Hands-on tutorial

Python basics

Toon Verstraelen

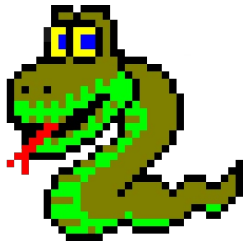
Center for Molecular Modeling (CMM), Ghent University, Belgium

ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

Compiled / JIT / Interpreted

Interpreted languages	**Just-in-time compilation**	**Compiled languages**
• Bash	Julia	• C
• Python	(Python)	• C++
• Perl		• Fortran
• Matlab / Octave		• Cython
• Basic		• Basic
• TCL		• (Java)
• ...		• ...
No compilation	**X**	**Compilation**
Faster development	**X**	**Slower development**
Slower execution	**X**	**Faster execution**
(More features)	**X**	**(Less features)**

Brief history of Python



1989 -- 0.x

- Main developer: Guido Van Rossum (BDFL)

1994 -- 1.x

- Influences from lisp, ABC, C++, ...
- Small community

2000 -- 2.x

- Mature language, easier to use
- Widespread adoption
- Latest in this series 2.7.x
- End-of-life: Jan 2020

2008 -- 3.x

- Remove redundancies: less features
- Fix design mistakes
- Not backward compatible

Why Python?

Principal advantages

- Easy to learn (syntax)
- Extensive language features
- Multiparadigm
- Rapid development
- *Batteries included*
- Cross-platform

Main disadvantage

- Performance. (JIT added as afterthought)

Why programming?

Why programming?

Time for hands-on

- How to use Python: notebook, interpreter, file
- Python as a calculator
- Variables, lists, dictionaries, sets
- Basic string handling
- Flow control
- One-liners
- Functions: `char2l`, `madelung`
- Classes
- Modules
- Unit testing

Hands-on tutorial

Numpy, Scipy, Matplotlib & Autograd

Toon Verstraelen

Center for Molecular Modeling (CMM), Ghent University, Belgium

ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

UNIVERSITEIT
GENT

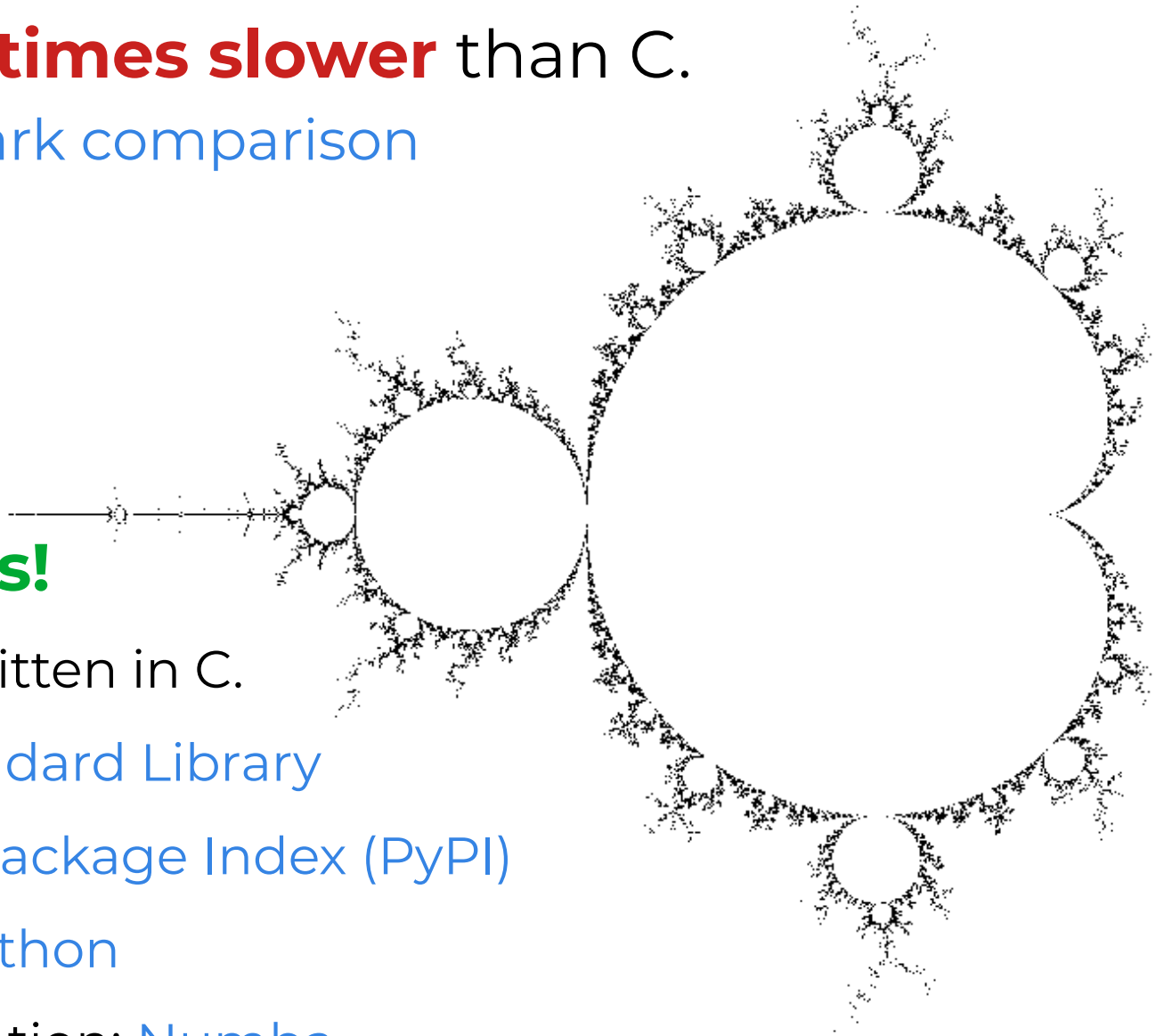
FACULTY
OF SCIENCES

CENTER FOR
MOLECULAR MODELING

Python is slow

Py3.7 **at least 150 times slower** than C.

Mandelbrot benchmark comparison



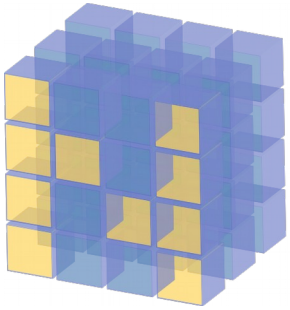
What can we do?

Python extensions!

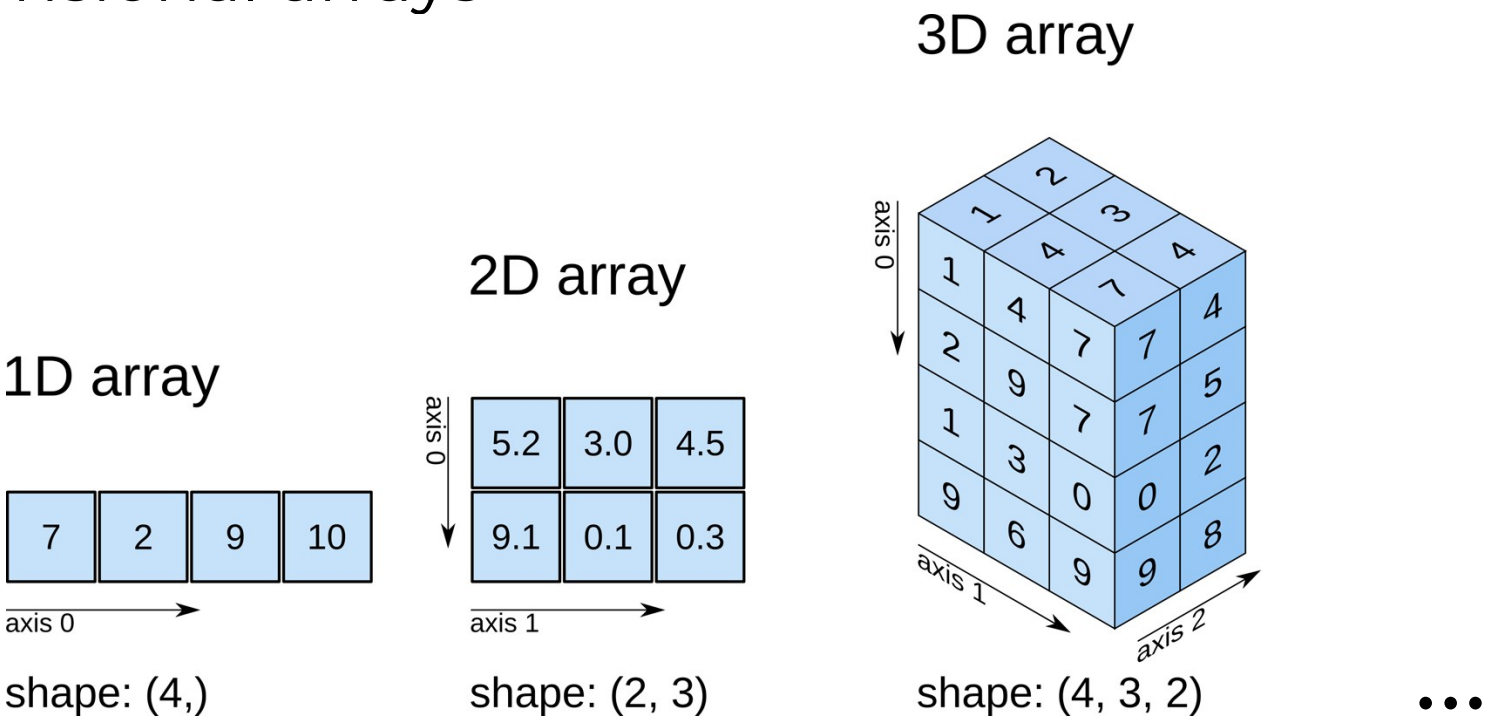
- = Python module written in C.
- Built-in: Python Standard Library
- Additional: Python Package Index (PyPI)
- Rolling your own: Cython
- Just-in-time compilation: Numba

Python Libraries for Scientific Computation

NumPy	**Numerical array library**
SciPy	Scientific Computation
MatPlotLib	2D (and 3D) plotting
Pandas	Mixed-type datasets & analysis
H5Py	Binary cross-platform array file format
Cython	Python C++ interface
Scikit-learn	(Old-school) machine learning & statistics
RDKit	Cheminformatics
SymPy	Symbolic calculus
Dask	Parallel workflows
Autograd	Algorithmic differentiation
Numba	Just-in-time compiler for Python



Main feature: Efficient computation with N-dimensional arrays



...

