentation tables from Standard Suite tests

Detailed capabilities of the DFMP2 module. "✓" runs analytically. Single underline "✓" is default module when QC_MODULE unspecified. Double underline "✓" is default algorithm type when type selector (e.g., CC_TYPE) unspecified.

		QC_MODULE=DFMP2 Capabilities																					
	REFERENCE →	Restricted (RHF)						Unrestricted (UHF)						Restricted Open (ROHF)					
name ↓ →		energy()			gradient()[3]			energy()			gradient()[3]			energy()			gradient()[3]		
	type[1] ↓ →	CV	DF	CD	CV	DF	CD	CV	DF	CD	CV	DF	CD	CV	DF	CD	CV	DF	CD
	FREEZE_CORE[2] →	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F	A F
mp2[4]	MP2_TYPE		✓ ✓			✓ ✓			✓ ✓						✓ ✓				

A focused module. Double-underlines indicate DF ints and this module are always default.

- A Standard Suite run dumps info from each test into a file, then a script processes these into various human-readable & reStructuredText tables. Hook into docs build.
- Functions as single-source-of-truth for non-exotic mtd cap. in Psi4.

COMMUNICATING QC CAPABILITIES

auto-generate documentation tables from Standard Suite tests

- A Standard Suite run dumps info from each test into a file, then a script processes these into various human-readable & reStructuredText tables. Hook into docs build.
- Functions as single-source-of-truth for non-exotic mtd cap. in Psi4.

Detailed capabilities of the SCF module. "✓" runs analytically. Single underline "✓" is defau

□	□	QC_MODULE=SCF Capabilities															
□	REFERENCE →	Restricted (RHF)															
name ↓ →	□	energy()						gradient()[3]						hessian()[4]			
□	type[1] ↓ →	CV		DF		CD		CV		DF		CD		CV		DF		CD	
□	FREEZE_CORE[2] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
hf	SCF_TYPE	✓		✓		✓		✓		✓				✓		✓			
svwn, LSDA DFT	SCF_TYPE	✓		✓		✓		✓		✓				✓		✓			
pbe, GGA DFT	SCF_TYPE	✓		✓		✓		✓		✓									
b3lyp, Hybrid DFT	SCF_TYPE	✓		✓		✓		✓		✓									
wb97x, LRC DFT	SCF_TYPE	✓		✓				✓		✓									
b2plyp, DH DFT[5]	SCF_TYPE	✓	✓	✓	✓	✓	✓												

Energies available for all DFT rungs. DF default. No CD for LRC.

COMMUNICATING QC CAPABILITIES

auto-generate documentation tables from Standard Suite tests

- A Standard Suite run dumps info from each test into a file, then a script processes these into various human-readable & reStructuredText tables. Hook into docs build.
- Functions as single-source-of-truth for non-exotic mtd cap. in PSI4.

Capabilities of Psi4, including details of overlapping modules. "✓" runs analytically. "::" runs derivative with internal finite difference. Single underline "✓" or "::" is default module when QC_MODULE unspecified. Double underline "✓" or "::" is default algorithm type when type selector (e.g., CC_TYPE) unspecified.

□	QC_MODULE ↓	Psi4 Capabilities																															
□	REFERENCE →	Restricted (RHF)										Unrestricted (UHF)										Restricted Open (ROHF)											
name ↓ →	□	energy()				gradient()[3]						energy()				gradient()[3]						energy()				gradient()[3]							
type[1] ↓ →	□	CV		DF		CD		CV		DF		CD		CV		DF		CD		CV		DF		CD		CV		DF		CD			
□	FREEZE_CORE[2] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
hf		✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::		✓		✓		✓		✓	
SCF_TYPE	SCF	✓		✓		✓		✓		✓				✓		✓		✓		✓		✓				✓		✓		✓			
mp2		✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	::	::
MP2_TYPE	DETCI	✓	✓																														
	DFMP2[4]			✓	✓					✓	✓					✓	✓											✓	✓				
	FNOCC	✓	✓																														
	OCC	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓		
mp2.5		✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::								
MP_TYPE	OCC	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓		✓	✓										
ccsd		✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓			✓	::		
CC_TYPE	CCENERGY	✓	✓					✓						✓	✓					✓						✓	✓			✓			
	FNOCC	✓	✓	✓	✓	✓	✓																										
	MRCC	✓	✓											✓	✓											✓	✓						
	OCC			✓	✓	✓	✓		✓	✓						✓	✓	✓	✓			✓	✓										
ccsd(t)		✓	✓	✓	✓	✓	✓	::	::	✓	✓	::	::	✓	✓					::	::					✓	✓			::	::		
CC_TYPE	CCENERGY[6]	✓	✓											✓	✓											✓	✓						
	FNOCC	✓	✓	✓	✓	✓	✓																										
	MRCC	✓	✓											✓	✓											✓	✓						
	OCC			✓	✓	✓	✓		✓	✓																							

Per-module and cummulative capabilities reveal default and users' options.

COMMUNICATING QC CAPABILITIES

auto-generate documentation tables from Standard Suite tests

hf	SCF_TYPE	✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::	
mp2	MP2_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	::	::	::	::	::	::
mp2.5	MP_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::												
mp3	MP_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::												
mp4(sdq)	MP_TYPE	✓	✓																																		
mp4, mpn[10]	MP_TYPE	✓	✓																																		
zapt2, zaptn[10]	MP_TYPE																									✓	✓										
cisd, cin[10]	CI_TYPE	✓	✓																							✓	✓										
qcisd	CI_TYPE	✓	✓																																		
qcisd(t)	CI_TYPE	✓	✓																																		
fci	CI_TYPE	✓	✓																							✓	✓										
remp2	CC_TYPE	✓	✓	✓	✓	✓	✓							✓	✓	✓	✓	✓	✓																		
lccd	CC_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::												
lccsd, cepa(0)	CC_TYPE	✓	✓																																		
cepa(1)	CC_TYPE	✓	✓																																		
cepa(3)	CC_TYPE	✓	✓																																		
acpf	CC_TYPE	✓	✓																																		
aqcc	CC_TYPE	✓	✓																																		
ccd	CC_TYPE			✓	✓	✓	✓			✓	✓	::	::			✓	✓	✓	✓			✓	✓	::	::												
bccd	CC_TYPE	✓	✓											✓	✓											✓	✓										
cc2	CC_TYPE	✓	✓					✓	::					✓	✓					::	::					✓	✓					::	::				
ccsd	CC_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓					✓	::				
ccsd(t)	CC_TYPE	✓	✓	✓	✓	✓	✓	::	::	✓	✓	::	::	✓	✓					::	::					✓	✓					::	::				
a-ccsd(t)[11]	CC_TYPE	✓	✓	✓	✓	✓	✓																														
bccd(t)	CC_TYPE	✓	✓											✓	✓											✓	✓										
cc3	CC_TYPE	✓	✓											✓	✓											✓	✓										
omp2	MP2_TYPE	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::
omp2.5	MP_TYPE	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::
omp3	MP_TYPE	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::
oremp2	CC_TYPE	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::
olccd	CC_TYPE	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::	✓		✓	✓	✓	✓	✓		✓	✓	::	::
svwn, LSDA DFT	SCF_TYPE	✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::													
pbe, GGA DFT	SCF_TYPE	✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::													
b3lyp, Hybrid DFT	SCF_TYPE	✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::													
wb97x, LRC DFT	SCF_TYPE	✓		✓				✓		✓				✓		✓				✓		✓															
b2plyp, DH DFT[12]	SCF_TYPE	✓	✓	✓	✓	✓	✓							✓	✓	✓	✓	✓	✓																		

I've more confidence in this whole-Psi4 capabilities table than a hand-updated one.

- A Standard Suite run dumps info from each test into a file, then a script processes these into various human-readable & reStructuredText tables. Hook into docs build.
- Functions as single-source-of-truth for non-exotic mtd cap. in Psi4.

COMMUNICATING QC CAPABILITIES

auto-generate documentation tables from Standard Suite tests

Capabilities of QCDB, including details of overlapping modules. "✓" runs analytically. "!" produces nonstandard results.

