# QMC Workshop 2021
# Optimization of Pseudopotentials

## Week 6 / 9 November 2021

Gani Annaberdiyev, aannabe@ncsu.edu

North Carolina State University

*https://github.com/QMCPACK/qmc_workshop_2021*

Funding: U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division, as part of the Computational Materials Sciences Program and Center for Predictive Simulation of Functional Materials.

# Conventional ECP Optimization Approaches

For a chosen subset $S$ of atomic spectrum, generate relativistic all-electron (AE) reference states using HF (mean-field). Two main approaches are:

1. Energy-consistency of the atomic spectrum gaps:

$$\mathcal{E}_{\mathrm{HF}}^2 = \sum_{i \in S} \left( \Delta E_{\mathrm{HF}}^{\mathrm{ECP}} - \Delta E_{\mathrm{HF}}^{\mathrm{AE}} \right)_i^2 \tag{1}$$

2. Shape-consistency:

$$\mathcal{N}^2 = \sum_{\ell} \left[ \left( \epsilon_\ell^{\mathrm{ECP}} - \epsilon_\ell^{\mathrm{AE}} \right)^2 + \left( N_\ell^{\mathrm{ECP}} - N_\ell^{\mathrm{AE}} \right)^2 + \left( V_\ell^{\mathrm{ECP}} - V_\ell^{\mathrm{AE}} \right)^2 + \left( S_\ell^{\mathrm{ECP}} - S_\ell^{\mathrm{AE}} \right)^2 \right] \tag{2}$$

where $\epsilon_\ell$: eigenvalues, $N_\ell^{\mathrm{AE}} = \int_0^{R_\ell} \left( r^{\ell+1} \phi_\ell^{\mathrm{AE}}(r) \right)^2 \mathrm{d}r$, $V_\ell^{\mathrm{AE}} = \phi_\ell^{\mathrm{AE}}(R_\ell)$, $S_\ell^{\mathrm{AE}} = \frac{\mathrm{d}}{\mathrm{d}r} \phi_\ell^{\mathrm{AE}}(r) \big|_{R_\ell}$, $R_\ell$: cutoff radius ($r^{\frac{4}{5}} \phi_\ell^{\mathrm{AE}}$).

# Correlation Consistent Effective Core Potentials (ccECP)

A new approach: go beyond mean-field HF, and consider electron-electron correlations.

1. AE reference data: scalar relativistic, fully-correlated (CC, CV, VV) CCSD(T).
2. Many-body method is used:

$$\mathcal{E}^2_{\mathrm{CCSD(T)}} = \sum_{i \in S} \left( \Delta E^{\mathrm{ECP}}_{\mathrm{CCSD(T)}} - \Delta E^{\mathrm{AE}}_{\mathrm{CCSD(T)}} \right)^2_i + \omega \mathcal{N}^2 \tag{3}$$

3. Simple form parametrized with few Gaussians.
4. More emphasis on testing/transferability, especially on compressed XH and XO bond lengths.

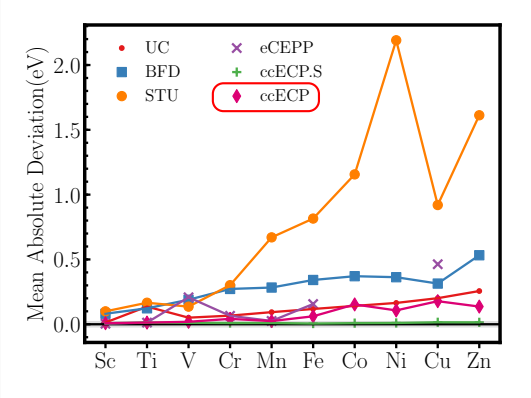
# Deviations between AE and ECP atomic spectrum energies

**AE Reference:** CCSD(T) with all-electrons correlated.

$$\mathrm{MAD} = \frac{1}{N_s} \sum_{s=1}^{N_s} \left| \Delta E_s^{\mathrm{ECP}} - \Delta E_s^{\mathrm{AE}} \right| \qquad (4)$$

BFD: Burkatzki-Filippi-Dolg DHF ECPs for QMC [4, 5]
STU: Stuttgart group DHF ECPs [8]
TN17: Trail-Needs correlated ECPs for QMC [10]
ccECP: New correlation-consistent ECP [2]
UC: is a uncorrelated core CCSD(T)

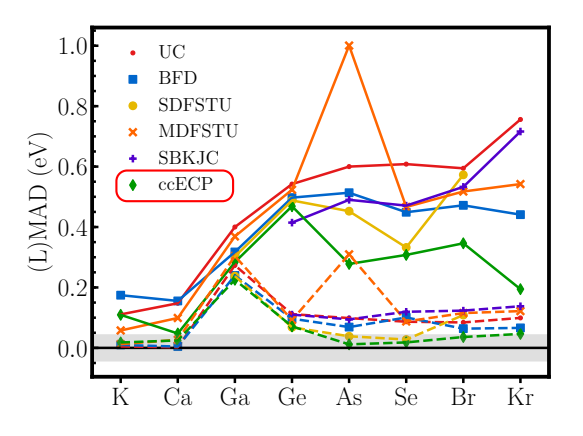
ccECP shows significant improvement over older ECPs.




Annaberdiyev et al, J. Chem. Phys. 149, 134108 (2018) [2]

- Many states are included: neutral excitations, EA, IPs down to the bare core.
- Accurate in low-lying excitations (LMAD) and for large set of states (MAD).
- Better iso-spectrality in all ccECPs: H-Kr.