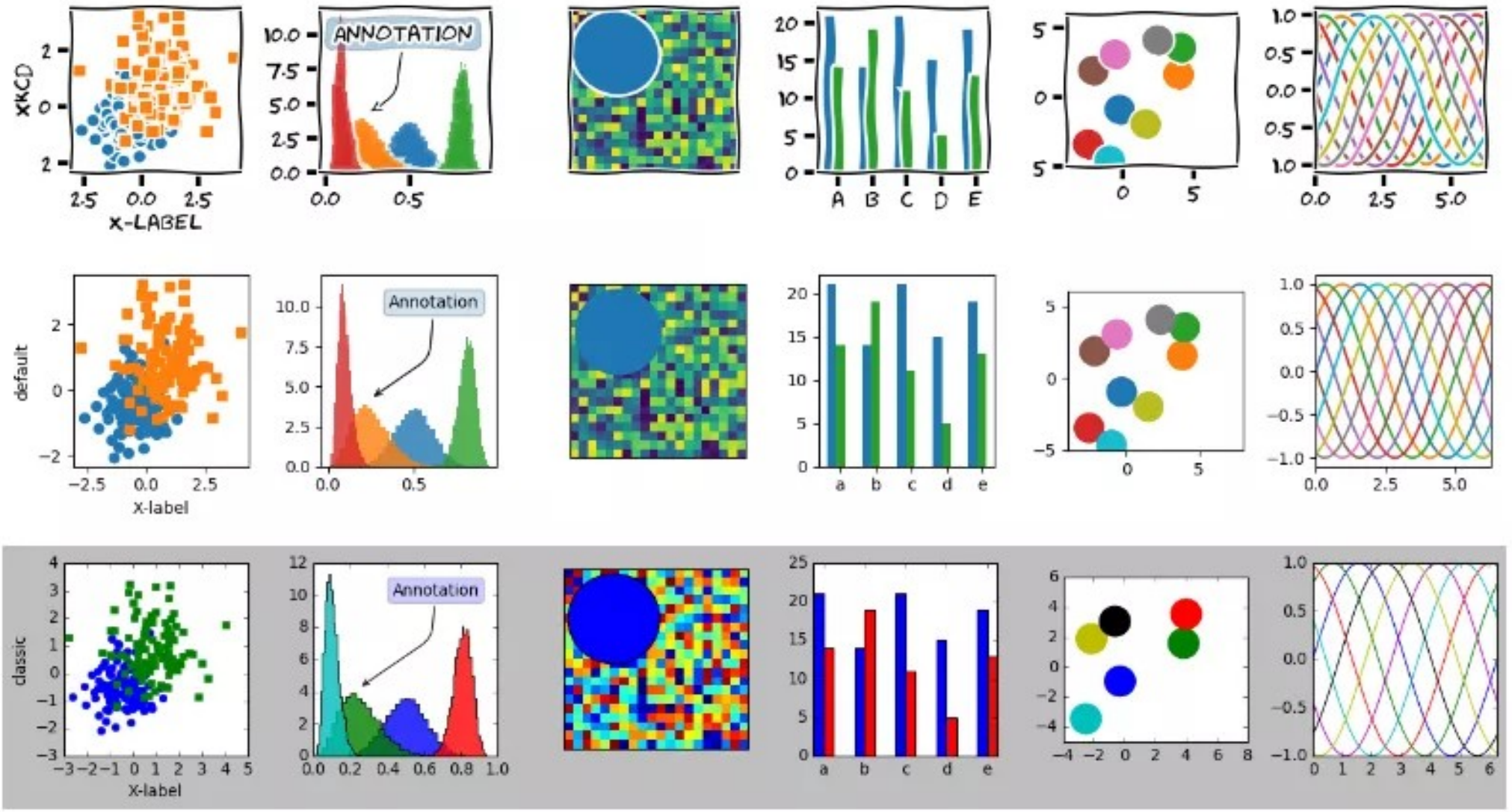
Related features: linear algebra (BLAS and LAPACK), fast Fourier transform, random numbers, polynomials & basic statistics

Bag of scientific computational tools

- Clustering package
- Constants
- Discrete Fourier transforms
- Integration and ODEs
- Interpolation
- Input and output
- Linear algebra
- Miscellaneous routines
- Multi-dimensional image processing
- Orthogonal distance regression
- Optimization and Root Finding
- Signal processing
- Sparse matrices
- Sparse linear algebra
- Compressed Sparse Graph Routines
- Spatial algorithms and data structures
- Special functions
- Statistical functions

MatPlotLib

Very powerful 2D (and 3D) plotting, also interactive



AutoGrad

- =Algorithmic (≠symbolic) differentiation of Python code.
- Analytic derivatives, not approximate.
- Popular for machine learning, but generally useful.

Time for hands-on

- Matrix-matrix multiplication in pure Python
- Matrix-matrix multiplication with NumPy
- Creating NumPy arrays
- Functions & ufuncs
- Accessing elements and slices
- Array contractions
- Basic plotting
- Numerical quadrature
- Numerical optimization: Rosenbrock
- Algorithmic differentiation

Hands-on practice

Tiny DFT

Toon Verstraelen

Center for Molecular Modeling (CMM), Ghent University, Belgium

ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

Tiny DFT

- Atomic Kohn-Sham Density Function Theory (KS-DFT) program in Python
- Uses only NumPy, SciPy, Matplotlib & Autograd
- 400 lines (with comments and docstrings)
- Closed shell & spherical atoms only

Slater determinant

- Impose anti-symmetry
- Orbitals can be chosen orthonormal without loss of generality
- subset of all anti-symmetric wavefunctions

$$\Psi_s(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_1) & \cdots & \psi_N(\mathbf{x}_1) \\ \psi_1(\mathbf{x}_2) & \psi_2(\mathbf{x}_2) & \cdots & \psi_N(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ \psi_1(\mathbf{x}_N) & \psi_2(\mathbf{x}_N) & \cdots & \psi_N(\mathbf{x}_N) \end{vmatrix}$$

Essentially a minimization problem...

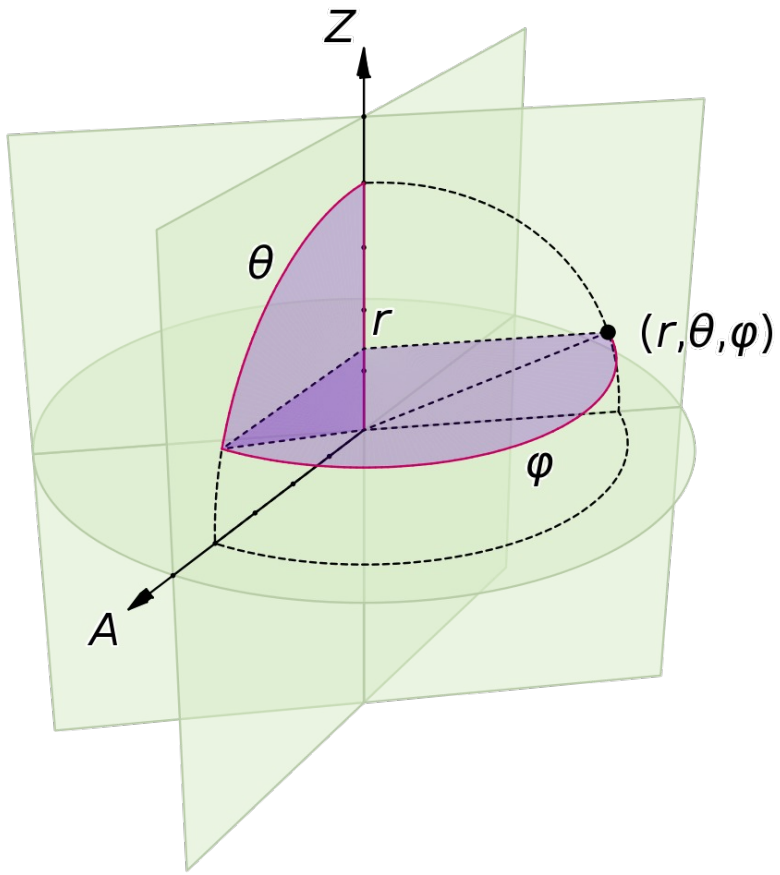
$$E_{\mathrm{gs}} = \min_{\Psi_s} \left[\left\langle \Psi_s \right| -\frac{1}{2}\nabla^2 \left| \Psi_s \right\rangle + J\left[\rho[\Psi_s]\right] + E_{\mathrm{xc}}[\rho[\Psi_s]] + \int \rho[\Psi_s](\mathbf{r}) V_{\mathrm{ext}}(\mathbf{r}) d\mathbf{r} \right]$$

...usually solved as an eigenvalue problem:

$$\left(-\frac{1}{2}\nabla^2 + \underbrace{V_{\mathrm{ext}}(\mathbf{r}) + \frac{\delta J[\rho]}{\delta\rho(\mathbf{r})} + \frac{\delta E_{\mathrm{xc}}[\rho]}{\delta\rho(\mathbf{r})}}_{V_{\mathrm{KS}}[\rho](\mathbf{r})} \right) \psi_i(\mathbf{x}) = \epsilon_i \psi_i(\mathbf{x})$$

Kohn-Sham DFT in spherical coordinates

- At every SCF iteration: spherically averaged density
- Central potential ⇒ spherical coordinates



Reduction to a radial problem

$$\psi_{\ell m n}(\mathbf{r}) = Y_\ell^m(\varphi,\theta) R_n(r) = Y_\ell^m(\varphi,\theta)\frac{U_{n\ell}(r)}{r}$$

$$\left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{\ell(\ell+1)}{2r^2} + V_{\mathrm{KS}}[\rho](r)\right] U_{n\ell}(r) = \epsilon_{n\ell} U_{n\ell}(r)$$

$$\rho_{\mathrm{spher}}(r) = \frac{1}{4\pi} \sum_{\substack{n\ell m \in \\ \text{occupied}}} \frac{|U_{n\ell}(r)|^2}{r^2}$$

The Hartree potential and energy

$$J[\rho] = \frac{1}{2}\iint \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,\mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}' \qquad V_h(\mathbf{r}) = \frac{\delta J[\rho]}{\delta\rho(\mathbf{r})} = \int \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}'$$

$$-\nabla^2 V_h(\mathbf{r}) = 4\pi\rho(\mathbf{r}) \qquad -\frac{\mathrm{d}^2[rV_h(r)]}{\mathrm{d}r^2} = 4\pi r\,\rho_{\mathrm{spher}}(r)$$

$$rV_h(r) = Ar + B + 4\pi\int_0^r \mathrm{d}r' \int_0^{r'} \mathrm{d}r'' r''\rho_{\mathrm{spher}}(r'')$$

$\lim_{r\to\infty} rV_h(r) = N_{\mathrm{elec}}$ $\quad$ $V_h(r)$ behaves like N_{elec}/r at large distances

$\lim_{r\to 0} rV_h(r) = 0$ $\quad$ $V_h(r)$ is finite at the origin

---

$$J[\rho_{\mathrm{spher}}] = \frac{1}{2}4\pi\int_0^\infty r^2\rho_{\mathrm{spher}}(r)V_h(r)\mathrm{d}r$$

Remaining terms

$$E_{\mathrm{xc}}[\rho] = \int \rho(\mathbf{r})\epsilon_{xc}(\rho(\mathbf{r}))\mathrm{d}\mathbf{r} = \int e_{xc}(\rho(\mathbf{r}))\mathrm{d}\mathbf{r}$$

xc

$$V_{\mathrm{xc}}[\rho] = \frac{\delta E_{\mathrm{xc}}[\rho]}{\delta\rho(\mathbf{r})} = \frac{\partial e_{xc}}{\partial\rho}(\mathbf{r})$$

$$E_{\mathrm{xc}}[\rho_{\mathrm{spher}}] = 4\pi\int_0^\infty r^2 e_{xc}(\rho(r))\mathrm{d}r$$

---

ext

$$E_{\mathrm{ext}}[\rho_{\mathrm{spher}}] = -4\pi\int_0^\infty r^2\frac{Z\rho(r)}{r}\mathrm{d}r$$

Transformation *r(x)* from *x* ∈ [−1, 1] to *r* ∈ [0, +∞]:

$$\int_0^\infty f(r)\mathrm{d}r = \int_{-1}^{1} f(r(x))\frac{\mathrm{d}r}{\mathrm{d}x}\mathrm{d}x \approx \sum_i w_i f(r(x_i))\frac{\mathrm{d}r}{\mathrm{d}x}(x_i)$$

where the last term uses Gauss-Chebyshev weights and nodes.