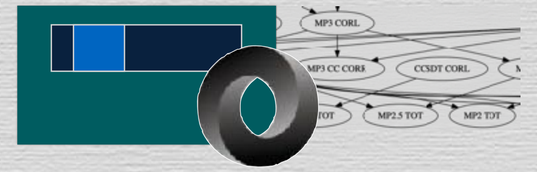
□	qc_module ↓	QCDB Capabilities																	
□	scf__reference →	Restricted (RHF)						Unrestricted (UHF)						Restricted Open (ROHF)					
name ↓ →	□	E		G		H		E		G		H		E		G		H	
□	freeze_core[1] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
mp2	CFOUR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	!	!
	GAMESS-DDI	✓	✓	✓	✓			✓	✓	✓	✓								
	GAMESS-IMS	✓	✓	✓	✓														
	GAMESS-SERIAL	✓	✓	✓	✓			✓	✓	✓	✓			✓	✓	!	!		
	NWChem-DIRECTMP2	✓	✓																
	NWChem-MP2GRAD	✓	✓	✓	✓			✓	✓	✓	✓								
	NWChem-TCE	✓	✓					✓	✓					!	!	!	!		
	Psi4-FNOCC	✓	✓																
	Psi4-OCC	✓	✓	✓				✓	✓	✓				✓	✓				
ccsd(t)	CFOUR-ECC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	CFOUR-NCC	✓	✓	✓	!														
	CFOUR-VCC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	GAMESS	✓	✓																
	NWChem-CC	✓	✓																
	NWChem-TCE	✓	✓					✓	✓					!	!	!			
	Psi4-CCENERGY	✓	✓					✓	✓					✓	✓				
	Psi4-FNOCC	✓	✓																
	Psi4-MRCC	✓	✓					✓	✓					✓	✓				

[1] Active orbital values to the right: all-electron `A` and frozen-core `F`.

Also useable across QC programs to check answers and show interfaced capabilities.

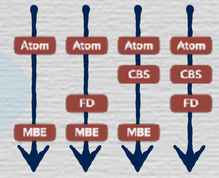
- A Standard Suite run dumps info from each test into a file, then a script processes these into various human-readable & reStructuredText tables. Hook into docs build.
- Functions as single-source-of-truth for non-exotic mtd cap. in Psi4.

★Automate the assessment & conveyance of detailed capabilities to users, lest contrib. from alternate modules become overwhelming



LIBINT

QCEngine

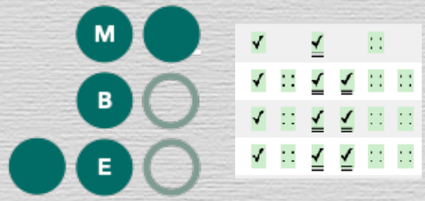


DFT-D3

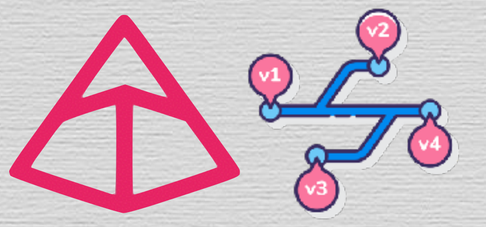
Psi4
CONDA INSTALL

Psi4

OPEN-SOURCE QUANTUM CHEMISTRY



VERSIONING

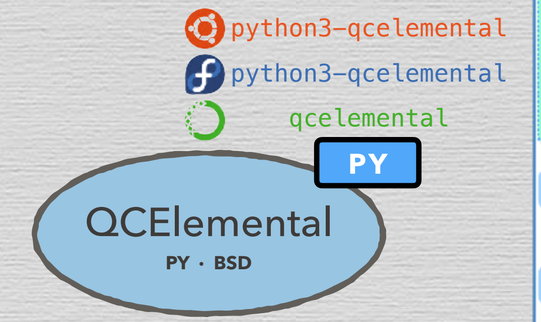


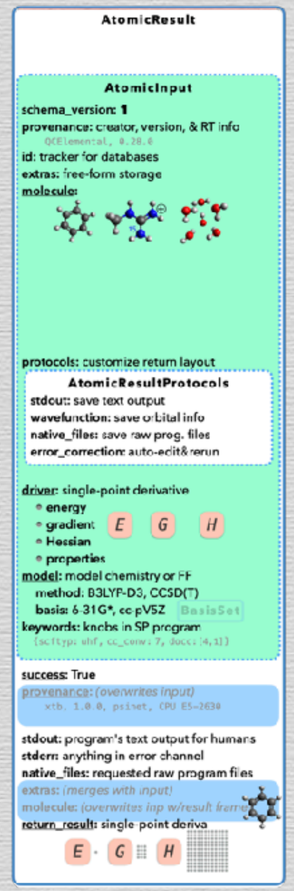
QCSCHEMA V2

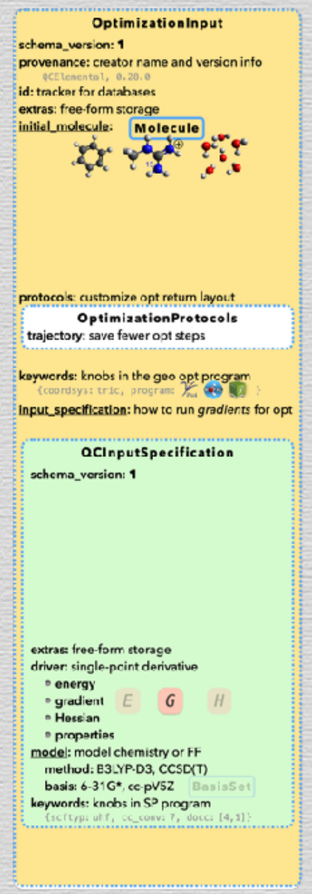
A QCSchema UPGRADE STORY

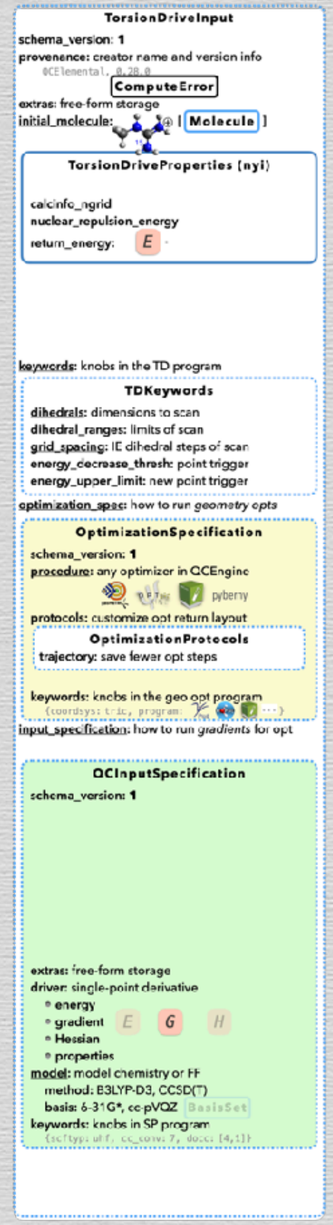
modular isn't independent

- **DEVELOPED** by MolSSI with early community advice 2017–2019. Tweaks and new procedures since.
- used in Psi4 for **PROGRAMMATIC** input/output
- used in Psi4 for **INTER-MODULAR** communication
- implemented as Pydantic **CLASSES** in QCElemental
- MolSSI prioritized building a **DEEP** science-results-focused stack atop their projects (QCSchema & QCArchive) to exercise and polish the software.
- Ongoing development revealed areas for **IMPROVEMENT** in schema predictability, removing redundancy, & aiding restartability, esp. for procedures.