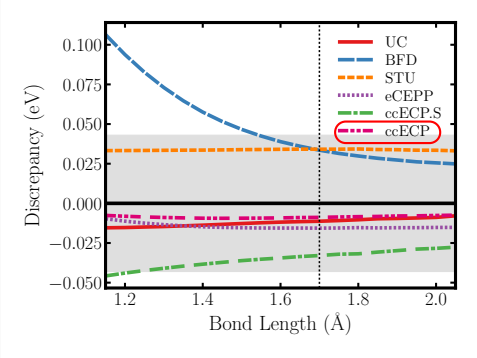
- Training data was optimized better. How does this carry to more complicated systems?




Wang et al, J. Chem. Phys. 151, 144110 (2019) [11]

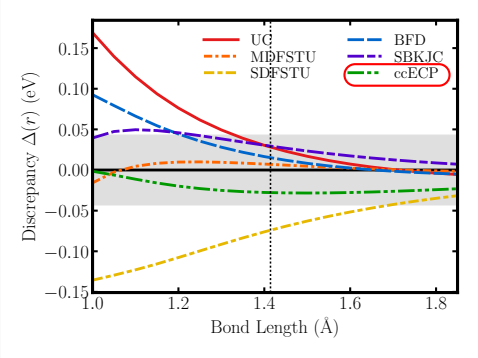
# Discrepancy between AE and ECP monohydride binding energies ($\sigma$ bonds)

VH molecule




Annaberdiyev et al, J. Chem. Phys. 149, 134108 (2018) [2]

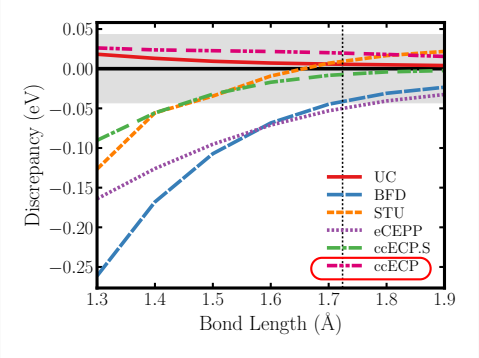
BrH molecule




Wang et al, J. Chem. Phys. 151, 144110 (2019) [11]

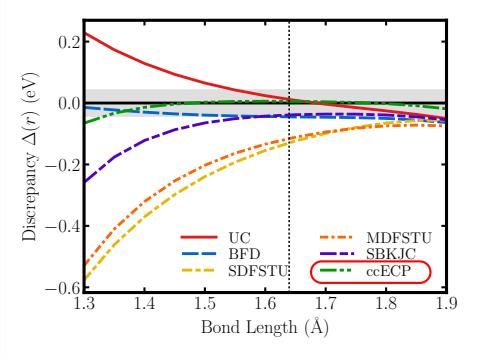
# Discrepancy between AE and ECP monoxide binding energies ($\pi$ bonds)

CuO molecule




Annaberdiyev et al, J. Chem. Phys. 149, 134108 (2018) [2]

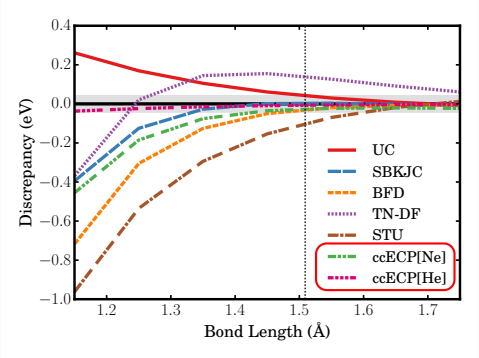
SeO molecule




Wang et al, J. Chem. Phys. 151, 144110 (2019) [11]

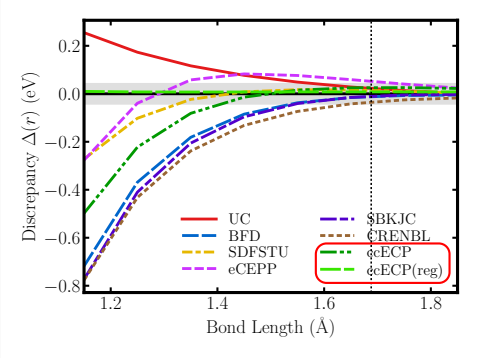
# Various core sizes are available for ccECP

SiO molecule




Bennett et al, J. Chem. Phys. 149, 104108 (2018) [3]

LiO molecule




Wang et al, J. Chem. Phys. 151, 144110 (2019) [11]

DMC singlet/triplet excitations for Si systems agree well with experiments.

**TABLE IV.** $Si_2H_6$ total energies (Ha) and vertical excitation gaps (eV) using various methods. Experimental geometry was used for this molecule.[22]

| | GS ($^1A_{1g}$) | EX ($^1E_{1u}$) | | EX ($^3A_{1g}$) | |
|---|---|---|---|---|---|
| Method | Total (hartree) | Total (hartree) | Gap (eV) | Total (hartree) | Gap (eV) |
| CISD/RHF[a] | −11.3287 | −10.8183 | 13.889 | −11.0825 | 6.699 |
| CISD/CAS(14$e^-$, 13o)[b] | −11.3400 | −11.0623 | 7.557 | | |
| RCCSD(T)/RHF[c] | −11.3766(3) | | | −11.1308(5) | 6.69(2) |
| CCSDT(Q)/RHF[d] | −11.3782(3) | | | −11.1336(5) | 6.66(2) |
| DMC/RHF | −11.3708(2) | −11.0855(2) | 7.763(8) | −11.1215(2) | 6.784(8) |
| DMC/PBE[e] | −11.3725(2) | −11.0934(2) | 7.595(8) | −11.1248(2) | 6.740(8) |
| Lehtonen *et al.*[f] | | | 7.61 | | |
| Experiment[41] | | | 7.6 | | ≈6.7[g] |
| Experiment[39] | | | 7.56 | | |

Wang et al, J. Chem. Phys. 153, 144303 (2020) [12]

# DMC Benchmarks of ccECP

DMC IP and EA of variety of molecular systems were tested by an *independent* groups.