Spectral method for derivative and anti-derivative:

1. Fit a polynomial through grid data.
2. (Anti-)Derivative of the polynomial.
3. Evaluate resulting polynomial back on the grid.

Time for hands-on

Go to

https://github.com/theochem/tinydft

and follow the README.

Hands-on tutorial

(Hirshfeld) Atoms-in-Molecules

Toon Verstraelen

Center for Molecular Modeling (CMM), Ghent University, Belgium

ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

UNIVERSITEIT
GENT

FACULTY
OF SCIENCES

CENTER FOR
MOLECULAR MODELING

Atoms in Molecules (AIMs)

Broad definition

Partitioning of molecular properties over atomic contributions (or atom pairs, triplets, ...)

Why?

1) Chemists often think in terms of atoms.
2) Distributed multipole expansions

Ambiguity

- Many AIM or fitting methods exist.
- Any *rigorous* definition is contestable.
- At best, one can define desirable properties.

Mathematically elegant & concise

Non-empirical

Optimality & Uniqueness

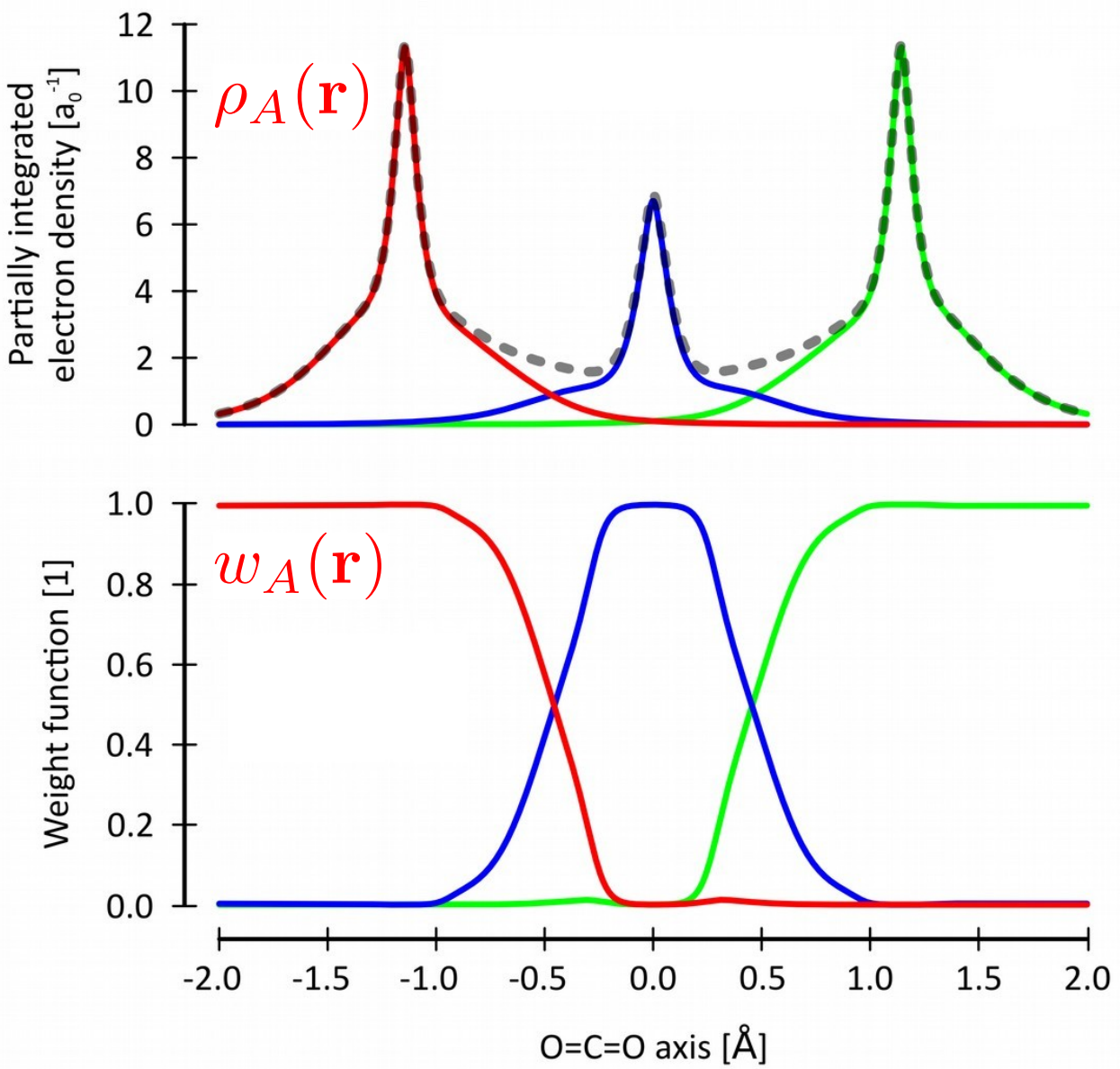
Local atomic densities

Rapidly converging atomic multipoles

Numerical & conformational robustness

Universally applicable

The Hirshfeld Method – Definition (1977)



$$\rho_A(\mathbf{r}) = \rho(\mathbf{r}) w_A(\mathbf{r})$$

$$w_A(\mathbf{r}) = \frac{\rho_A^0(\mathbf{r})}{\sum_B \rho_B^0(\mathbf{r})}$$

*[Hirshfeld, Theor. Chem. Acc. **1977**, v44, p129]*

Hirshfeld partitioning minimizes KL-divergence

$$\Delta S[\{\rho_A\};\{\rho_A^0\}] = \sum_{A=1}^{N_{\text{atoms}}} \int \rho_A(\mathbf{r}) \ln \frac{\rho_A(\mathbf{r})}{\rho_A^0(\mathbf{r})} d\mathbf{r}$$

subject to $\rho(\mathbf{r}) = \sum_{A=1}^{N_{\text{atoms}}} \rho_A(\mathbf{r}) \qquad \forall \mathbf{r} \in \mathbb{R}$

Iterative Hirshfeld (2007)

Charged pro-atoms:
linear interpolation

...
dication
cation
neutral
anion
dianion
...

*[Bultinck, Van Alsenoy, Ayers, Carbó-Dorca, JCP **2007**, v126, p144111]*

Iterative Stockholder (2008)

1) Initial guess pro-atoms

2) Hirshfeld partitioning

3) Spherical average of AIM

Charged pro-atoms:
spherical
average of AIM

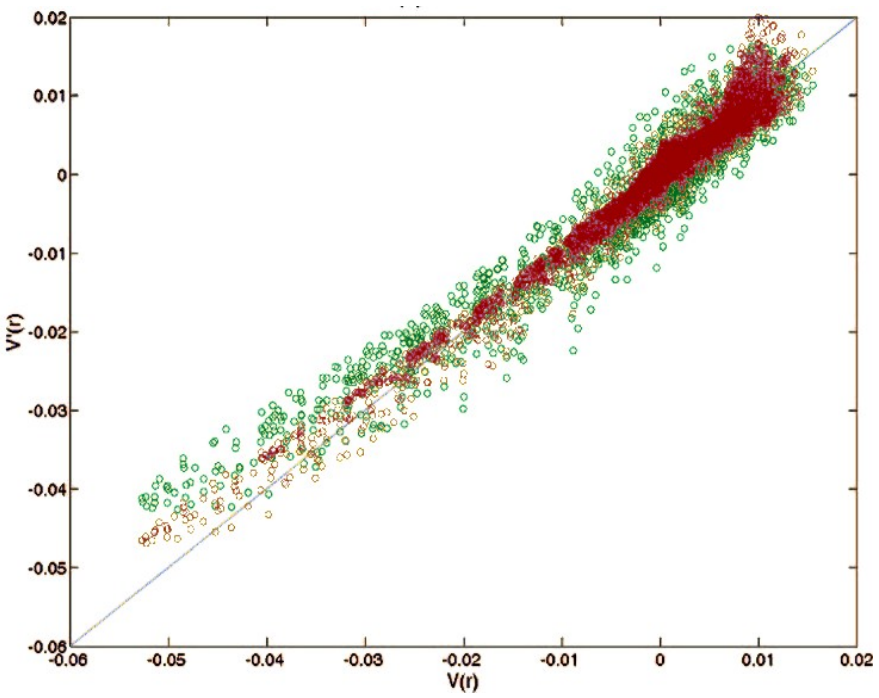
$$\rho_A^0(\mathbf{r}) = f_A(|\mathbf{r} - \mathbf{R}_A|)$$

$$f_A(r) = \frac{1}{4\pi r^2} \int \rho_A(\mathbf{r}) \delta(r - |\mathbf{r} - \mathbf{R}_A|) d\mathbf{r}$$

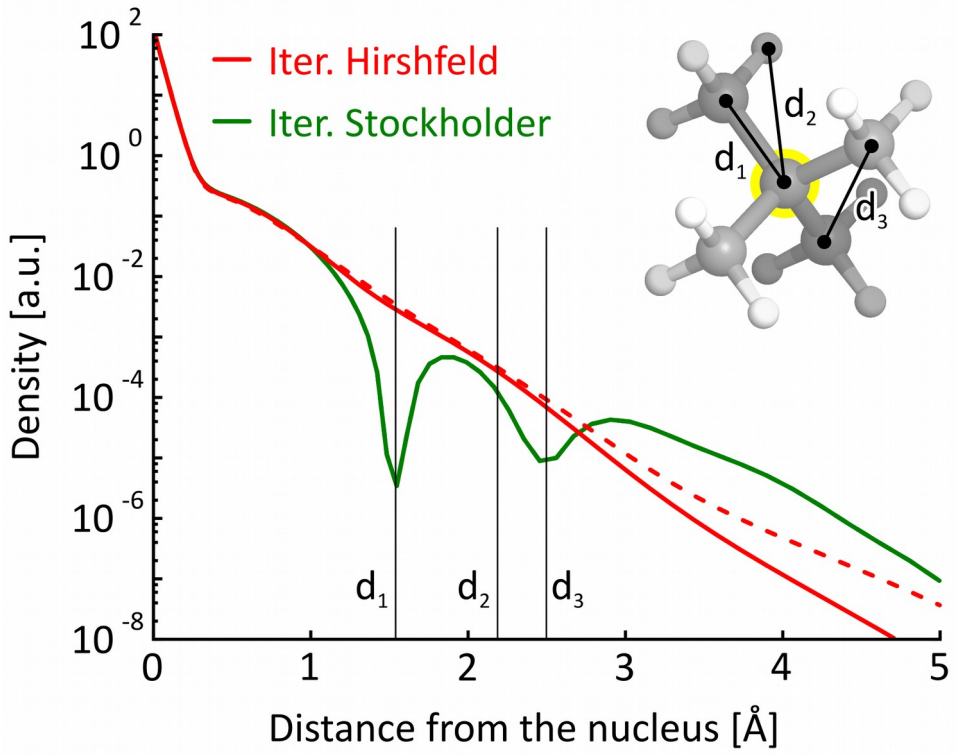
*[Lillestolen & Wheatley Chem. Commun. **2008**, v44, p5909]*

State of the art in 2008

Electrostatic potentials:
Hirshfeld-I versus MSK



Numerical instability
Iterative Stockholder
Analysis



*[S. Van Damme, P. Bultinck, and S. Fias, JCTC **2009**, v5, p334]*

State of the art in 2008

	Hirshfeld	Iterative Hirshfeld	Iterative Stockholder
Robustness	+++	++	–
Electrostatic potentials	–	++	+++

After 2008 this topic exploded!