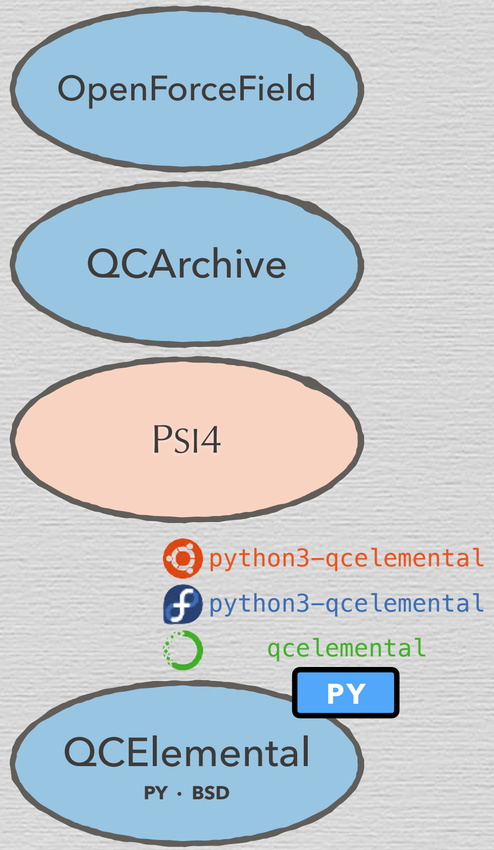


Overlapping or overwritten schema shown by color variations

A QCSchema UPGRADE STORY

modular isn't independent

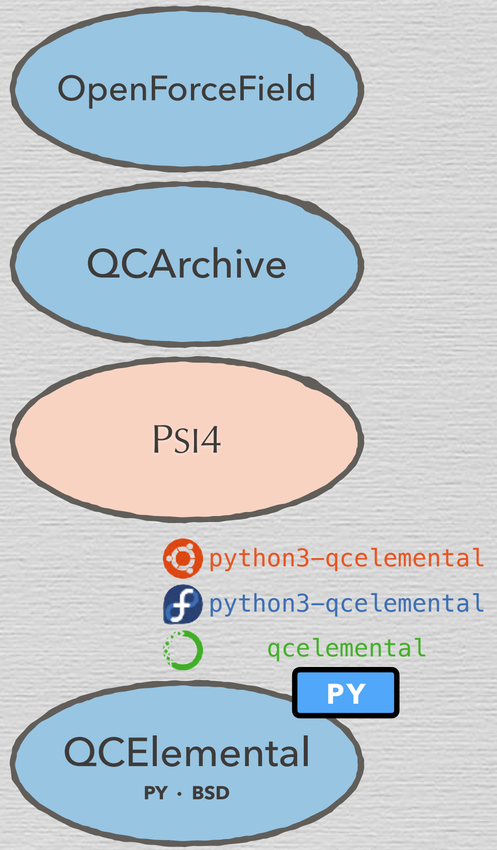
- **DEVELOPED** by MolSSI with early community advice 2017–2019. Tweaks and new procedures since.
- used in Psi4 for **PROGRAMMATIC** input/output
- used in Psi4 for **INTER-MODULAR** communication
- implemented as Pydantic **CLASSES** in QCElemental
- MolSSI prioritized building a **DEEP** science-results-focused stack atop their projects (QCSchema & QCArchive) to exercise and polish the software.
- Ongoing development revealed areas for **IMPROVEMENT** in schema predictability, removing redundancy, & aiding restartability, esp. for procedures.
- **Pydantic V2** API released June 2023, which is a mild upgrade *except* it forbids mixing v1/v2 models, requiring concurrent full-stack migration.

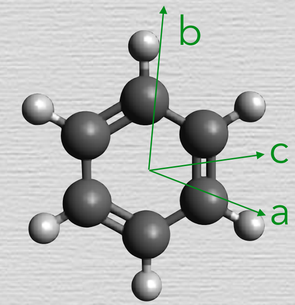


A QCSchema UPGRADE STORY

modular isn't independent

- **DEVELOPED** by MolSSI with early community advice 2017–2019. Tweaks and new procedures since.
- used in Psi4 for **PROGRAMMATIC** input/output
- used in Psi4 for **INTER-MODULAR** communication
- implemented as Pydantic **CLASSES** in QCElemental
- MolSSI prioritized building a **DEEP** science-results-focused stack atop their projects (QCSchema & QCArchive) to exercise and polish the software.
- Ongoing development revealed areas for **IMPROVEMENT** in schema predictability, removing redundancy, & aiding restartability, esp. for procedures.
- **Pydantic V2** API released June 2023, which is a mild upgrade *except* it forbids mixing v1/v2 models, requiring concurrent full-stack migration.
- At a CECAM meeting last year, I learned that QCSchema would be useful to a new community if only `Molecule` included **UNIT CELL** specification.



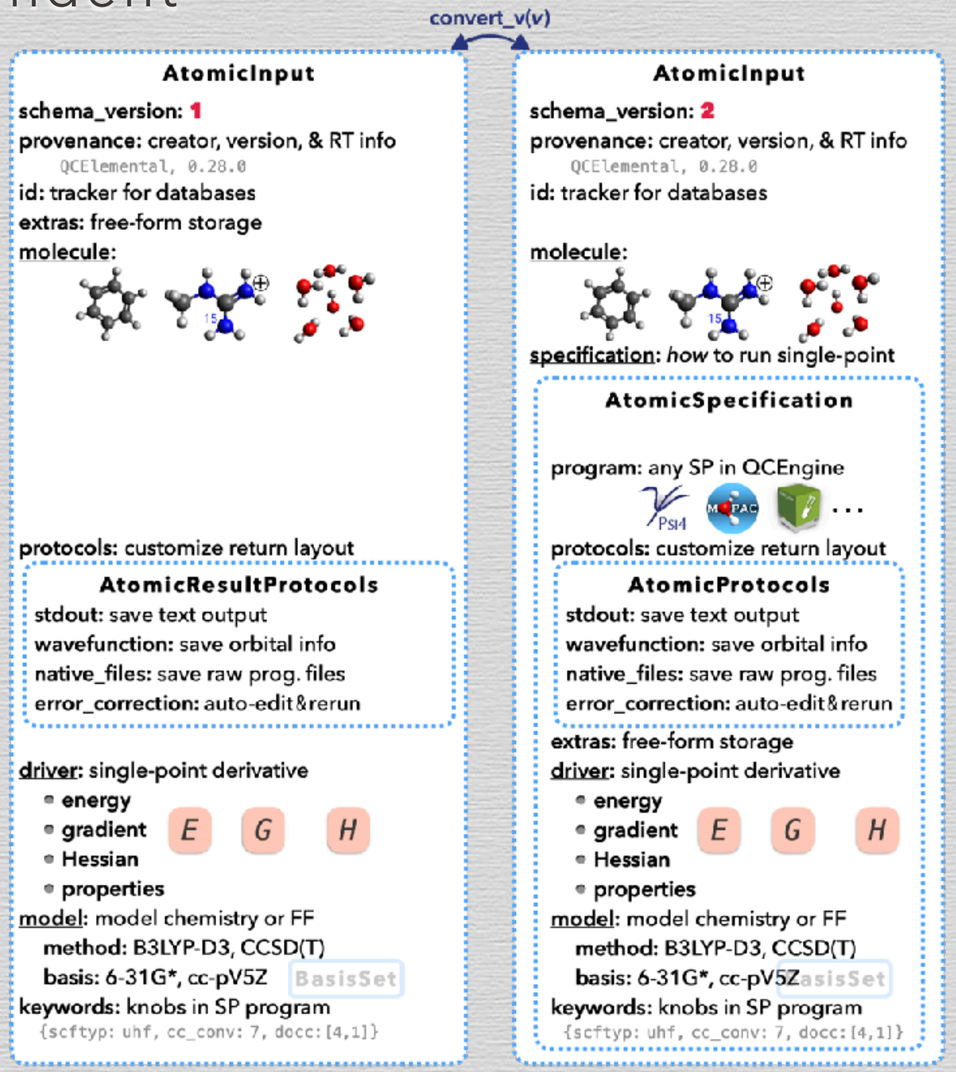


A QCSchema UPGRADE STORY

modular isn't independent

Technical not content, upgrade to **v2** in Fall 2024

- Refactored **AtomicInput**/**Result** & procedure schema layouts to be more composable, reuseable, & intuitive.
- Updated schema v2 models to Pydantic v2 API and provided converter functions.