The molecular systems tested:

| System | |
|---|---|
| $C^+$ | $PH^+$ |
| C | PH |
| $C^-$ | $PH^-$ |
| $N^+$ | $PH_2^+$ |
| N | $PH_2$ |
| $O^+$ | $PH_2^-$ |
| O | $S^+$ |
| $O^-$ | S |
| $OH^+$ | $S^-$ |
| OH | $SH^+$ |
| $OH^-$ | SH |
| $O_2^+$ | $SH^-$ |
| $O_2$ | $S_2^+$ |
| $O_2^-$ | $S_2$ |
| $Si^+$ | $S_2^-$ |
| Si | $Cl^+$ |
| $Si^-$ | Cl |
| $P^+$ | $Cl^-$ |
| P | $Cl_2^+$ |
| $P^-$ | $Cl_2$ |
| | $Cl_2^-$ |

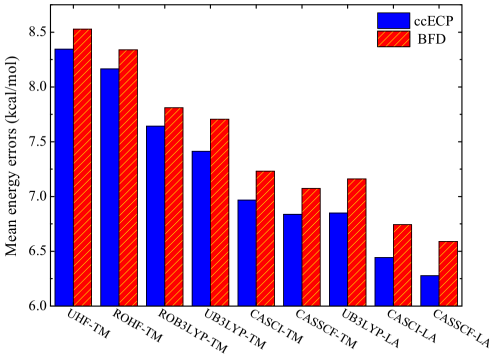
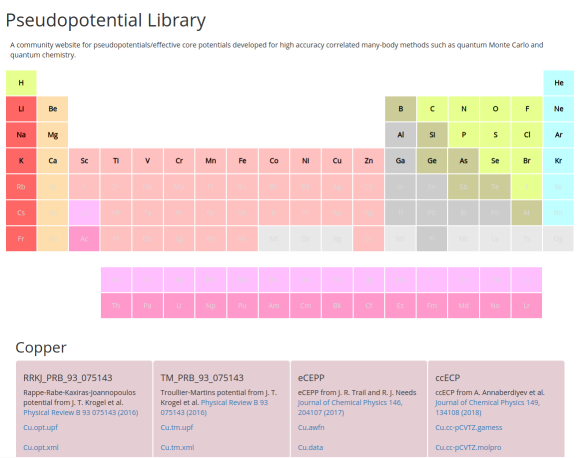



FIG. 5. Mean errors in DMC energies using ccECP and BFD potentials with different trial wave functions and nonlocal-treating strategies.

X. Zhou et al, Phys. Rev. A 100, 062502, 2019 [14]

# ECP Library: `https://pseudopotentiallibrary.org`

- The library is hosted on GitHub: *`https://github.com/QMCPACK/pseudopotentiallibrary`*
- AREP ccECPs for H-Kr elements are available now.
- (aug)-cc-p(C)V$n$Z [$n = D - 6$] basis-sets are available.
- Commonly used formats are available: MOLPRO, GAMESS, NWCHEM, PYSCF.
- Also, UPF files are available for use in plane wave basis codes.
- Heavier elements with SOC will be added later, attend Week-8 to learn how to run with SOREP.



# An example using ccECP: Sc atom via NEXUS

See Week 3 and Week 4 material for introduction to NEXUS.

Update workshop example files:

```
cd home/apps/qmcpack
git pull

cd home/qmcuser/qmc_workshop_2021
git pull
```

Enter week 6 example directory:

```
cd week6_ecps_and_observables/02_ccECP_slides_runs/Sc_atom/workstation/
```

```
ls
geom.xyz  pseudo  runs  scf_template.py  workflow.py
```

```
ls runs/
c4q  gs_noj  optJ12  optJ123  qmc_no  qmc_v0  qmc_v1  scf
```

```
ls pseudo/
download.sh  Sc.ccECP.xml
```

```
vim pseudo/download.sh
```

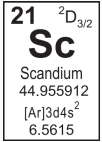
```
#!/usr/bin/env bash

atom1="Sc"

wget https://pseudopotentiallibrary.org/recipes/${atom1}/ccECP/${atom1}.ccECP.xml
```

```
vim geom.xyz
```

```
1

Sc   0.0000  0.0000  0.0000
```

*vim scf_template.py*

```python
#!/usr/bin/env python3

import sys, os
from pyscf import scf, gto
import numpy as np
from urllib.request import urlretrieve

### Set the current working directory
cwd = os.getcwd()
pplib = "http://pseudopotentiallibrary.org/recipes"
pptype = "ccECP"

### Retrieve basis and ECP files from pseudopotentiallibrary.org
atom1 = "Sc"
bastype1 = "cc-pVTZ"
basfile1 = "{0}.{1}.nwchem".format(atom1,bastype1)
ecpfile1 = "{0}.{1}.nwchem".format(atom1,pptype)
xmlfile1 = "{0}.{1}.xml".format(atom1,pptype)
urlretrieve("{0}/{1}/{2}/{3}".format(pplib,atom1,pptype,basfile1), filename=basfile1)
urlretrieve("{0}/{1}/{2}/{3}".format(pplib,atom1,pptype,ecpfile1), filename=ecpfile1)
urlretrieve("{0}/{1}/{2}/{3}".format(pplib,atom1,pptype,xmlfile1), filename=xmlfile1)

###~~~~ Build the molecule ~~~~

# Nexus expands this with Mole info
$system

with open(os.path.join(cwd,basfile1)) as f:
    bas1 = f.read()
with open(os.path.join(cwd,ecpfile1)) as f:
    ecp1 = f.read()
mol.basis = {atom1: gto.basis.parse(bas1)}
mol.ecp = {atom1: gto.basis.parse_ecp(ecp1)}
mol.build()
```




```python
#~~~Run HF on molecule~~~~
mf = scf.ROHF(mol)
mf.irrep_nelec = {
'Ag' : (3,2),   # s, dz^2, dx^2-y^2
'B3u': (1,1),   # x    1
'B1u': (1,1),   # z    0
'B2u': (1,1),   # y   -1
'B2g': (0,0),   # xz   1
'B3g': (0,0),   # yz  -1
'B1g': (0,0),   # xy  -2
'Au' : (0,0)    # xyz
}
mf.max_cycle=100
mf.chkfile='scf.chkfile'
en=mf.kernel()
mf.analyze()
```