- Further improve robustness & electrostatics
- Dealing with unstable anions
- Improved foundations in information theory
- *My method is bigger than yours!*
- …

Relative entropy

$$\Delta S = \int f(\boldsymbol{r}) \ln \frac{f(\boldsymbol{r})}{m(\boldsymbol{r})} \, \mathrm{d}\boldsymbol{x}$$

"real" distribution

"model" distribution, prior or Lebesgue measure

with $$\int f(\boldsymbol{r}) \, \mathrm{d}\boldsymbol{r} = \int m(\boldsymbol{r}) \, \mathrm{d}\boldsymbol{r}, \quad f(\boldsymbol{r}) > 0, \quad m(\boldsymbol{r}) > 0$$

Minimization of ΔS:

1) Find best *m* for given *f*.

2) Find best *f*, given measurements (constraints) and prior *m*.

3) Combination of (1) and (2)

Why relative entropy?

- Additive

  Contributions from disjoint domains in $\boldsymbol{r}$ add up. (for ΔS, f and m)

- Coordinate invariance

- No accidental correlations

  In absence of evidence for correlations, f and m are a product of marginal distributions.

- Without data or restrictions on f or m: $f = m$

- Lagrange multipier for normalization = −1

  So for optimization of f and/or m one may minimize:

$$\Delta S[f, m] = \int m(\boldsymbol{r}) - f(\boldsymbol{r}) + f(\boldsymbol{r}) \ln \frac{f(\boldsymbol{r})}{m(\boldsymbol{r})} \, \mathrm{d}\boldsymbol{r}$$

J. Skilling “The Axioms of Maximum Entropy”, in “Maximum-Entropy and Bayesian Methods in Science and Engineering”, 1988, pp 173—178.

Principal idea: start from relative entropy.

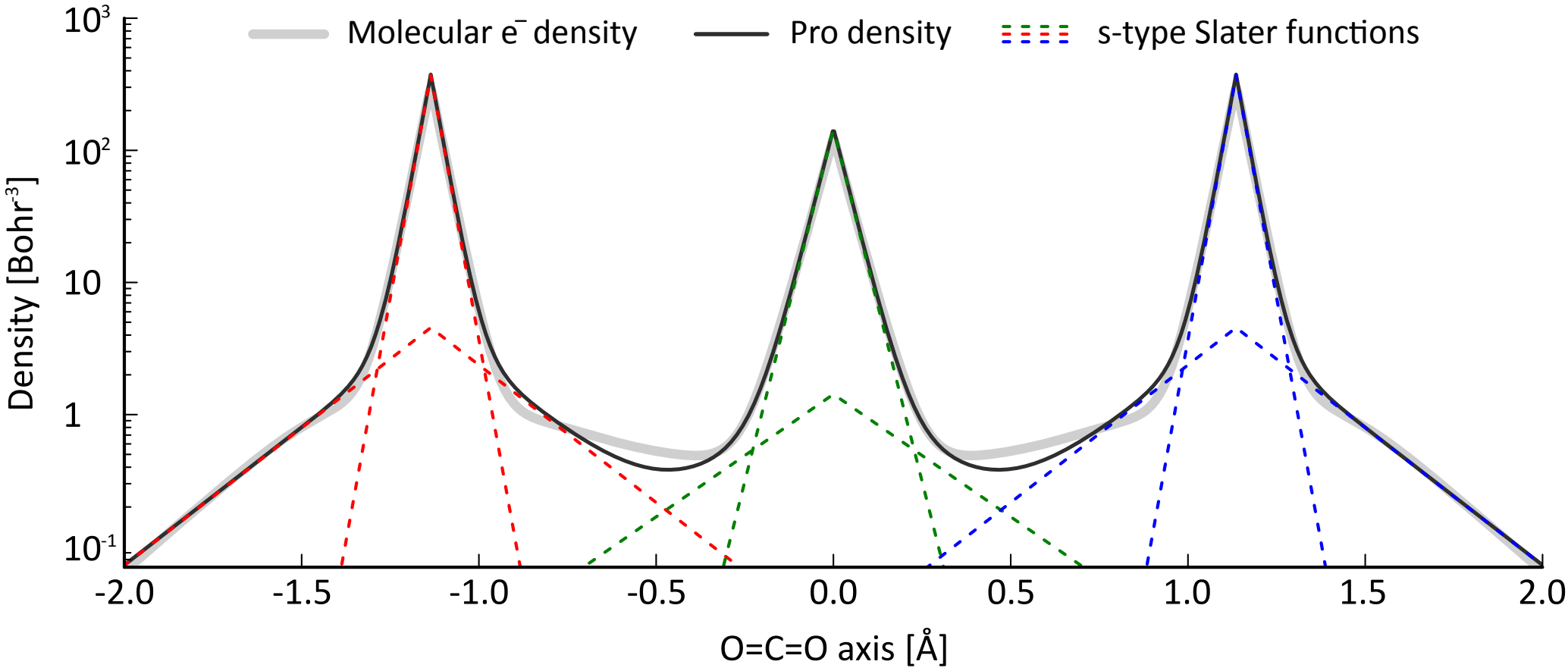
Minimize $\Delta S[\rho, \rho^0] = \int \rho(\boldsymbol{r}) \ln \frac{\rho(\boldsymbol{r})}{\rho^0(\boldsymbol{r}; \{\alpha_{Ai}\})} \, \mathrm{d}\boldsymbol{r}$

subject to $\int \rho(\boldsymbol{r}) \, \mathrm{d}\boldsymbol{r} = \int \rho^0(\boldsymbol{r}; \{\alpha_{Ai}\}) \, \mathrm{d}\boldsymbol{r}, \quad \rho > 0, \quad \rho^0 > 0$

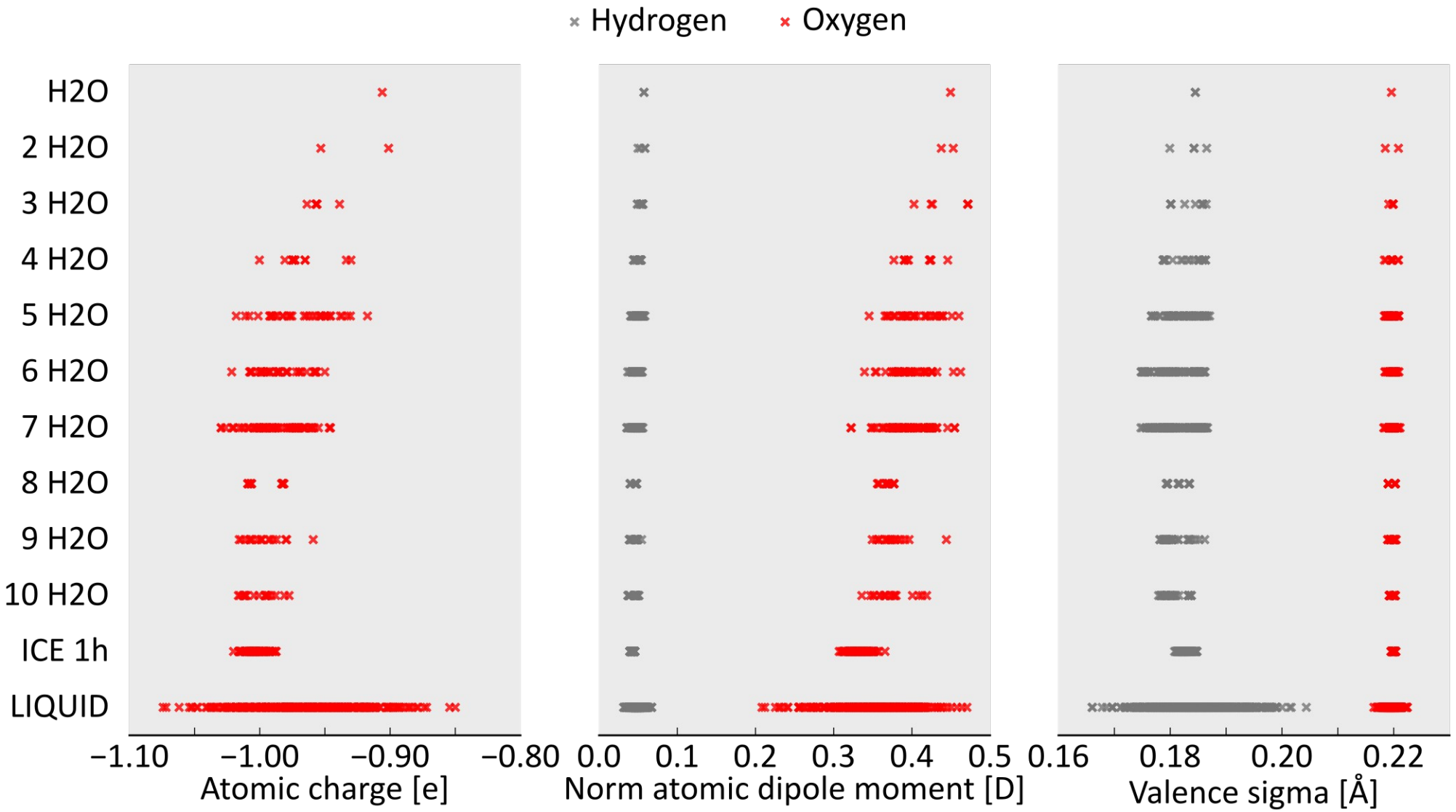
with $\rho^0(\boldsymbol{r}; \{\alpha_{Ai}\}) = \sum_{A=1}^{N_{\mathrm{atom}}} \rho_A^0(\boldsymbol{r}; \{\alpha_{Ai}\})$

Minimal Basis Iterative Stockholder

$$\rho^0(\boldsymbol{r}) = \sum_{A=1}^{N_{\text{atoms}}} \sum_{i=1}^{m_A} \frac{N_{Ai}}{8\pi\sigma_{Ai}^3} \exp\left(-\frac{|\boldsymbol{r} - \boldsymbol{R}_i|}{\sigma_{Ai}}\right)$$