A QCSchema UPGRADE STORY

modular isn't independent

Technical not content, upgrade to **v2** in Fall 2024

- Refactored **AtomicInput**/**Result** & procedure schema layouts to be more composable, reuseable, & intuitive.
- Updated schema v2 models to Pydantic v2 API and provided converter functions.
- To allow projects to update at their own pace while keeping single maintained codebase, QCSchema v1 & v2 are both accessible from single QCElemental and are both runable from single QCEngine.

ACCESS QCSchema v1

(PYDANTIC v1 API; c. 2019 field layout)

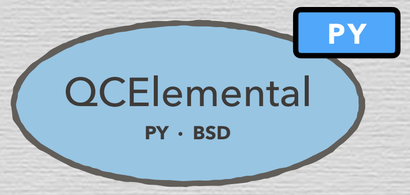
```
# longstanding until v0.70
> from qcelemental.models import Molecule

# longstanding until v0.70 but remove ASAP!
> from qcelemental.models.molecule import Molecule

# v0.50a1 until v1.0
> from qcelemental.models.v1 import Molecule
```

ACCESS QCSchema v2

(PYDANTIC v2 API; c. 2024 field layout)

```
# v0.50a1 onwards
> from qcelemental.models.v2 import Molecule

# v0.70a1 onwards
> from qcelemental.models import Molecule
```

Either v1 or v2 immediately accessible,
but no oblivious v1 code will break for 1y

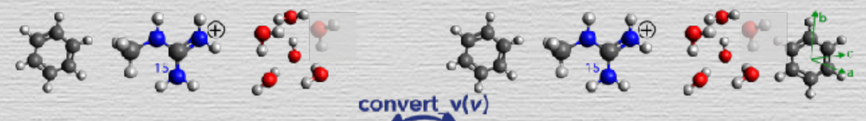


A QCSchema UPGRADE STORY

modular isn't independent

Technical not content, upgrade to **v2** in Fall 2024

- Refactored **AtomicInput**/**Result** & procedure schema layouts to be more composable, reuseable, & intuitive.
- Updated schema v2 models to Pydantic v2 API and provided converter functions.
- To allow projects to update at their own pace while keeping single maintained codebase, QCSchema v1 & v2 are both accessible from single QCElemental and are both runable from single QCEngine.
- Foundational **Molecule** & **BasisSet** will recieve small but useful additions from expressed need.



convert_v(v)

Molecule

schema_version: 2 *(used w/QCSk v1)*
provenance: creator info
id: tracker for databases
extras: free-form storage
name: user label
comment: further user text
symbols: elemental symb. for atoms
geometry: Cartesian coord. in Bohr
molecular_charge: net elst. charge
molecular_multiplicity: total mult.
masses: per atom atomic masses
real: per atom real vs. ghost/virtual
atom_labels: further per atom labels
atomic_numbers: per atom numbers
mass_numbers: per atom isotope #

fragments: group atoms into fragmts
fragment_charges: partitioned chg
fragment_multiplicities: ptn'd mult
fix_com: conserve input COM
fix_orientation: conserve input frame
fix_symmetry: use lesser point group
connectivity: bond info
validated: passed physics checks
identifiers: registration/classification

Identifiers

Molecule

schema_version: 3 *(used w/QCSk v2)*
provenance: creator info
id: tracker for databases
extras: free-form storage
name: user label
comment: further user text
symbols: elemental symb. for atoms
geometry: Cartesian coord. in Bohr
molecular_charge: net elst. charge
molecular_multiplicity: total mult.
masses: per atom atomic masses
real: per atom real vs. ghost/virtual
atom_labels: further per atom labels
atomic_numbers: per atom numbers
mass_numbers: per atom isotope #
abc: periodic box vectors *(nyi)*
fragments: group atoms into fragmts
fragment_charges: partitioned chg
fragment_multiplicities: ptn'd mult
fix_com: conserve input COM
fix_orientation: conserve input frame
fix_symmetry: use lesser point group
connectivity: bond info
validated: passed physics checks
identifiers: registration/classification

Identifiers

PY

QCElemental

PY · BSD

For technical reasons, Molecule v2 needs a difference, and periodic box vectors will be ideal.

A QCSchema UPGRADE STORY

modular isn't independent

Technical not content, upgrade to **v2** in Fall 2024

- Refactored **AtomicInput**/**Result** & procedure schema layouts to be more composable, reuseable, & intuitive.
- Updated schema v2 models to Pydantic v2 API and provided converter functions.
- To allow projects to update at their own pace while keeping single maintained codebase, QCSchema v1 & v2 are both accessible from single QCElemental and are both runable from single QCEngine.
- Foundational **Molecule** & **BasisSet** will recieve small but useful additions from expressed need.

- **PAYOFF** every project gets unstuck from Pydantic stasis.
- **PAYOFF** after v2 roll-out, QCSchema will be ready to add community-requested features. Hopefully news soon.
- **PAYOFF** established pathway to upgrade in future.

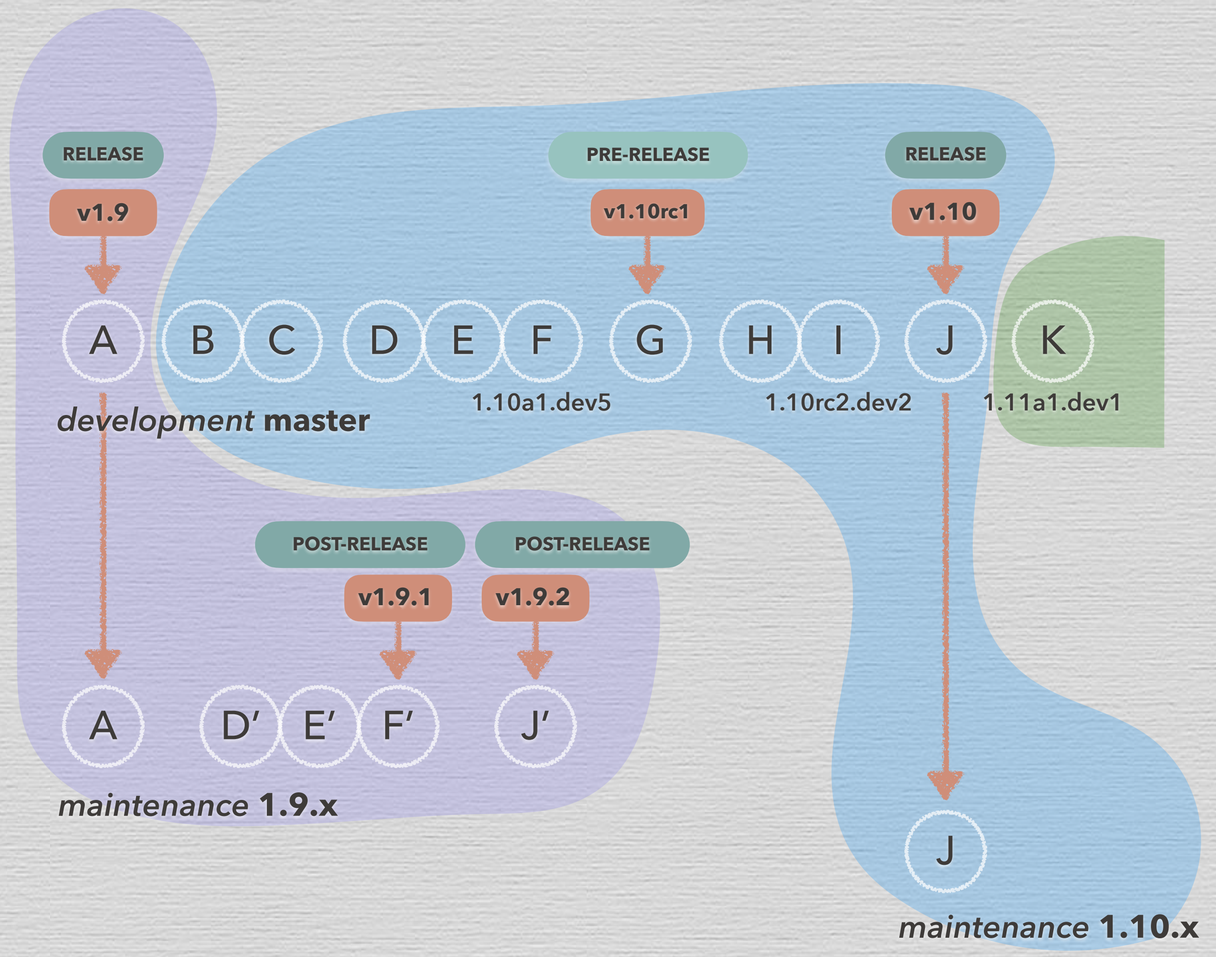
- ★ modularity can mean dependence in deep software stacks that needs careful guiding
- ★ generous acceptance of feature requests can lead to high-cohesion module but beyond your comfort
- ★ research demands implementing some things quickly for downstream, lest they be blocked. Research also demands your own dependencies be stable and not interrupted by new nonsense fields. Need some governance to balance these interests.

