pyOptking

a robust, flexible optimizer for Psi4

Dr. Rollin A. King
Bethel University
St. Paul, Minnesota

BETHEL
UNIVERSITY

Department of
Chemistry

Contributions by
Andreas Copan
John Cayton
Alex Heide

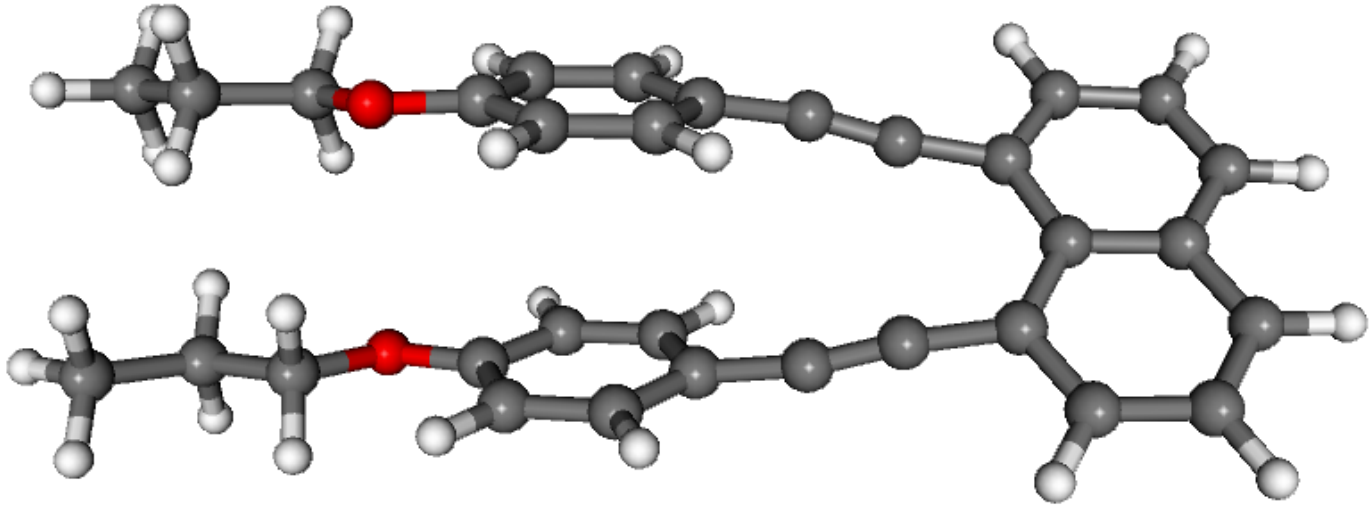
Objectives of pyOptking

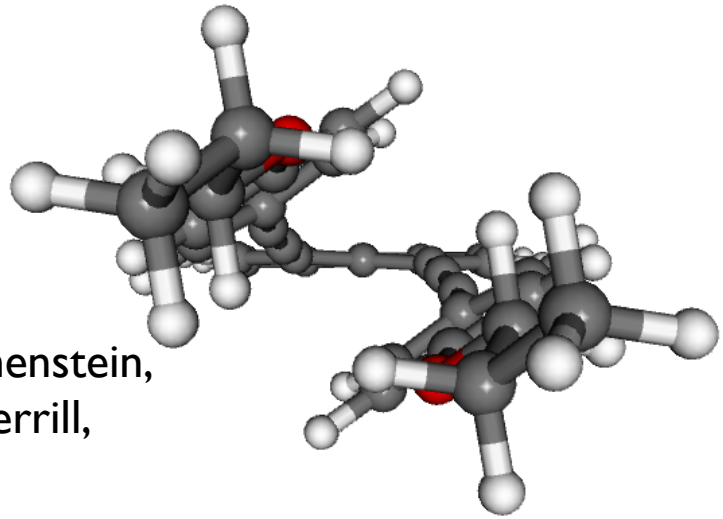
- improve power and flexibility by 'inverting' the optimization process
  - old way: successive calls to optimizer(gradient-value)
  - new way: one call to optimizer(gradient-function)
- produce an optimizer readily compatible with other atom-based optimization programs