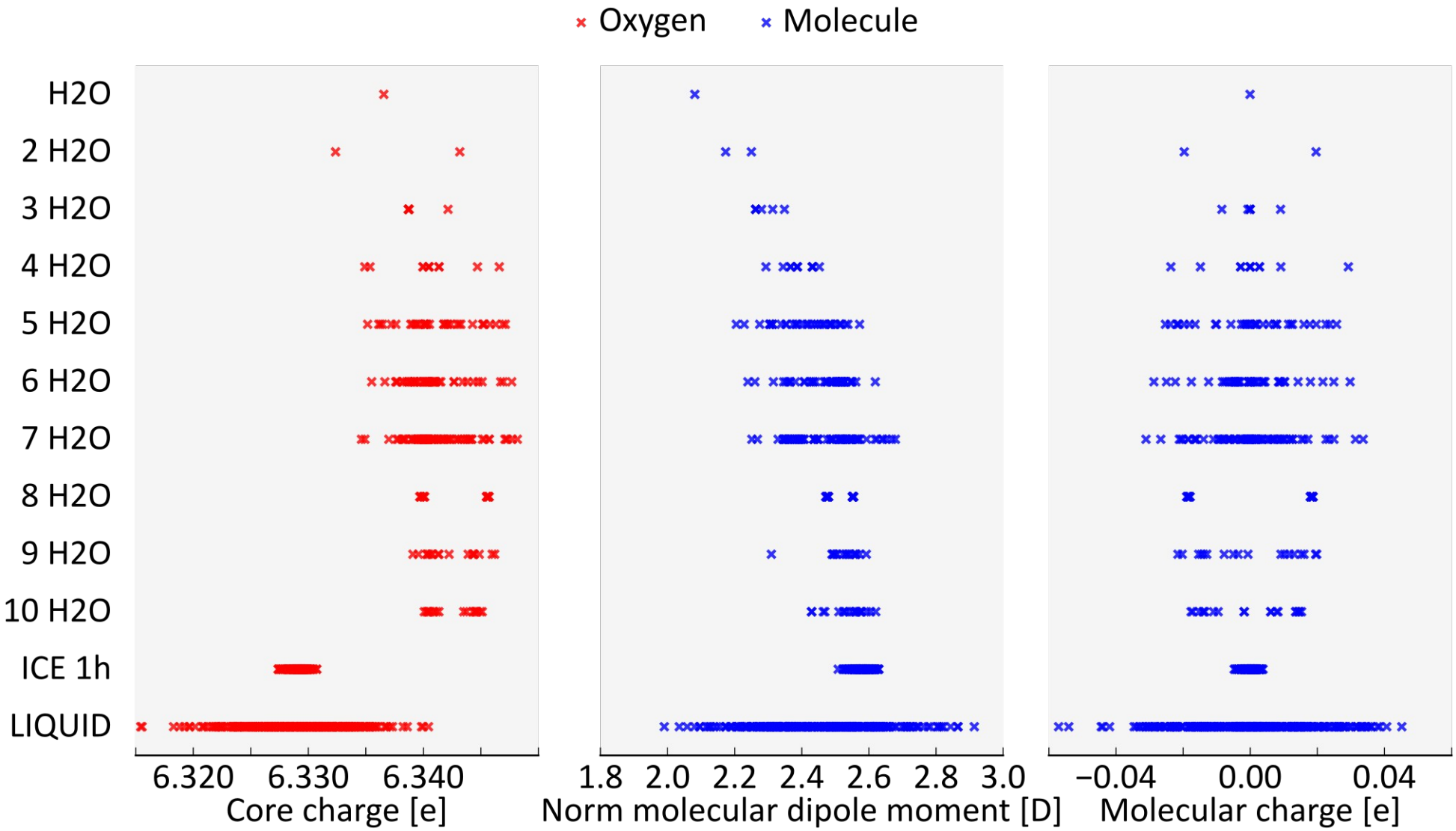


Typical results for water (1)



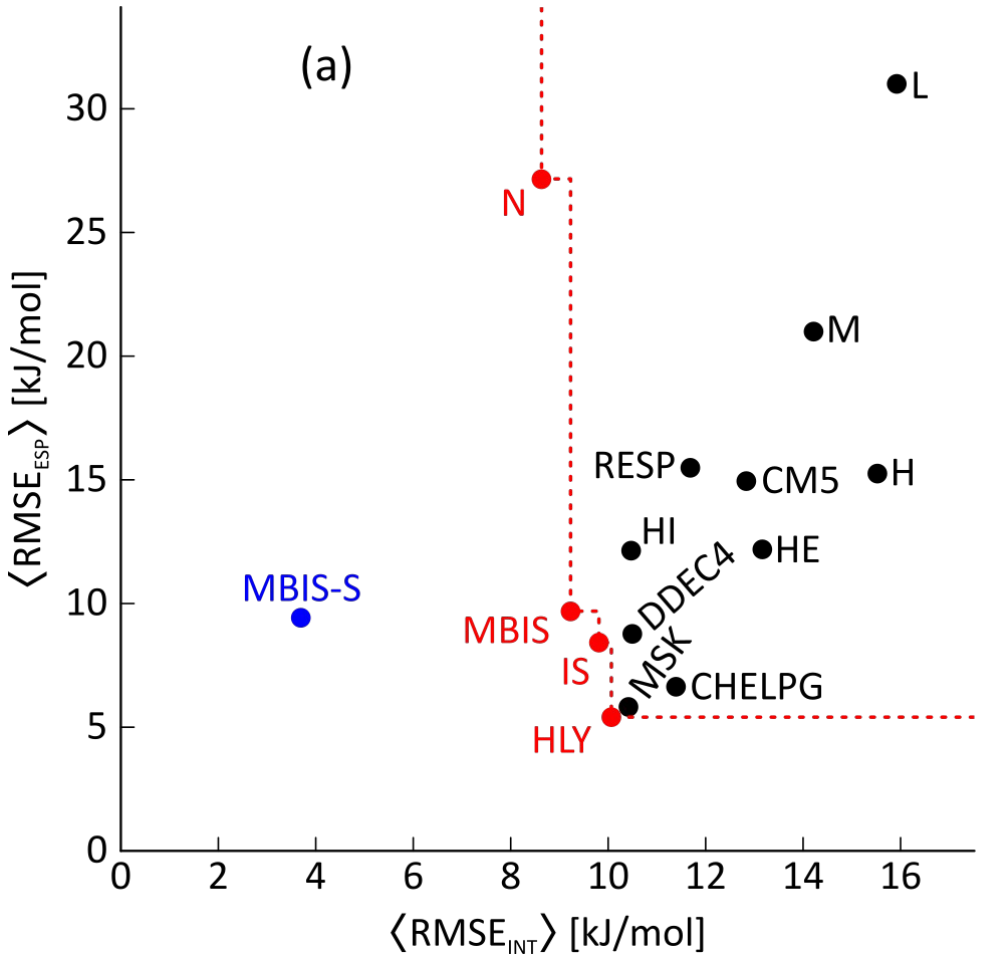
Computational details: (i) isolated water: geo,rho BLYP/6-311++G(2df,p), (ii) clusters: geo MP2/aug-CC-pVDZ, rho BLYP/6-311++G(2df,p) [Temelso et al JPCA v115 p12034 y2011], (iii) NVT 300k BLYP/DZVP-MOLOPT 100ps, 10 snapshots, rho BLYP GPAW (h=0.1), ice 1h geo [Hayward and Reimers JCP v106 p1518 y1997], geo tuned BLYP/DZVP-MOLOPT, rho BLYP GPAW (h=0.1)

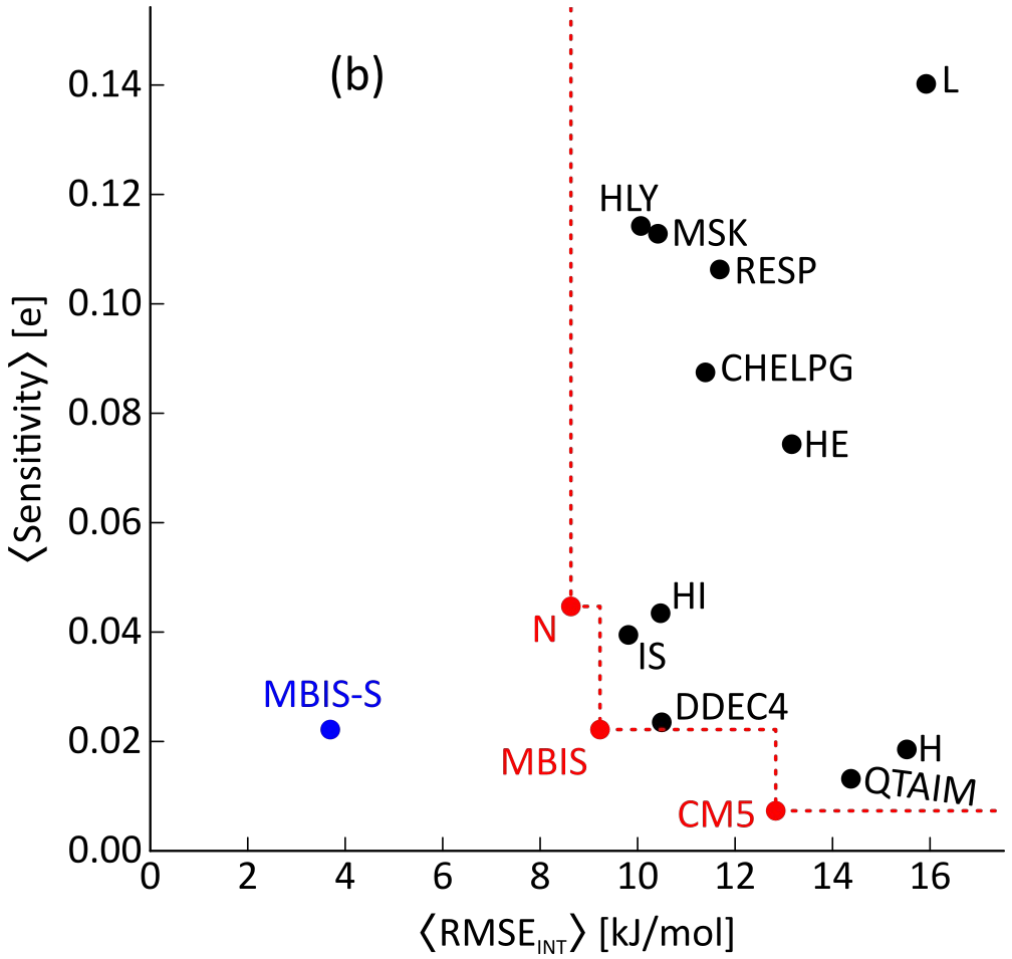
Typical results for water (2)



Computational details: (i) isolated water: geo,rho BLYP/6-311++G(2df,p), (ii) clusters: geo MP2/aug-CC-pVDZ, rho BLYP/6-311++G(2df,p) [Temelso et al JPCA v115 p12034 y2011], (iii) NVT 300k BLYP/DZVP-MOLOPT 100ps, 10 snapshots, rho BLYP GPAW (h=0.1), ice 1h geo [Hayward and Reimers JCP v106 p1518 y1997], geo tuned BLYP/DZVP-MOLOPT, rho BLYP GPAW (h=0.1)