VERSIONING

ecosystems run on version compatibility

- **AUTOMATED** version reckoning after `git tag` signals a release. Upon `make`, info from `git describe` computes a unique, sortable version at every commit. Stamped into output.
- **CONDA** allows a package to advise any downstream consumers how many versions are binary compatible (`x` for semver or `x.x.x` more realistically).
- In an ecosystem arrangment, adequately tested and pushed to GitHub or even merged to master isn't sufficient for "**AVAILABLE**". A feature (or more importantly a bugfix or buildfix) isn't avail. until it's in a released version (or patched binary dist.) because far downstream it takes too much intervention to use a specialty changeset.
- Incline toward mercy on thine downstream and don't just fix things quickly but **PUBLISH** them.

VERSION COMPUTED EVERY `cmake --build` FROM GIT INFO



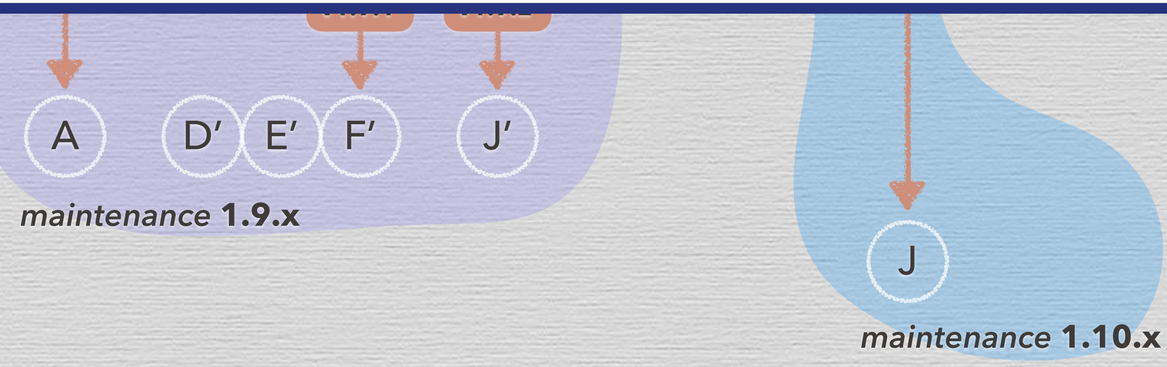
VERSIONING

ecosystems run on version compatibility

- **AUTOMATED** version reckoning after **`git tag`** signals a release. Upon **`make`**, info from **`git describe`** computes a unique, sortable version at every commit. Stamped into output.
- **CONDA** allows a package to advise any downstream consumers how many versions are binary compatible (**`x`** for semver or **`x.x.x`** more realistically).
- In an ecosystem arrangment, adequately tested and pushed to GitHub or even merged to master isn't sufficient for "**AVAILABLE**". A feature (or more importantly a bugfix or buildfix) isn't avail. until it's in a released version (or patched binary dist.) because far downstream it takes too much intervention to use a specialty changeset.
- Incline toward mercy on thine downstream and don't just fix things quickly but **PUBLISH** them.

VERSION COMPUTED EVERY `cmake --build` FROM GIT INFO

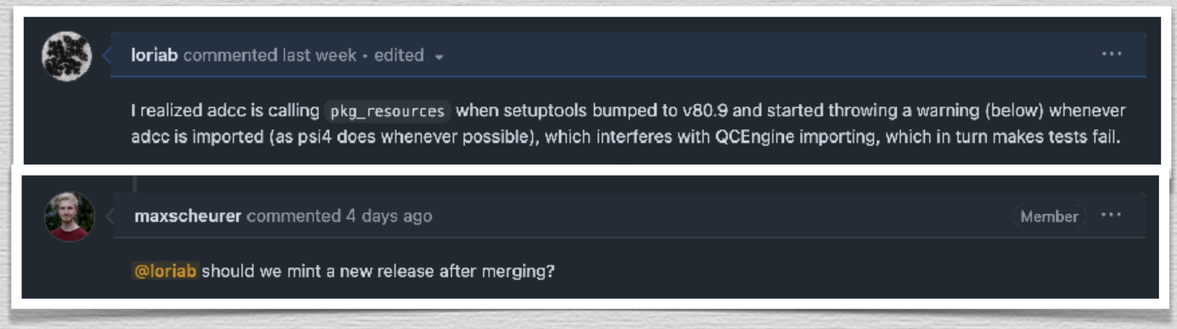
```
import psi4

print(psi4.__version__)
#> 1.10a1.dev119

print(psi4.version_formatter())
#> 1.10a1.dev119

print(psi4.version_formatter('all'))
#> 1.10a1.dev119 {f12_draft_rb4} e2cc1fd 1.9.0.999 dirty  1.9 <-- 1.10a1.dev119+e2cc1fd

print(psi4.version_formatter('branch: {branch}\nsortable: {versionlong}'))
#> branch: f12_draft_rb4
#> sortable: 1.10a1.dev119+e2cc1fd
```



VERSIONING

ecosystems run on version compatibility

- **AUTOMATED** version reckoning after **`git tag`** signals a release. Upon **`make`**, info from **`git describe`** computes a unique, sortable version at every commit. Stamped into output.
- **CONDA** allows a package to advise any downstream consumers how many versions are binary compatible (**`x`** for semver or **`x.x.x`** more realistically).
- In an ecosystem arrangment, adequately tested and pushed to GitHub or even merged to master isn't sufficient for "**AVAILABLE**". A feature (or more importantly a bugfix or buildfix) isn't avail. until it's in a released version (or patched binary dist.) because far downstream it takes too much intervention to use a specialty changeset.
- Incline toward mercy on thine downstream and don't just fix things quickly but **PUBLISH** them.