Desired characteristics

- Efficiency
- Robustness
- Predictability

auxiliary bonds
steepest-descent

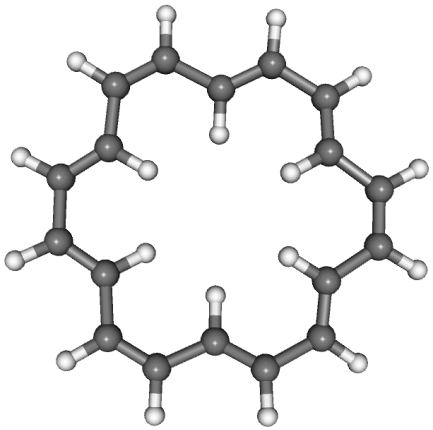




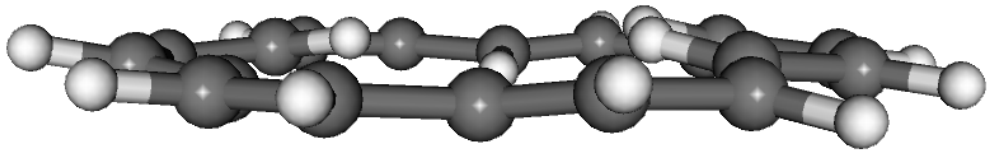
Collaborators: B.E. Carson, T.M. Parker, E.G. Hohenstein, G.L. Brizius, W. Komorner, D.M. Collard, C.D. Sherrill, Georgia Institute of Technology.

Chemistry – A European Journal, **21**, 19168 (2015).

18-annulene revisited



What is the symmetry of this (4n+2) π-electron system? D_{6h} , D_{3h} , C_i , C_2 ,…?



Resorted to steepest-descent and *manual line-searching* for C_2 CCSD(T)!

Nemirowski and P. Schreiner, Justus-Liebig University, Giessen.
T. D. Crawford, Virginia Tech.

Code Status

- virtually all capabilities of C++ code
  - RFO, NR, P-RFO (TS)
  - internals (frozen and fixed), cartesians
  - Hessian guesses, updates, transformations
  - flexible convergence criteria
- current tasks
  - debugging fixed cartesians; IRC; "run-levels"
- new capabilities
  - automated line-searching
  - return trajectory

Code snippets

```
# class method to make an optking system from a psi4 molecule
mol = core.get_active_molecule()
OptMol = optking.molsys.MOLSYS.fromPsi4Molecule(mol)

# Takes and returns a numpy array
def setGeometry_func( newGeom ):
    psi_geom = core.Matrix.from_array( newGeom )
    mol.set_geometry( psi_geom )
    mol.update_geometry()
    return np.array( mol.geometry() )
```

Code snippets

```
calcName = 'ccsd'

# Returns energy and gradient.  In- and out- formats are numpy.
def gradient_func(xyz):
    xyz[:] = setGeometry_func(xyz)
    psi4gradientMatrix, wfn = driver.gradient(calcName, molecule=mol, return_wfn=True)
    gradientMatrix = np.array( psi4gradientMatrix )
    E = wfn.energy()
    return E, np.reshape(gradientMatrix, (gradientMatrix.size))

optking.optimize( OptMol, optking_user_options, setGeometry_func,
gradient_func, hessian_func, energy_func)
```

Discussion Points

- Keep the current C++ optking?
- Should Psi4 check keywords?
- Easiest tool to visualize trajectories (minimization or IRC)?
- Desired future features?

Other projects

- Raman optical activity
  - Alex Heide and Kate Rynders
- absorption spectra of photovoltaic donor polymer candidates
  - Mitchell Lahm
- optical activity of solvated species
  - Sarah Greteman-Leo, Kendra Folsom
- partitioning of contributions to optical activity
  - Sarah Elliott