Benchmark results





Hands-on tutorial & practice

IOData, Grid & GBasis

Toon Verstraelen

Center for Molecular Modeling (CMM), Ghent University, Belgium

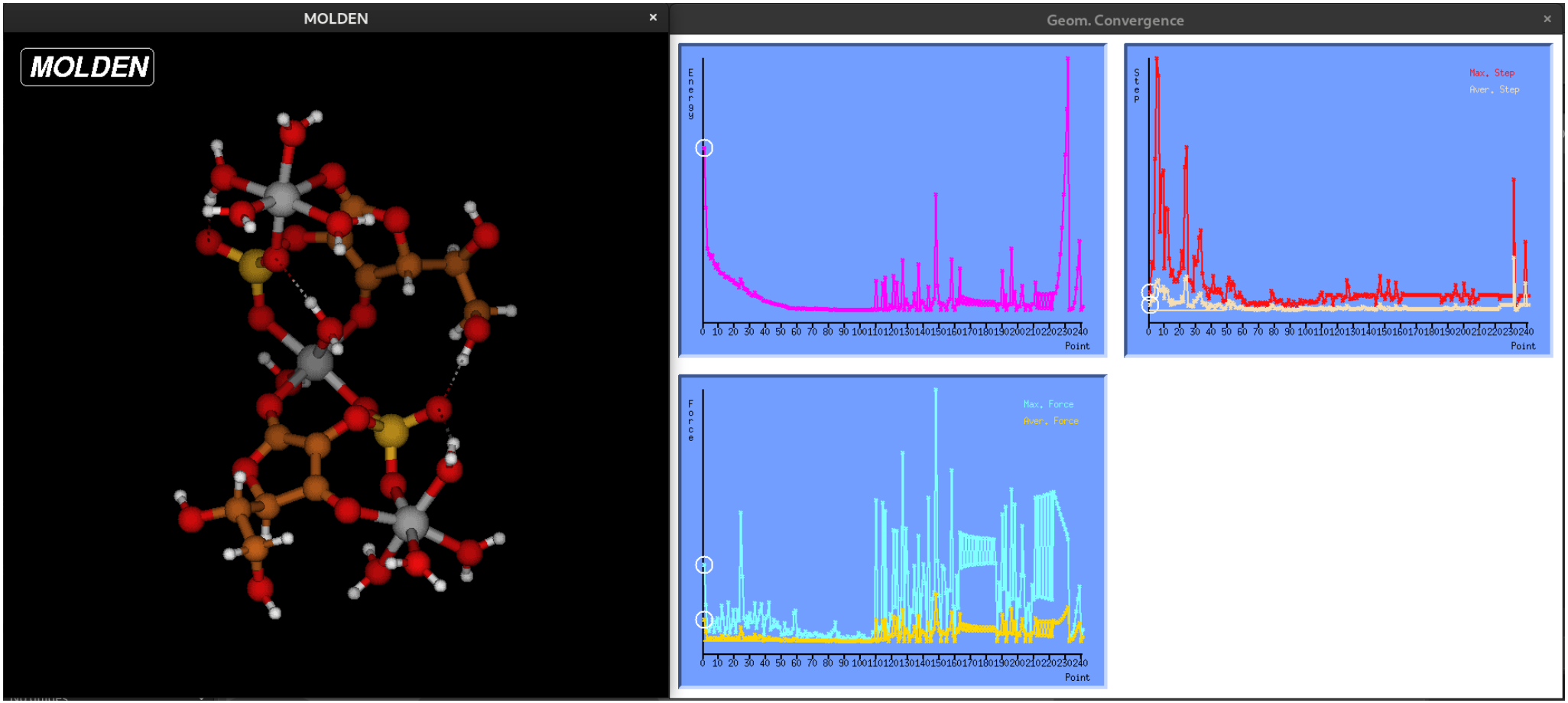
ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

UNIVERSITEIT
GENT

FACULTY
OF SCIENCES

CENTER FOR
MOLECULAR MODELING

Gaussian users will recognize this...



Problem: best geometry for restart?

Hands-on

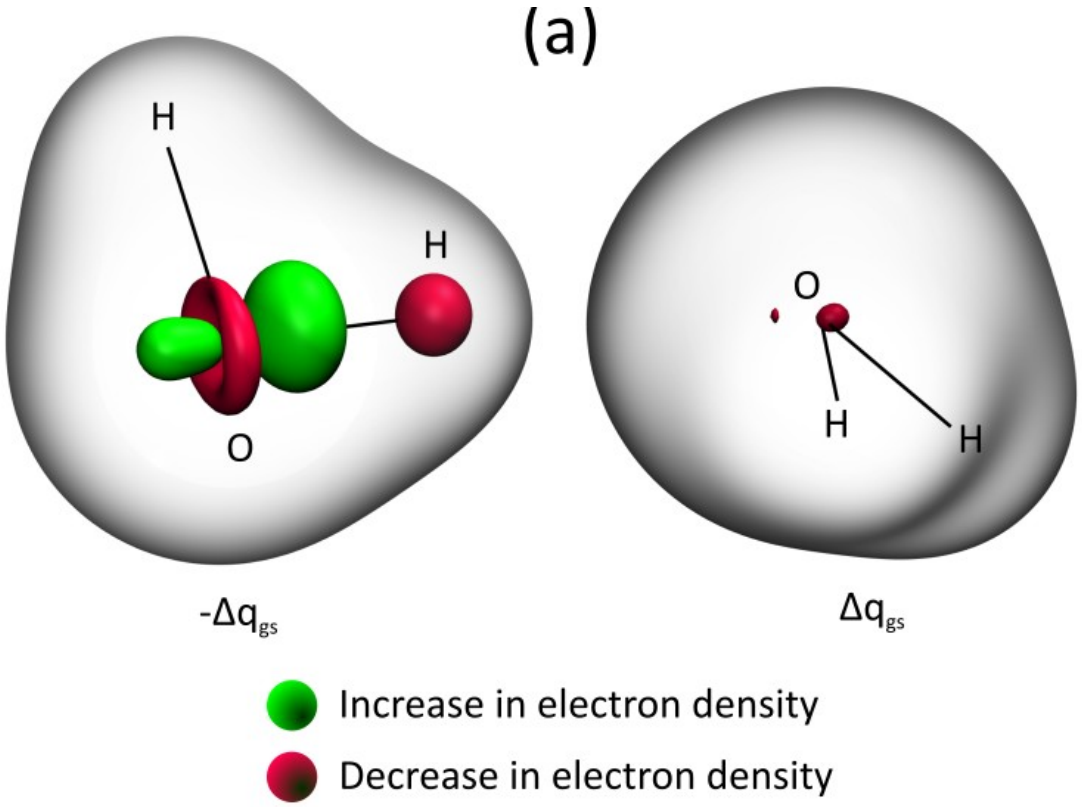
Tutorial

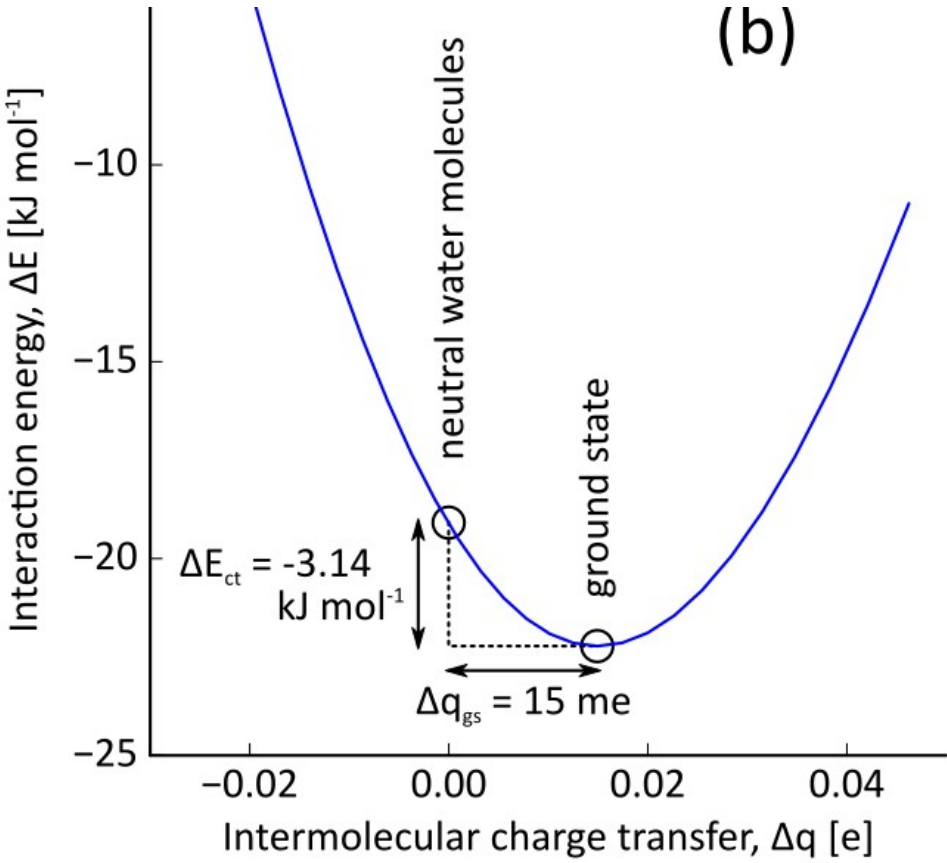
- Load FCHK file with IOData
- Select geometry with lowest energy gradient

Practice

- Select geometry with lowest rmsd gradient

Molecular Hirshfeld





$$\rho_A(\mathbf{r}) = \frac{\rho_A^0(\mathbf{r})}{\rho_A^0(\mathbf{r}) + \rho_B^0(\mathbf{r})}\rho(\mathbf{r})$$

$$N_A = \int \rho_A(\mathbf{r})\mathrm{d}\mathbf{r}$$

Hands-on

Tutorial

- Load FCHK files with IOData of:
  - isolated water molecule A
  - isolated water molecule B
  - water dimer AB
- Construct grid for AB
- Evaluate densities of A, B and AB on grid
- Compute partitioned densities and populations

Practice

- Repeat for ion pair
- Implement molecular Iterative Hirshfeld for ion pair

Bonus round

How to become a Python developer, without messing up?

Toon Verstraelen

Center for Molecular Modeling (CMM), Ghent University, Belgium

ChemTools Workshop, Sorbonne Université, Paris
May 20-24, 2019

UNIVERSITEIT GENT

FACULTY OF SCIENCES

CENTER FOR MOLECULAR MODELING

You know Python. Great!

Now the real fun begins...

- Reliable software?
- Managing complexity?
- Sustainable software development?


¿How to debug?
¿How to avoid bugs?