*vim workflow.py*

```python
#!/usr/bin/env python3

from nexus import settings,job,run_project,obj
from nexus import generate_physical_system
from nexus import generate_pyscf
from nexus import generate_convert4qmc
from nexus import generate_qmcpack,loop,linear,vmc,dmc

# Obtain the core count of the local machine (lab only)
import os
cores = os.cpu_count()

settings(
    pseudo_dir    = 'pseudo',
    results       = '.',
    runs          = 'runs',
    status_only   = 0,
    generate_only = 0,
    sleep         = 10,          # In seconds
    machine     = 'ws'+str(cores),
    )

system = generate_physical_system(
    structure = 'geom.xyz',
    Sc         = 11,    # 11 valence electrons
    net_spin   = 1,     # 2S
    net_charge = 0,
    )

sims = []

## Perform SCF
scf = generate_pyscf(
    identifier = 'scf', # Log output goes to scf.out
    path       = 'scf', # Directory to run in
    job        = job(serial=True,app='python3'),
    system     = system,
    template   = 'scf_template.py',      # pyscf template file
    mole       = obj(                    # used for the Mole() object
        verbose  = 4,
        symmetry = 'D2h',
        ),
    save_qmc   = True,                   # save wfn data for qmcpack
    )
sims.append(scf)

###### convert orbitals to QMCPACK format
c4q = generate_convert4qmc(
    identifier   = 'c4q',
    path         = 'c4q',
    job          = job(cores=1),
    no_jastrow   = True,
    dependencies = (scf,'orbitals'),
    )
sims.append(c4q)

### collect dependencies relating to orbitals
orbdeps = [(c4q,'particles'), # pyscf changes particle positions
           (c4q,'orbitals' ),]
```

```
python workflow.py
...

ls runs/scf/
Sc.ccECP.nwchem  Sc.ccECP.xml  Sc.cc-pVTZ.nwchem  scf.chkfile  scf.err
scf.h5  scf.out  scf.py  scf.sbatch.in  scf.struct.xyz  sim_scf


vim runs/scf/scf.py
```

```python

###~~~~ Build the molecule ~~~~

# Nexus expands this with Mole info

### generated system text ###
from pyscf import gto as gto_loc
mol = gto_loc.Mole()
mol.verbose  = 4
mol.atom     = '''
               Sc   0.00000000   0.00000000   0.00000000
               '''
mol.unit     = 'A'
mol.charge   = 0
mol.spin     = 1
mol.symmetry = 'D2h'
mol.build()
### end generated system text ###

```

```python

### generated conversion text ###
from PyscfToQmcpack import savetoqmcpack
savetoqmcpack(mol,mf,'scf')
### end generated conversion text ###