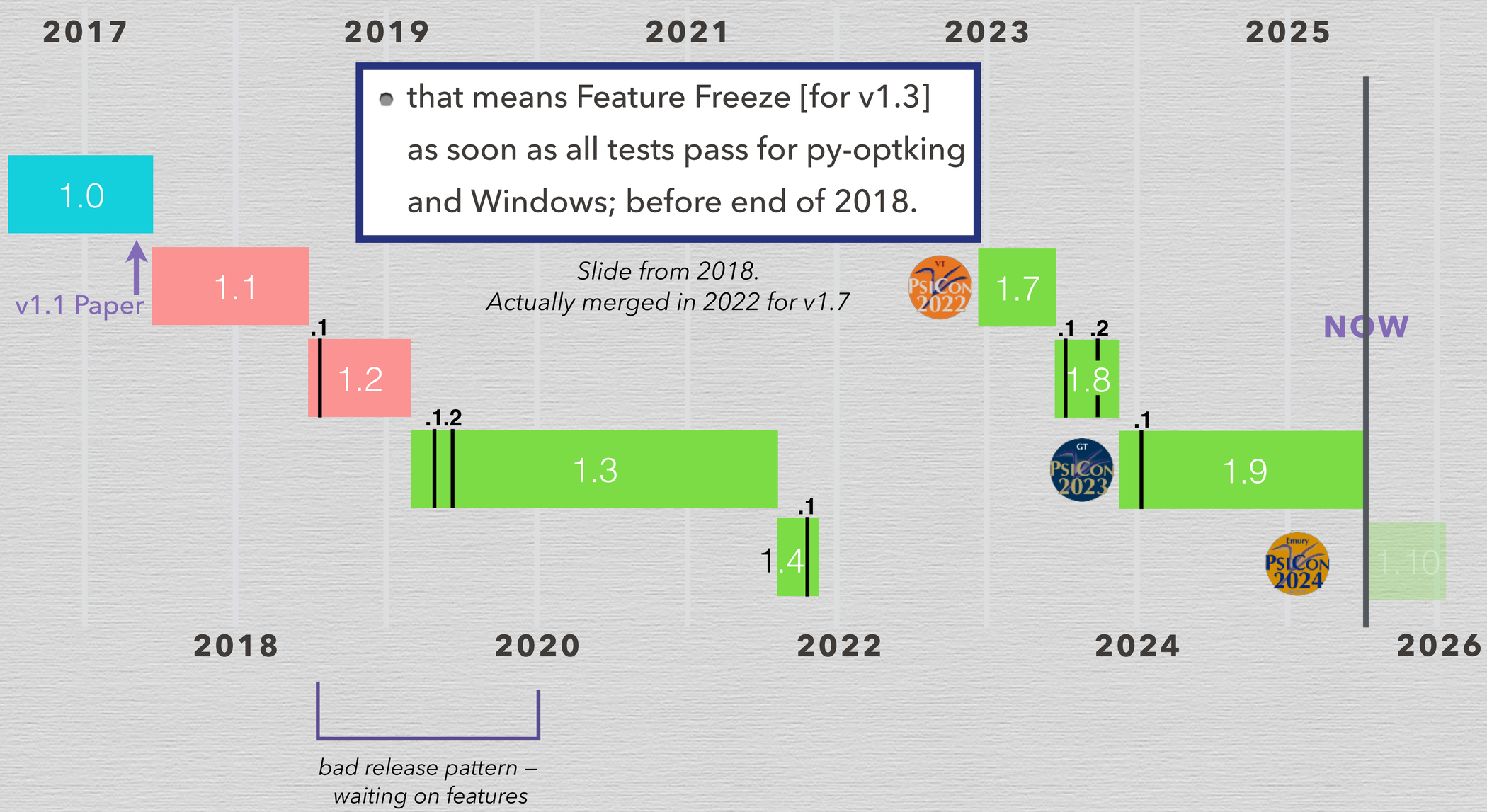
Bless the maintainer

Psi4 RELEASE HISTORY

what *not* to do

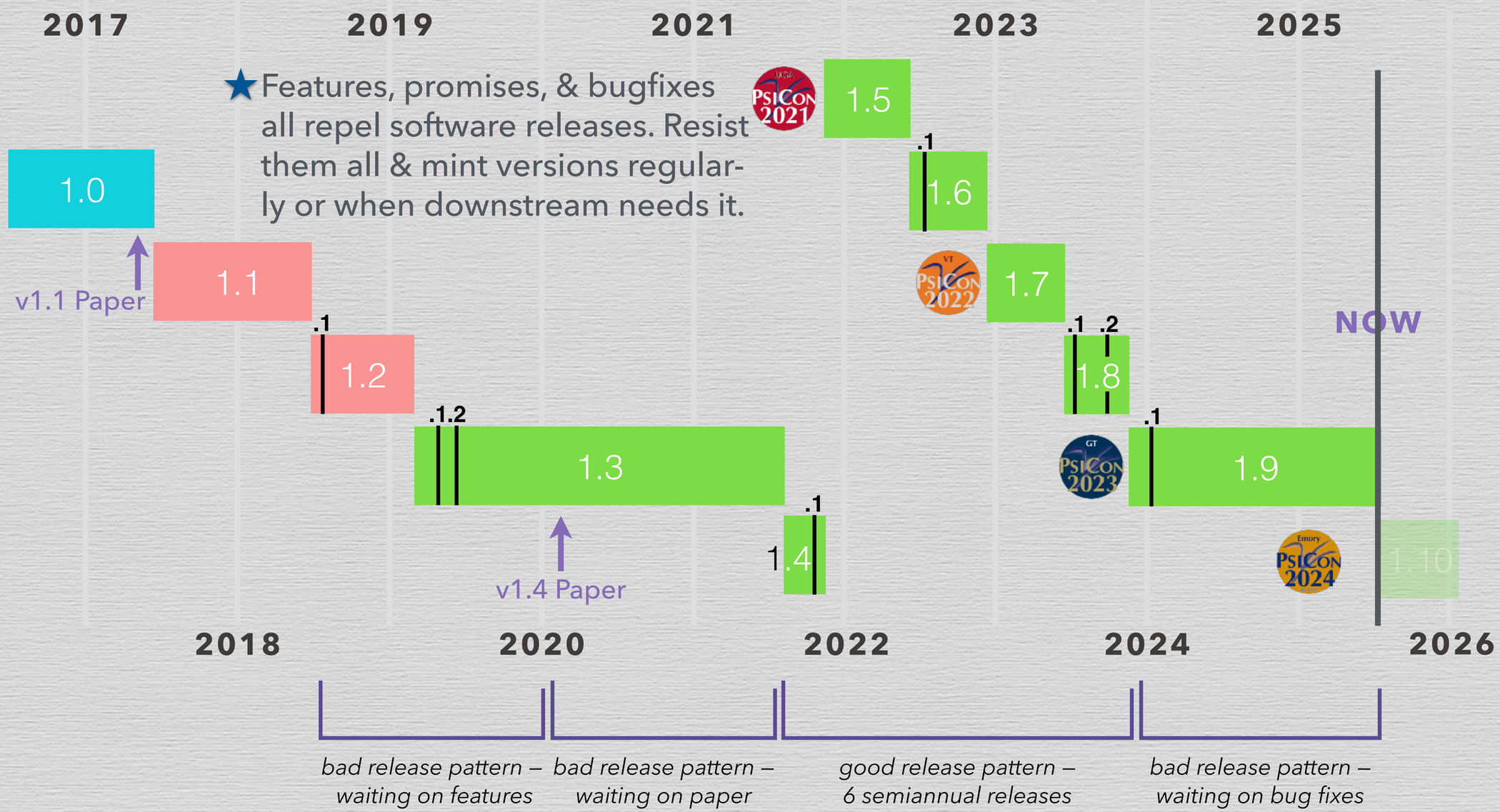
- that means Feature Freeze [for v1.3] as soon as all tests pass for py-optking and Windows; before end of 2018.

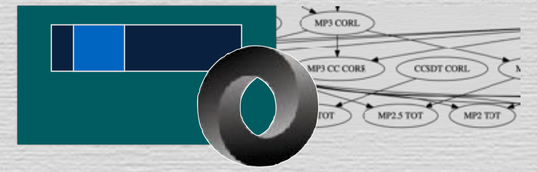
Slide from 2018.
Actually merged in 2022 for v1.7

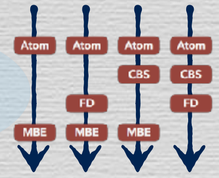


Psi4 RELEASE HISTORY

what *not* to do

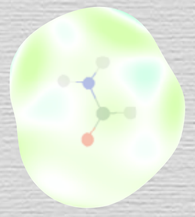






Psi4

OPEN-SOURCE QUANTUM CHEMISTRY



Features, promises, & bugfixes all repel software releases. Resist them all & mint versions regularly or when downstream needs it.

Generous acceptance of feature requests can lead to a high-cohesion module but beyond your comfort.

Prepare multiple entry points – API, driver, driven.

Mapping calculation input into a structured format can open new ways of interacting with users & workflows.

Automate the assessment & conveyance of detailed cap-abilities to users, lest contrib. from alternate modules become overwhelming.

Joining a packaging community can prod software engineering aspects.