Oh no! I wrote
this code when I was 25.
This is going
to be a long night!
imgflip.com

Write code in style

- = rules for readability
- In general, adhere to PEP8
  https://www.python.org/dev/peps/pep-0008/
- Tools:
  - `pip install --user` **`pycodestyle`**
    Checks a subset of PEP8 rules
  - `pip install --user` **`pydocstyle`**
    Checks a subset of PEP257 (docstrings)
  - `pip install --user` **`pylint`**
    Checks subset of PEP8 and other things.
  - Cardboardlint => our wrapper for many checkers

Write transparent code

- **Single line of code = self-explaining**
  - Give variables, functions, … **sensible names**.
  - **Not too much** stuff in one line. No crazy one-liners.
- **Comments explain code** (implementation)
  - **English**, please.
  - Comment on **groups of lines**, rarely individual lines.
- **Docstrings explain usage of code** (API)
  - **Document** a function, class, module, …
  - **Describe** parameters, return values, exceptions & behavior

```
def fire_in_the_disco(msg):
    """Contributed by https://pythondev.slack.com/team/staticmethod
    This code was written for obfuscation contest.
    """
    reconstitute(msg,wwpd)
    try:
        f=type((lambda:(lambda:None for n in range(len(((((),((((),((((((((((
().next())
        u=(lambda:type((lambda:(lambda:None for n in
range(len(zip((((((((((((((((((((.func_code))()
        n=f(u(int(wwpd[4][1]),int(wwpd[7][1]),int(wwpd[6][1]),int(wwpd[9]
[1]),wwpd[2][1],
            (None,wwpd[10][1],wwpd[13][1],wwpd[11][1],wwpd[15][1]),(wwpd[20]
[1],wwpd[21][1]),
            (wwpd[16][1],wwpd[17][1],wwpd[18][1],wwpd[11][1],wwpd[19]
[1]),wwpd[22][1],wwpd[25][1],int(wwpd[4][1]),wwpd[0][1]),
            {wwpd[27][1]:__builtins__,wwpd[28][1]:wwpd[29][1]})
        c=partial(n, [x for x in map(lambda i:n(i),range(int(0xbeef)))])
        FIGHT = f(u(int(wwpd[4][1]),int(wwpd[4][1]),int(wwpd[5]
[1]),int(wwpd[9][1]),wwpd[3][1],
                (None, wwpd[23][1]), (wwpd[14][1],wwpd[24][1]),(wwpd[12]
[1],),wwpd[22][1],wwpd[26][1],int(wwpd[8][1]),wwpd[1][1]),
                {wwpd[14][1]:c,wwpd[24][1]:urlopen,wwpd[27]
[1]:__builtins__,wwpd[28][1]:wwpd[29][1]})
        FIGHT(msg)
    except:
        pass
```

```
def compute_surface_polygon(x, y):
    """Compute the surface area of a 2D polygon.

    Parameters
    ----------
    x : np.array
        X-coordinates of the polygon's corners.
    y : np.array
        Y-coordinates of the polygon's corners.

    Returns
    -------
    area : type of x and y
        The surface area of the polygon.

    """
    # Shoelace algorithm, Meister, 1769
    if len(x) != len(y):
        raise TypeError("Arguments x and y must have the same length.")
    if len(x) <= 2:
        return 0.0
    else:
        return abs( x[-1]*y[0] + np.dot(x[:-1], y[1:])
                    -x[0]*y[-1] - np.dot(x[1:], y[:-1]))/2
```

$$A = \frac{1}{2}\left| x_N y_1 - x_1 y_N + \sum_{i=1}^{N-1} x_i y_{i+1} - x_{i+1} y_i \right|$$

Write unit tests

- = function to validate another function
- Runs fast, easy to start
- Write tests first, certainly not months later.
- Write many!
- Think of corner cases
- Coverage analysis = check if code is tested

```
def check_single(x, y, area):
    np.testing.assert_almost_equal(compute_area_polygon(x, y), area)

def check_variants(x, y, area):
    x = np.asarray(x)
    y = np.asarray(y)
    check_single(x, y, area)
    check_single(x[::-1], y[::-1], area)
    check_single(x + 0.3, y - 0.5478, area)
    check_single(-2*x, 0.8*y, 1.6*area)
    xp = np.cos(0.3)*x - np.sin(0.3)*y
    yp = np.sin(0.3)*x + np.cos(0.3)*y
    check_single(xp, yp, area)

def test_compute_area_polygon():
    # Simple geometries
    check_single([0, 0, 1, 1], [0, 1, 1, 0], 1.0)
    check_single([0.0, 0.0, 2.0], [0.0, 1.0, 1.0], 1.0)
    check_single([-0.5, 2.5, 1.0, 0.0], [0.0, 0.0, 0.5, 0.5], 1.0)

    # Corner cases: flat, coinciding points, too short vectors
    check_single([0.0, 2.0, -1.0], [0.0, 2.0, -1.0], 0.0)
    check_single([0.0, 0.0, 2.0, 2.0], [0.0, 1.0, 1.0, 1.0], 1.0)
    check_single([], [], 0.0)
    check_single([1], [2], 0.0)
    check_single([2.0, 1.0], [0.0, 0.0], 0.0)
```

Live demo

```
# Plain, without coverage
pytest -v meister.py

# With coverage analysis
pytest -v meister.py \
    --cov=meister \
    --cov-report term-missing
```

Optional static typing & mypy

From Python 3.5, one can add "type hints":

```
def compute_surface_polygon(
        x: np.ndarray, y: np.ndarray) -> float:
    ...
```

Type correctness can be checked with mypy.

Live demo

```
# Add type hints first and then ...
mypy meister.py
```

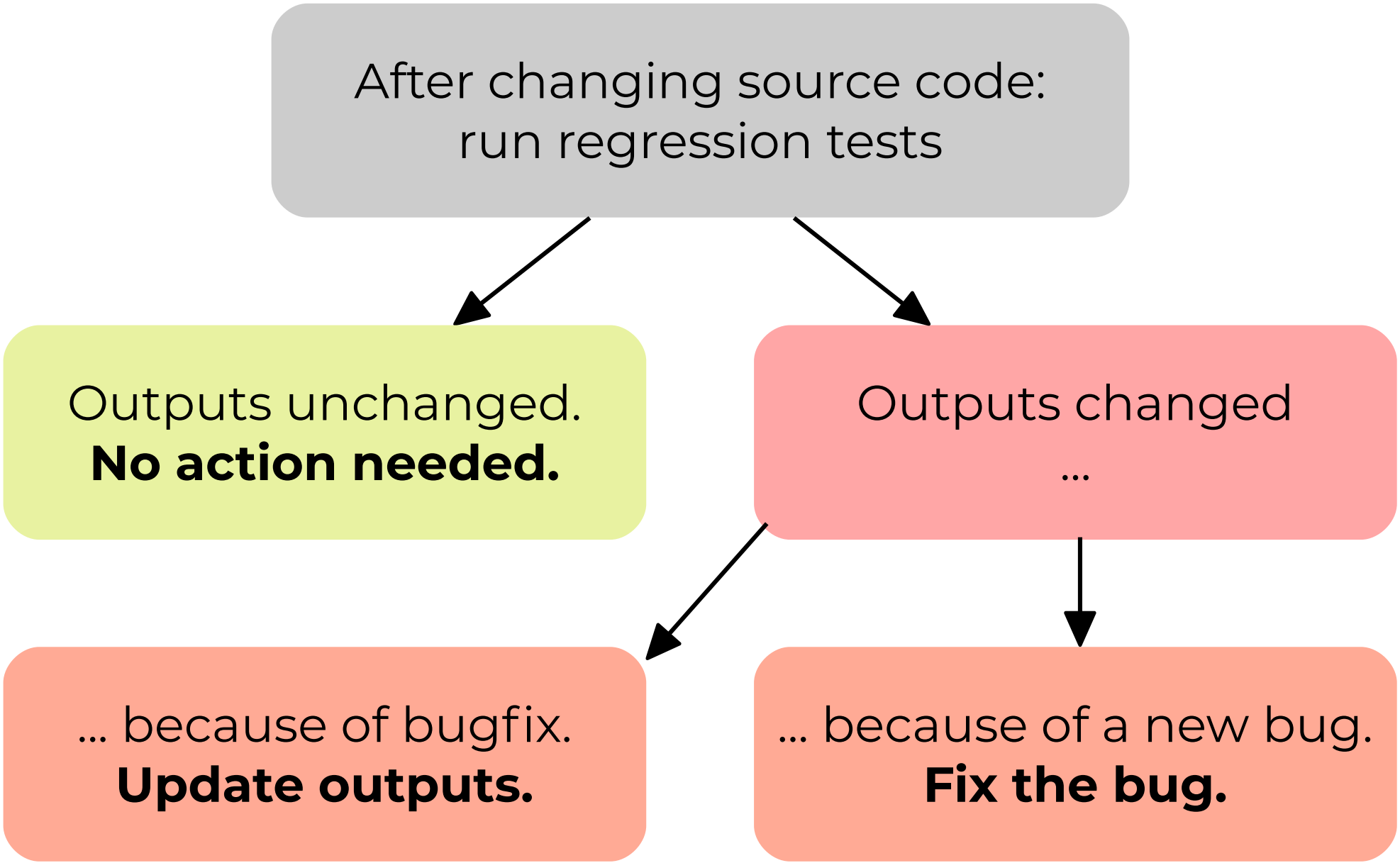
Write regression tests

- = tests for entire program
- Slower than unit tests
- Test whether program changes behavior.

Pairs of input and output
for every feature of your program

test1.in	test1.out
test2.in	test2.out
test3.in	test3.out
test4.in	test4.out
...	...

Regression test workflow



Hands-on: the Kabsch algorithm

1. Write the function signature & docstring.
2. Write one unit test.
3. Implement the Kabsch algorithm.
4. Write more unit tests.
5. Perform coverage analysis.
6. Corner cases?
7. Review your neighbour's code.

~~¿How to write complex software?~~
¿How to hide complexity?

Use boilerplate packages

Ideal for reducing code:

- argparse
  command-line argument parser
- collections, dataclasses, attrs
  facilitate implementation of datastructures
- glob & fnmatch
  UNIX-style pattern matching: "foo*_???.txt"
- json
  JSON = simple data representation, very widely used.
- pyyaml
  YAML = JSON generalization,
  better suited for humans

Live demo

```
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
p = Point(11, y=22)
p[0] + p[1]
x, y = p
x, y
p.x + p.y
p
p._replace(x=100)

Point = namedtuple('Point', ['x', 'y'], verbose=True)
```

Split code into modules

```python
# foo.py
def add(a, b):
    return a+b
```

```python
# bar.py
import foo
print(foo.add(1, 2))

from foo import add
print(add(1, 2))

from foo import *
```

Python package = group of modules

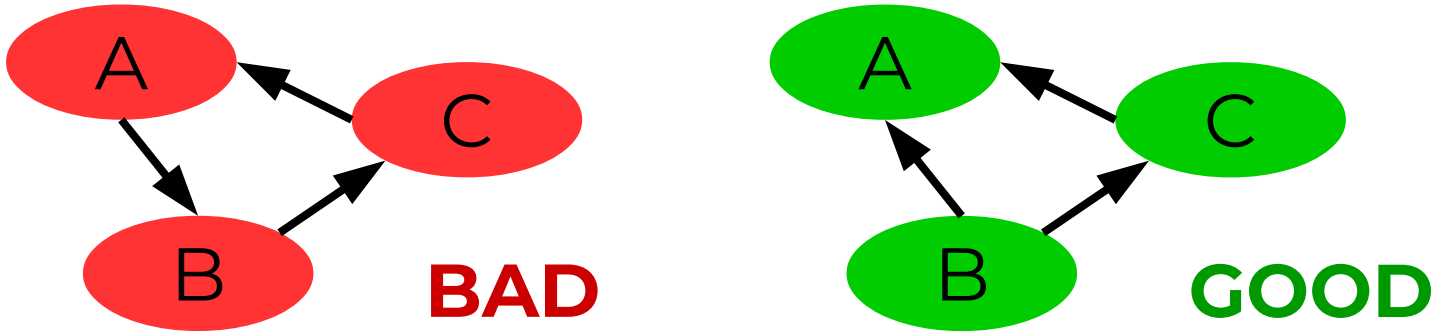
```
### Directory layout:
# bar.py
# pack/__init__.py
# pack/foo.py

# Content of pack/__init__.py
def subtract(a, b):
    return a-b

# Content of pack/foo.py
def add(a, b):
    return a+b

# bar.py
import pack.foo
print(pack.foo.add(1, 2))
import pack
print(pack.subtract(1, 2))
```

Installing & distributing packages

```
### Directory layout:
# setup.py
# pack/__init__.py
# pack/foo.py

# Content of setup.py
from distutils.core import setup

setup(name='Your package name',
      version='1.0',
      description='Bla bla bla',
      author='Your name',
      author_email='your.email@server.com',
      url='https://website.com',
      packages=['pack'])

# Run as folows:
python setup.py install
```

Make *modular* modules

1. **No cyclic dependencies** between modules.
   You use a module ⇒ module does not use you.