```

*vim runs/scf/scf.out*

```
cycle= 15 E= -46.121844325316  delta_E= 2.48e-06  |g|= 0.0001  |ddm|= 0.0232
HOMO (Ag) = -0.0512860808024703  LUMO (B1u) = 0.0212213370518054
cycle= 16 E= -46.1218430105355  delta_E= 1.31e-06  |g|= 0.000361  |ddm|= 0.0316
HOMO (Ag) = -0.0511165418814432  LUMO (B1u) = 0.0212239988441713
cycle= 17 E= -46.1218430689892  delta_E= -5.85e-08  |g|= 0.000251  |ddm|= 0.00115
HOMO (Ag) = -0.0511178095270498  LUMO (B1u) = 0.0212239829283067
cycle= 18 E= -46.1218430876165  delta_E= -1.86e-08  |g|= 0.000204  |ddm|= 0.00084
HOMO (Ag) = -0.0509412957047003  LUMO (B1u) = 0.0212268278098667
cycle= 19 E= -46.1218431223565  delta_E= -3.47e-08  |g|= 4.31e-05  |ddm|= 0.00101
HOMO (Ag) = -0.0508978866963989  LUMO (B1u) = 0.0212266815922 3
cycle= 20 E= -46.1218431246405  delta_E= -2.28e-09  |g|= 3.64e-06  |ddm|= 0.00103
HOMO (Ag) = -0.0508900896655348  LUMO (B1u) = 0.0212268078866954
cycle= 21 E= -46.1218431242121  delta_E= 4.28e-10  |g|= 5.44e-06  |ddm|= 0.00042
HOMO (Ag) = -0.0508976447814269  LUMO (B1u) = 0.0212266453759735
Extra cycle  E= -46.1218431242235  delta_E= -1.14e-11  |g|= 4.05e-06  |ddm|= 1.69e-05
converged SCF energy = -46.1218431242235
**** SCF Summaries ****
Total Energy =                          -46.121843124223474
Nuclear Repulsion Energy =                0.000000000000000
One-electron Energy =                   -76.798902125225936
Two-electron Energy =                    30.677059001002462
Wave-function symmetry = Ag
occupancy for each irrep:     Ag  B1g  B2g  B3g   Au  B1u  B2u  B3u
double occ                     2    0    0    0    0    1    1    1
single occ                     1    0    0    0    0    0    0    0
**** MO energy ****
                           Roothaan              | alpha                | beta
MO #1   (Ag  #1 ) energy= -2.59527820108142  | -2.62444340458707  | -2.56611299757682  occ= 2
MO #2   (B1u #1 ) energy= -1.58555599442954  | -1.59253948744459  | -1.57857250141318  occ= 2
MO #3   (B3u #1 ) energy= -1.57875675542816  | -1.63032348524351  | -1.52718908656128  occ= 2
MO #4   (B2u #1 ) energy= -1.57875438901967  | -1.63033557807     | -1.52717319996934  occ= 2
MO #5   (Ag  #2 ) energy= -0.210835058082764 | -0.215125004312978 | -0.206545111853038 occ= 2
MO #6   (Ag  #3 ) energy= -0.0508976447814269 | -0.346452079424369 | 0.2446589667793811  occ= 1
MO #7   (B1u #2 ) energy= 0.0212266453759735 | 0.0209452515858588  | 0.0215080391660666 occ= 0
MO #8   (B3u #2 ) energy= 0.024412614689053  | 0.0228037087932995 | 0.0260215205848067 occ= 0
MO #9   (B2u #2 ) energy= 0.0244135828037429 | 0.0228041642565865 | 0.0260230013508989 occ= 0
MO #10  (Ag  #4 ) energy= 0.0638801098953974 | 0.0633192197670475 | 0.0644410000219459 occ= 0
MO #11  (B2g #1 ) energy= 0.0642676989705254 | 0.0638575172384601 | 0.0646778807025906 occ= 0
```

```
MO #61  (B3u #9 ) energy= 2.38408239635703    | 2.35
MO #62  (B2u #9 ) energy= 2.38408249146256    | 2.35
MO #63  (Ag  #16) energy= 5.42588680466658    | 5.42
 ** Mulliken pop on meta-lowdin orthogonal AOs  **
 ** Mulliken pop  **
pop of  0 Sc 3s        1.99988
pop of  0 Sc 4s        1.99833
pop of  0 Sc 5s        0.00000
pop of  0 Sc 6s        0.00000
pop of  0 Sc 7s        0.00000
pop of  0 Sc 3px       1.99781
pop of  0 Sc 3py       1.99781
pop of  0 Sc 3pz       1.99833
pop of  0 Sc 4px       0.00083
pop of  0 Sc 4py       0.00083
pop of  0 Sc 4pz       0.00063
pop of  0 Sc 5px       0.00114
pop of  0 Sc 5py       0.00114
pop of  0 Sc 5pz       0.00091
pop of  0 Sc 6px       0.00006
pop of  0 Sc 6py       0.00006
pop of  0 Sc 6pz       0.00007
pop of  0 Sc 7px       0.00003
pop of  0 Sc 7py       0.00003
pop of  0 Sc 7pz       0.00002
pop of  0 Sc 3dxy      0.00000
pop of  0 Sc 3dyz      0.00000
pop of  0 Sc 3dz^2     0.00145
pop of  0 Sc 3dxz      0.00000
pop of  0 Sc 3dx2-y2   0.92126
pop of  0 Sc 4dxy      0.00000
pop of  0 Sc 4dyz      0.00000
pop of  0 Sc 4dz^2     0.00029
pop of  0 Sc 4dxz      0.00000
pop of  0 Sc 4dx2-y2   0.06730
pop of  0 Sc 5dxy      0.00000
pop of  0 Sc 5dyz      0.00000
pop of  0 Sc 5dz^2     0.00004
pop of  0 Sc 5dxz      0.00000
```

*vim workflow.py*

```python

### Run VMC without Jastrow for sanity check
gs_noj = generate_qmcpack(
    identifier   = 'gs_noj',
    path         = 'gs_noj',
    job          = job(cores=cores),
    system       = system,
    pseudos        = ['Sc.ccECP.xml'],
    jastrows     = [],
    calculations   = [
        vmc(
            walkers     =   1,
            warmupsteps =  20,
            blocks      =   5,
            steps       =  50,
            substeps    =   2,
            timestep    = 0.7,
            ),
        ],
    dependencies = orbdeps,
    )
sims.append(gs_noj)

```

*cd runs/gs_noj/*

*qmca -q ekpvar -e 3 --sac *.scalar.**

```
gs_noj  series 0
  LocalEnergy           =         -46.12177 +/-          0.00072    1.3
  Variance              =            4.493 +/-            0.024    1.3
  Kinetic               =           18.7966 +/-           0.0027    1.0
  LocalPotential        =          -64.9184 +/-           0.0028    1.0
  AcceptRatio           =          0.498003 +/-         0.000029    1.0
```