Ship a Py reference implementation or MWE to aid downstream adoption

CI testing is both a repository's lifeline and anchor.

Find good separations of concern for software stack & build proj. around those missions.

Adapt established modular code with new interfaces.

Give your module the tools to convey its requirements, capabilities, & attributions.

Hold your interfaces and tests close, lest others write them for you.

Consume your own data standards to exercise them.

At the ecosys level, even simple dependencies logic can get overwhelming, so consolidate & encode.

Extract code into a module to concentrate performance optimizations.

Automate systematic coverage & test-driven dev for correctness, schema compliance, completeness, & resiliance for faster onboarding.

Decoupling into more cohesive internal modules is progress, too.

Beware that choice of license can limit uptake.

Avoid collecting into workflow silos like QCA or Aiida, as we have QC program silos.

Modularity can mean dependence in deep software stacks that needs careful guiding.

A SSOT deps file can broadcast fixes & be a path to recovery when the build breaks.

Getting a production-quality binary to users for multiple architectures can be fairly easy with conda-forge.

Prefer and define programmatic output.

Find your good, targeted code & set it free for broader use.

Provide a variety of installation routes & consolidate the choices.

Defer loading to runtime for licensing and modularity.

Defer ordering configuration to runtime & build broadly for consumers.

New syntax advice is more helpful at runtime than in docs. A grace period is a bonus.

Research demands implementing some things quickly for downstream, lest it be blocked. Research also demands one's dependencies be stable and not interrupted by new nonsense fields. Need some governance to balance these interests.

Bring the changing env of diff arch, compilers, dep ver into freq. contact w/code.

Aim for copy/paste install even for compiled software.

Use CI to assure developers always getting a full & proved working environment.

Interfacing two codes is rarely unilateral. Seize any opportunity to work with other devs in real time.

★ Features, promises, & bugfixes all repel software releases. Resist them all & mint versions regularly or when downstream needs it.

★ Generous acceptance of feature requests can lead to a high-cohesion module but beyond your comfort.

★ Prepare multiple entry points – API, driver, driven.

★ Mapping calculation input into a structured format can open new ways of interacting with users & workflows.

★ Automate the assessment & conveyance of detailed cap-abilities to users, lest contrib. from alternate modules become overwhelming.

★ Joining a packaging community can prod software engineering aspects.

★ Ship a Py reference implementation or MWE to aid downstream adopti

★ CI testing is both a repository's lifeline and anchor.

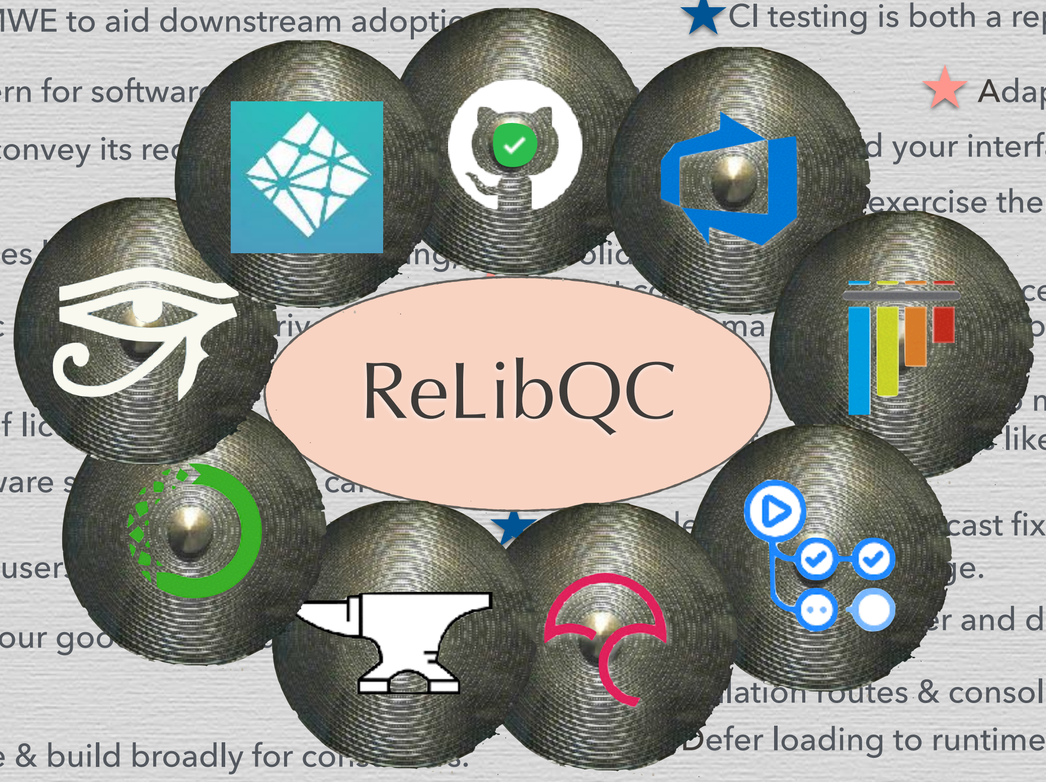
★ Find good separations of concern for softwar

★ Adapt established modular code with new interfaces.

★ Give your module the tools to convey its re

d your interfaces and tests close, lest others write them for you.

exercise them.



★ At the ecosys level, even simple dependencies

centrate performance optimizations.

★ Automate systematic

pleteness, & resiliance for faster onboarding.

more cohesive internal modules is progress, too.

★ Beware that choice of lic

like QCA or Aiida, as we have QC program silos.

★ Modularity can mean dependence in deep software s

cast fixes & be a path to recovery when the build breaks.

★ Getting a production-quality binary to user

ge.

★ Find your goo

er and define programmatic output.

lation routes & consolidate the choices.

★ Defer ordering configuration to runtime & build broadly for co

Defer loading to runtime for licensing and modularity.

★ New syntax advice is more helpful at runtime than in docs. A grace period is a bonus.

★ Research demands implementing some things quickly for downstream, lest it be blocked. Research also demands one's dependencies be stable and not interrupted by new nonsense fields. Need some governance to balance these interests.

★ Bring the changing env of diff arch, compilers, dep ver into freq. contact w/code.

★ Aim for copy/paste install even for compiled software.

★ Use CI to assure developers always getting a full & proved working environment.

★ Interfacing two codes is rarely unilateral. Seize any opportunity to work with other devs in real time.

★ Features, promises, & bugfixes all repel software releases. Resist them all & mint versions regularly or when downstream needs it.

★ Generous acceptance of feature requests can lead to a high-cohesion module but beyond your comfort.

★ Prepare multiple entry points – API, driver, driven.

★ Mapping calculation input into a structured format can open new ways of interacting with users & workflows.

★ Automate the assessment
users, lest contrib. from alt
are engineering aspects.

★ Ship a Py reference
d anchor.

★ Find good
lar code with new interfaces.

★ Give your
lest others write them for you.

★ At the ecosys level, eve

★
optimizations.
e for faster onboarding.

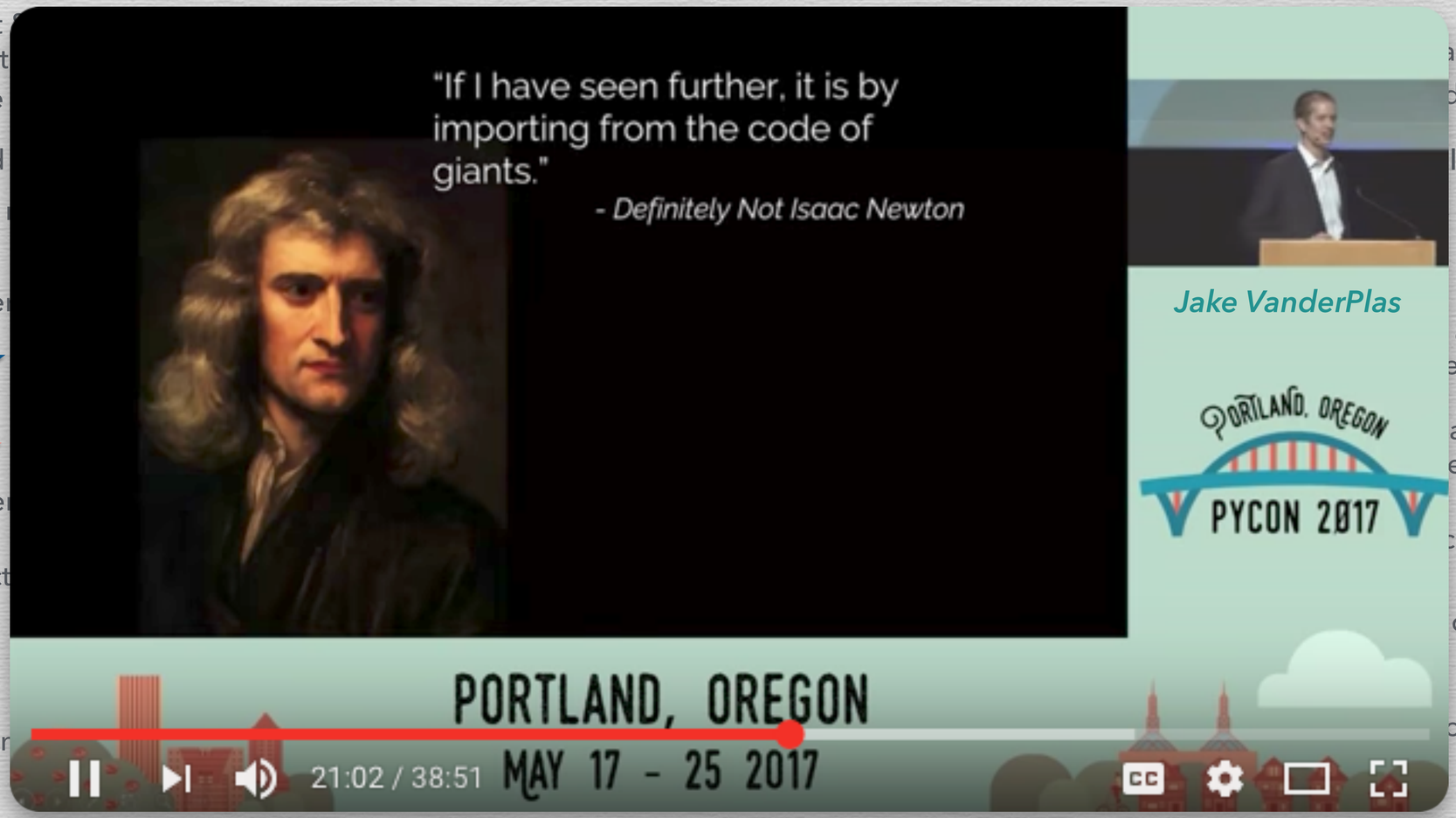
★
al modules is progress, too.

★ Modularity can mean depe
e have QC program silos.

★ Getting a product
covery when the build breaks.

output.

★ Defer ordering cor
odularity.



★ Research demands implementing some things quickly for downstream, lest it be blocked. Research also demands one's dependencies be stable and not interrupted by new nonsense fields. Need some governance to balance these interests.

★ Bring the changing env of diff arch, compilers, dep ver into freq. contact w/code.

★ Aim for copy/paste install even for compiled software.

★ Use CI to assure developers always getting a full & proved working environment.

★ Interfacing two codes is rarely unilateral. Seize any opportunity to work with other devs in real time.

Psi4

Jerome Gonthier
Berkeley → QC Ware

Rob Parrish
Stanford → QC Ware

Holger Kruse
Czech Acad. Sci.

Rollin King
Bethel

Alexander Sokolov
Ohio State

David Sherrill
GaTech

Lori Burns
GaTech

Asim Alenaizan
KFUPM

Maximilian Scheurer
Heidelberg → Covestro

Zach Glick
GaTech → Lavo

Jeff Schriber
Iona

Francesco Evangelista
Emory

Eugene DePrince
FSU

Fritz Schaefer
UGA

Justin Turney
UGA

Daniel Crawford
VaTech

Daniel Smith
MolSSI → Abiologics

Konrad Patkowski
Auburn

Ben Pritchard
MolSSI

Jonathon Misiewicz
VaTech

Ed Valeev
VaTech

Andy Simmonett
NIH → QC Ware

Ed Hohenstein
Stanford → QC Ware

Roberto Di Remigio
ENCCS

Ugur Bozkaya
Hacettepe

Peter Kraus
TUB

Susi Lehtola
Helsinki

QCENGINE

ADCC

Maximilian Scheurer
Heidelberg

Michael Herbst
EPFL

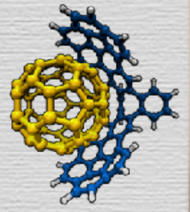
Andreas Dreuw
Heidelberg

CFOUR

John Stanton
UFL

Devin Matthews
SMU

SE / DISP.



Sebastian Ehlert
Bonn

Holger Kruse
Czech Acad. Sci.

Jiří Šponer
Czech Acad. Sci.

GAMESS

Mark Gordon
Iowa State

Nuwan de Silva
W. New England U

ML

Farhad Ramezanghorbani
UFL → Schrödinger

MOLPRO

Sebastian Lee
CalTech

MRCHEM

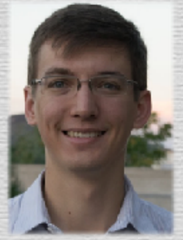
Roberto Di Remigio
Uppsala

NWCHEM

Theresa Windus
Iowa State

Jiyoung Lee
U Texas

Annabelle Lolinco
Iowa State



Logan Ward
Argonne

Adrian Hurtado
Stony Brook

OPENFF / MM

John Chodera
MSKCC

Jeffrey Wagner
UC Irvine

David Dotson
UC Boulder

Joshua Horton
Newcastle

Jamshed Anwar
Lancaster

TERACHEM

Fang Liu
Emory

Heather Kulik
MIT

Colton Hicks
Stanford

Todd Martínez
Stanford

TURBOMOLE

Johannes Steinmetzer
Friedrich Schiller U

HIST./DB/TEST

Nick Petosa
GaTech → Two Sigma

Daniel Nascimento
U Memphis

Dom Sirianni
GaTech

Chaya Stern
MSKCC

ENGINES

Taylor Barnes
MolSSI

OPTIMIZERS

Lee-Ping Wang
UC Davis

Jan Hermann
Free U Berlin

Alexander Heide
UGA

Rollin King
Bethel

PROCEDURES

Asim Alenaizan
GaTech → KFUPM

Peter Kraus
Curtin

Jonathon Misiewicz
Emory

Jeff Schriber
GaTech → Iona

Carlos Borca
Princeton

Philip Nelson
GaTech

QCArchive Acknowledgements

QCArchive

Daniel Smith
MolSSI → Abiologics

Levi Naden
MolSSI

Doaa Altarawy
MolSSI → Alexandria

Matthew Welborn
MolSSI → Entos

Ben Pritchard
MolSSI

MolSSI

Theresa Windus
Iowa State

Daniel Crawford
VaTech

Jessica Nash
MolSSI

Sam Ellis
MolSSI

Susi Lehtola
Helsinki

Open Force Field Consortium

openforcefield.org

psicode.org

molssi.github.io/QCFractal

QCSchema

Daniel Smith
MolSSI → Abiologics

Aaron Virshup
Arzeda

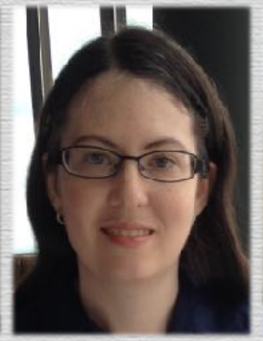
Bert de Jong
LBNL

Geoff Hutchison
U Pittsburgh

Marcus Hanwell
Kitware → Voltron

Eric Berquist
Sandia

Ben Pritchard
MolSSI



Lori Burns
GaTech

Matthew Welborn
MolSSI → Entos

Colton Hicks
Stanford