SCF energy = -46.121843 Ha

*vim workflow.py*

```python
##### OPTIMIZATION methods
linopt1 = linear(
    energy               = 0.0,
    unreweightedvariance = 1.0,
    reweightedvariance   = 0.0,
    samples              = int(3e3),
    substeps             = 2,
    steps                = 20,
    blocks               = 20,
    nonlocalpp           = True,
    usedrift             = True,
    minmethod            = 'OneShiftOnly',
    minwalkers           = 1e-3,
    timestep             = 1.0,
    )
linopt2 = linopt1.copy()
linopt2.minwalkers = 0.10
linopt2.samples = linopt1.samples*2

linopt3 = linopt2.copy()
linopt3.unreweightedvariance = 0.0
linopt3.reweightedvariance = 0.10
linopt3.energy = 0.90
linopt3.minwalkers = 0.30
linopt3.samples = linopt2.samples*2

####### optimize 1, 2-body Jastrow
optJ12 = generate_qmcpack(
    identifier     = 'optJ12',
    path           = 'optJ12',
    job            = job(cores=cores),
    system         = system,
    J2             = True,          # 2-body B-spline Jastrow
    J1_rcut        = 12.0,          # Bohr cutoff for J1
    J2_rcut        = 12.0,          # Bohr cutoff for J2
    pseudos        = ['Sc.ccECP.xml'],
    calculations   = [
                loop(max=5, qmc=linopt1),
                loop(max=5, qmc=linopt2),
                loop(max=5, qmc=linopt3),
        ],
    dependencies = orbdeps,
    )
sims.append(optJ12)
```

*cd runs/optJ12*

*qmca -q ev -e 1 --sac *.scalar.**

| | | LocalEnergy | | Variance | | ratio | |
|---|---|---|---|---|---|---|---|
| optJ12 | series 0 | -46.138778 +/- 0.002003 | 1.0 | 3.076381 +/- 0.006039 | 1.0 | 0.0667 | |
| optJ12 | series 1 | -46.419171 +/- 0.001346 | 2.2 | 0.606835 +/- 0.002252 | 1.0 | 0.0131 | |
| optJ12 | series 2 | -46.452039 +/- 0.000896 | 1.0 | 0.625210 +/- 0.003446 | 1.0 | 0.0135 | |
| optJ12 | series 3 | -46.458866 +/- 0.000884 | 1.0 | 0.654501 +/- 0.005287 | 1.0 | 0.0141 | |
| optJ12 | series 4 | -46.461309 +/- 0.001270 | 1.0 | 0.663231 +/- 0.005007 | 1.2 | 0.0143 | |
| optJ12 | series 5 | -46.464978 +/- 0.000693 | 1.0 | 0.665643 +/- 0.003202 | 1.0 | 0.0143 | |
| optJ12 | series 6 | -46.466492 +/- 0.000964 | 1.0 | 0.670662 +/- 0.004712 | 1.0 | 0.0144 | |
| optJ12 | series 7 | -46.467072 +/- 0.000584 | 1.0 | 0.669408 +/- 0.003340 | 1.0 | 0.0144 | |
| optJ12 | series 8 | -46.469110 +/- 0.000763 | 1.0 | 0.686031 +/- 0.013676 | 1.0 | 0.0148 | |
| optJ12 | series 9 | -46.469448 +/- 0.000838 | 2.5 | 0.680276 +/- 0.005176 | 1.0 | 0.0146 | |
| optJ12 | series 10 | -46.468701 +/- 0.000573 | 1.0 | 0.675707 +/- 0.002290 | 1.4 | 0.0145 | |
| optJ12 | series 11 | -46.469053 +/- 0.000484 | 1.0 | 0.678638 +/- 0.002415 | 1.0 | 0.0146 | |
| optJ12 | series 12 | -46.469395 +/- 0.000546 | 1.7 | 0.676652 +/- 0.001945 | 1.0 | 0.0146 | |
| optJ12 | series 13 | -46.469334 +/- 0.000357 | 1.0 | 0.680689 +/- 0.003857 | 1.0 | 0.0146 | |
| optJ12 | series 14 | -46.470180 +/- 0.000619 | 1.0 | 0.683609 +/- 0.007195 | 4.6 | 0.0147 | |

*vim workflow.py*

```
128
129 ###### optimize 3-body Jastrow
130 optJ123 = generate_qmcpack(
131     identifier     = 'optJ123',
132     path           = 'optJ123',
133     job            = job(cores=cores),
134     system         = system,
135     J3             = True,
136     J3_rcut        = 12.0,
137     pseudos        = ['Sc.ccECP.xml'],
138     calculations   = [
139                 loop(max=10, qmc=linopt3),
140         ],
141     dependencies   = orbdeps+[(optJ12,'jastrow')],
142     )
143 sims.append(optJ123)
144
```

*cd runs/optJ123*

*qmca -q ev -e 1 --sac *.scalar.**

```
                                LocalEnergy               Variance           ratio
optJ123  series 0  -46.467934 +/- 0.000419    1.0   0.672600 +/- 0.002882    1.0   0.0145
optJ123  series 1  -46.489434 +/- 0.000408    1.0   0.578860 +/- 0.008589    1.0   0.0125
optJ123  series 2  -46.490477 +/- 0.000430    1.0   0.656540 +/- 0.029653    1.0   0.0141
optJ123  series 3  -46.491878 +/- 0.000578    1.1   0.656817 +/- 0.006533    1.0   0.0141
optJ123  series 4  -46.492373 +/- 0.000553    1.0   0.681497 +/- 0.012269    1.0   0.0147
optJ123  series 5  -46.492559 +/- 0.000996    3.2   0.673054 +/- 0.008965    1.3   0.0145
optJ123  series 6  -46.492327 +/- 0.000518    1.0   0.666388 +/- 0.008933    1.0   0.0143
optJ123  series 7  -46.493250 +/- 0.000335    1.0   0.683002 +/- 0.008520    1.0   0.0147
optJ123  series 8  -46.493760 +/- 0.000440    1.0   0.682064 +/- 0.015371    1.0   0.0147
optJ123  series 9  -46.493525 +/- 0.000588    1.0   0.680150 +/- 0.005577    1.1   0.0146
```

*vim workflow.py*

```python
t_moves = ['no', 'v0', 'v1']
locality_run = {}

#### run DMC with QMCPACK
MY_TARGET_WALKERS = 16   # Small number for presentation purposes. Needs to be at least a few thousand walkers.

for moves in t_moves:
    name = 'qmc_' + moves

    locality_run[moves] = generate_qmcpack(
        identifier   = name,
        path         = name,
        job          = job(cores=cores),
        system       = system,
        pseudos        = ['Sc.ccECP.xml'],
        jastrows     = [],
        calculations    = [
            vmc(
                #walkers     =    1,
                walkers     = int(MY_TARGET_WALKERS/cores),   # Per MPI
                warmupsteps =  20,
                blocks      =   5,
                steps       =  50,
                substeps    =   2,
                timestep    = 0.5,
                ),
            dmc(targetwalkers = MY_TARGET_WALKERS,  # Total walkers
                timestep = 0.02,
                warmupsteps =  int(5/0.02),
                blocks = 5,
                steps = int(1.0/0.02),
                nonlocalmoves = moves,
                checkpoint = 5
                ),
```

```python
            dmc(targetwalkers = MY_TARGET_WALKERS,  # Total walkers
                timestep = 0.01,
                warmupsteps =  int(2/0.01),
                blocks = 5,
                steps = int(1.0/0.01),
                nonlocalmoves = moves,
                checkpoint = 5
                ),
            dmc(targetwalkers = MY_TARGET_WALKERS,  # Total walkers
                timestep = 0.005,
                warmupsteps =  int(1/0.005),
                blocks = 5,
                steps = int(1.0/0.005),
                nonlocalmoves = moves,
                checkpoint = 5
                ),
            dmc(targetwalkers = MY_TARGET_WALKERS,  # Total walkers
                timestep = 0.0025,
                warmupsteps =  int(1/0.0025),
                blocks = 5,
                steps = int(1.0/0.0025),
                nonlocalmoves = moves,
                checkpoint = 5
                ),
            ],
        dependencies = orbdeps+[(optJ123,'jastrow')],
        )
    sims.append(locality_run[moves])

run_project(sims)
```

For more info see:

*https://qmcpack.readthedocs.io/en/develop/methods.html#dmc-driver*

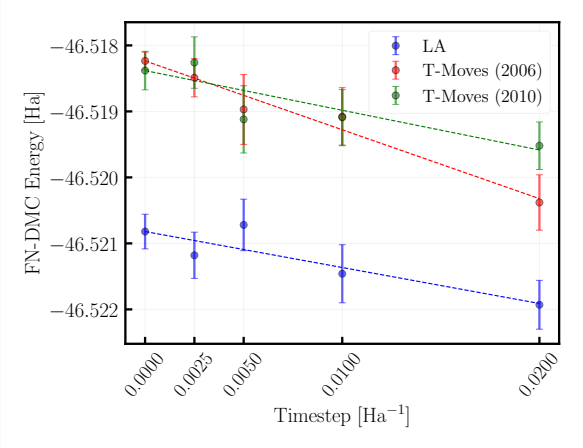
```
ls runs/
c4q  gs_noj  optJ12  optJ123
qmc_no  qmc_v0  qmc_v1  scf
```

```
cd runs/qmc_v1/
```

```
qmca -q ev -e 3 *scalar*
```

```
                          LocalEnergy               Variance           ratio
qmc_v1  series 0  -46.488935 +/- 0.000314   0.575068 +/- 0.003038   0.0124
qmc_v1  series 1  -46.519518 +/- 0.000364   0.593276 +/- 0.003472   0.0128
qmc_v1  series 2  -46.519089 +/- 0.000422   0.584251 +/- 0.002304   0.0126
qmc_v1  series 3  -46.519124 +/- 0.000513   0.578059 +/- 0.001688   0.0124
qmc_v1  series 4  -46.518257 +/- 0.000394   0.574395 +/- 0.001310   0.0123
```

# Sc atom results




| Locality Treatment | Execution Time (min) |
|---|---|
| LA [9] | 18.3 |
| T-Moves (2006) [6] | 21.0 |
| T-Moves (2010) [7] | 26.6 |

**Table 1:** Comparison of execution times for various locality treatments. NERSC, 8 KNL nodes.

# Checking the results against reference data

**Table 17. Most Accurate Total Energies for ccECP[Ne] K–Zn Elements along with Fixed-Node DMC Energies with Single-Reference HF Trial Wave Functions**[a]

| atom | state | "exact" (Ha) | DMC/HF (Ha) | $\epsilon$ (mHa) | $\eta$ |
|---|---|---|---|---|---|
| K | ($^2$S) | −28.25243(25) | −28.2394(2) | 13.0(3) | 4.1(1) |
| Ca | ($^1$S) | −36.72897(46) | −36.7055(2) | 23.5(5) | 6.2(1) |
| Sc | ($^2$D) | −46.55704(81) | −46.5202(4) | 36.8(9) | 8.4(2) |
| Ti | ($^3$F) | −58.09263(76) | −58.0458(2) | 46.8(8) | 9.6(2) |
| V | ($^4$F) | −71.44178(59) | −71.3829(2) | 58.9(6) | 10.8(1) |
| Cr | ($^7$S) | −86.64109(33) | −86.5876(2) | 53.5(4) | 9.03(7) |
| Mn | ($^6$S) | −103.8919(10) | −103.8260(3) | 66(1) | 10.2(2) |
| Fe | ($^5$D) | −123.38804(93) | −123.3100(3) | 78(1) | 10.5(1) |
| Co | ($^4$F) | −145.1541(10) | −145.0709(3) | 83(1) | 10.1(1) |
| Ni | ($^3$F) | −169.3912(12) | −169.2973(6) | 94(1) | 10.3(1) |
| Ni | ($^3$D) | −169.3932(12) | −169.3056(6) | 88(1) | 9.0(1) |
| Cu | ($^2$S) | −196.4038(10) | −196.3178(3) | 86(1) | 8.1(1) |
| Zn | ($^1$S) | −226.3699(18) | −226.2775(4) | 92(2) | 8.4(2) |

Annaberdiyev et al, J. Chem. Theory Comput. 2020, 16, 3, 1482–1502 [1]

# Another reason to use T-Moves: Avoiding instabilities

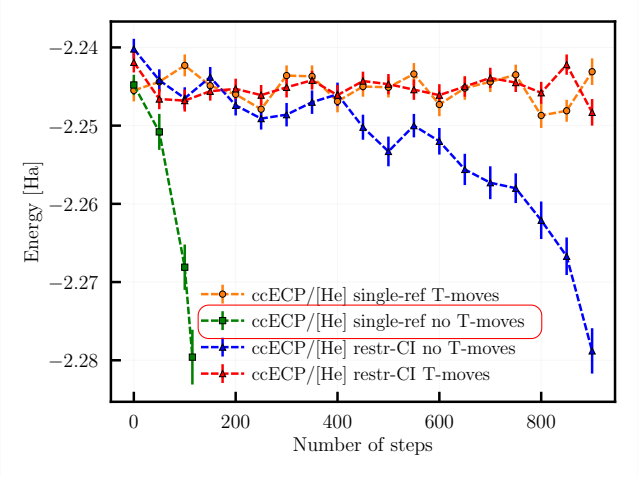



**Figure:** FNDMC energies vs number of steps (in time steps of 0.001 $Ha^{-1}$) for $BeH_2$ molecule. Be atom is represented by ccECP/[He].

H. Zhou et al, arXiv:2109.08653, [13]

Key takeaway message: The quality of pseudopotentials require thorough testing for use in production calculations.

So far, we have only focused on ECPs with *averaged* spin-orbit effects. Calculations with explicit spin-orbit terms will be covered in Week 8.

```
home/qmcuser/qmc_workshop_2021/week8_spin-orbit_qmc/
```

# References i


A. Annaberdiyev, C. A. Melton, M. C. Bennett, G. Wang, and L. Mitas.
**Accurate Atomic Correlation and Total Energies for Correlation Consistent Effective Core Potentials.**
*Journal of Chemical Theory and Computation*, 16(3):1482–1502, Mar. 2020.

A. Annaberdiyev, G. Wang, C. A. Melton, M. C. Bennett, L. Shulenburger, and L. Mitas.
**A new generation of effective core potentials from correlated calculations: 3d transition metal series.**
*The Journal of Chemical Physics*, 149(13):134108, Oct. 2018.

M. C. Bennett, G. Wang, A. Annaberdiyev, C. A. Melton, L. Shulenburger, and L. Mitas.
**A new generation of effective core potentials from correlated calculations: 2nd row elements.**
*The Journal of Chemical Physics*, 149(10):104108, Sept. 2018.

M. Burkatzki, C. Filippi, and M. Dolg.
**Energy-consistent pseudopotentials for quantum Monte Carlo calculations.**
*The Journal of Chemical Physics*, 126(23):234105, June 2007.

# References ii


M. Burkatzki, C. Filippi, and M. Dolg.
**Energy-consistent small-core pseudopotentials for 3d-transition metals adapted to quantum Monte Carlo calculations.**
*The Journal of Chemical Physics*, 129(16):164115, Oct. 2008.

M. Casula.
**Beyond the locality approximation in the standard diffusion Monte Carlo method.**
*Physical Review B*, 74(16):161102, Oct. 2006.

M. Casula, S. Moroni, S. Sorella, and C. Filippi.
**Size-consistent variational approaches to nonlocal pseudopotentials: Standard and lattice regularized diffusion Monte Carlo methods revisited.**
*The Journal of Chemical Physics*, 132(15):154113, Apr. 2010.

M. Dolg, U. Wedig, H. Stoll, and H. Preuss.
**Energy-adjusted ab initio pseudopotentials for the first row transition elements.**
*The Journal of Chemical Physics*, 86(2):866–872, Jan. 1987.

# References iii


L. Mitáš, E. L. Shirley, and D. M. Ceperley.
**Nonlocal pseudopotentials and diffusion Monte Carlo.**
*The Journal of Chemical Physics*, 95(5):3467–3475, Sept. 1991.

J. R. Trail and R. J. Needs.
**Shape and energy consistent pseudopotentials for correlated electron systems.**
*The Journal of Chemical Physics*, 146(20):204107, May 2017.

G. Wang, A. Annaberdiyev, C. A. Melton, M. C. Bennett, L. Shulenburger, and L. Mitas.
**A new generation of effective core potentials from correlated calculations: 4s and 4p main group elements and first row additions.**
*The Journal of Chemical Physics*, 151(14):144110, Oct. 2019.

G. Wang, A. Annaberdiyev, and L. Mitas.
**Binding and excitations in SixHy molecular systems using quantum Monte Carlo.**
*The Journal of Chemical Physics*, 153(14):144303, Oct. 2020.

# References iv


H. Zhou, A. Scemama, G. Wang, A. Annaberdiyev, B. Kincaid, M. Caffarel, and L. Mitas.
**A quantum Monte Carlo study of systems with effective core potentials and node nonlinearities.**
*arXiv:2109.08653 [physics]*, Sept. 2021.

X. Zhou, H. Zhao, T. Wang, and F. Wang.
**Diffusion quantum Monte Carlo calculations with a recent generation of effective core potentials for ionization potentials and electron affinities.**
*Physical Review A*, 100(6):062502, Dec. 2019.