2. Modules should have a **minimal API**.
3. Modules should have a well-defined **purpose**, which can be **summarized in 1 sentence**.

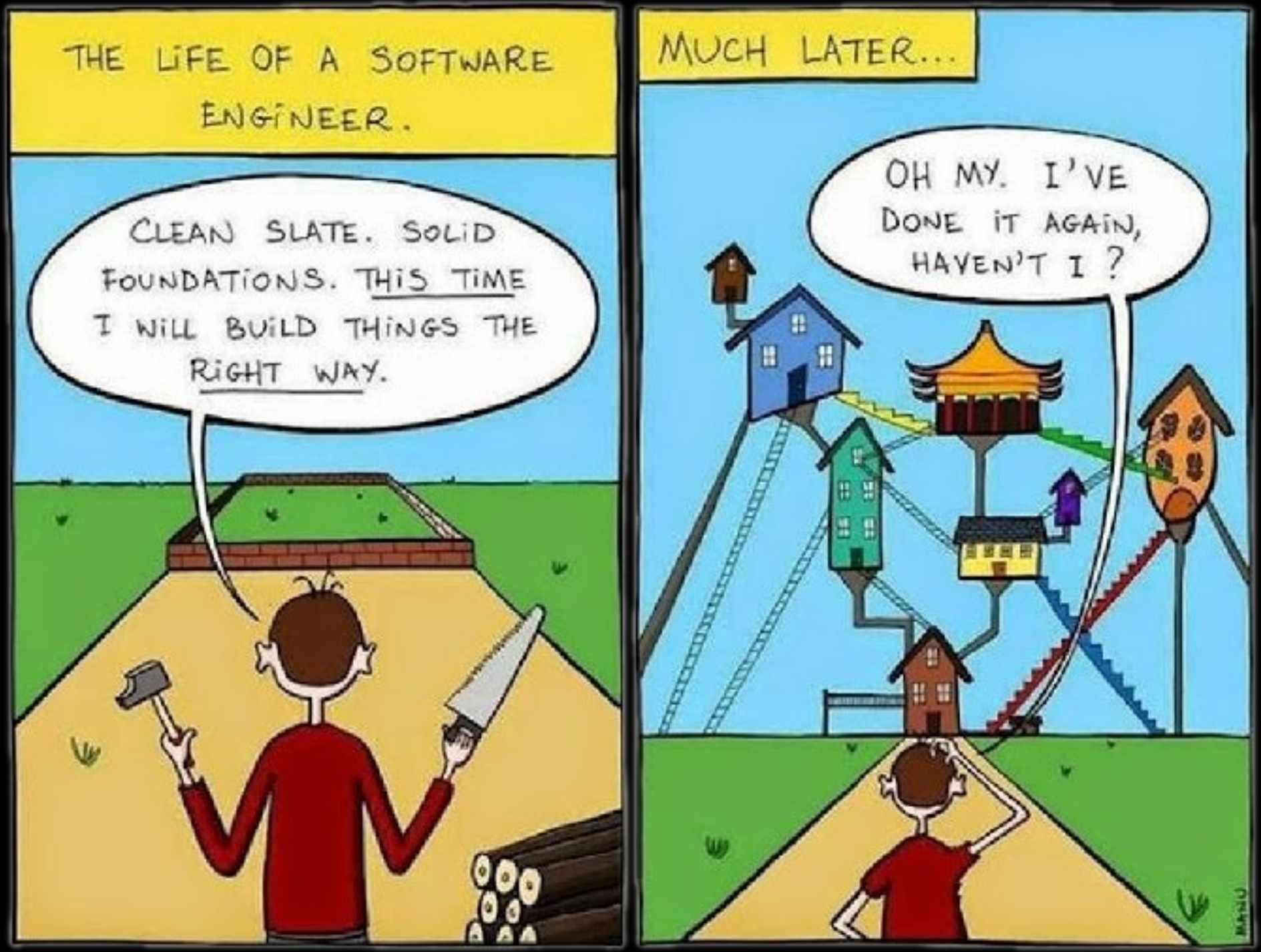
Idiomatic Python

```
# Pythonic code, use context manager ("with") and enumerate:
with open("somefile.txt") as fh:
    for counter, line in enumerate(fh):
        print(counter, " ", line[: -1])

# C++ish code:
fh = None
try:
    fh = open("somefile.txt")
    counter = 0
    line = fh.readline()
    while len(line) > 0:
        print(counter, " ", line[:-1])
        counter += 1
        line = fh.readline()
finally:
    if fh is not None:
        fh.close()
```

See also:
https://docs.python.org/3.0/howto/doanddont.html

Before going into detail:

- OOP is sometimes over-rated. (Java)
- OOP does not solve all your problems.
- Keep it simple.
- Python does not support all OOP concepts. Hooray!


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?

- Next to built-in types (int, list, str, ...), you can define more general "**objects**" with **attributes** and a **behavior**.

Live demo

- Classes can "**inherit**" from other classes, and add & override attributes & methods.

Live demo

"Polymorphism" justifies inheritance.

= Difference in behavior with the same API

Benefits

- Related elements (data and code) are also nearby in source.
- Higher-level programming, in terms of objects
- Polymorphism can reduce many "if" statements.

Limitations

- Methods are essentially unary operators.

Pitfalls

- Too many classes.
- Too complex inheritance diagrams. Use composition where possible.
- Too many methods.

Free functions

- = method “degraded” to a function.
  See https://www.youtube.com/watch?v=nWJHhtmWYcY
- Goal: keep classes simple & easy to understand
- When to write a free function?
  - Attributes are not modified (directly).
  - Algorithms that "work with" objects
  - Binary (or higher) operators.
  - When a class becomes too complicated.

Hands-on: polygon & regular polygon

1. Write polygon class and add features:

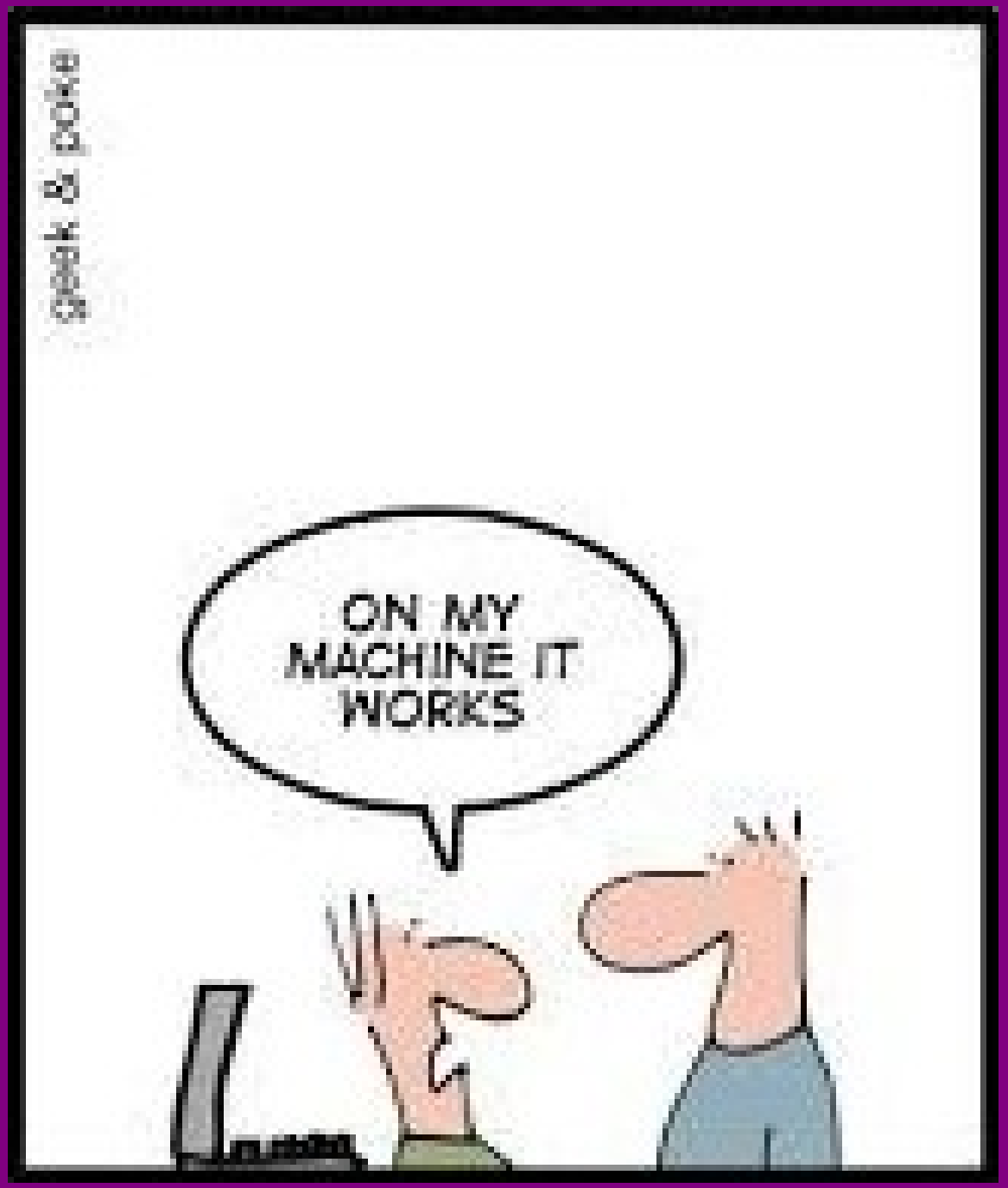
- compute_area and compute_perimeter
- rotate, scale and translate
- regular polygon

Select the “best” patterns: inheritance or composition, method or function.

2. Minimization of perimeter/area ratio

- Implement function for ratio with 1 argument: x & y arrays concatenated.
  Add regularization term, 1e-6*(1−area)**2.
- Implement gradient with autograd.
- Minimize with scipy.optimize.fmin_lbfgs_b.


geek & poke
YESTERDAY IT WORKED


geek & poke
ON MY MACHINE IT WORKS

Even if you don't collaborate...

Long-term maintenance

≈

Collaboration with your future self

More than *avoiding bugs* & *hiding complexity*.

Semantic versioning

https://semver.org/

Given a version number MAJOR.MINOR.PATCH, increment the:

- **MAJOR** version when you make incompatible API changes,
- **MINOR** version when you add functionality in a backwards-compatible manner, and
- **PATCH** version when you make backwards-compatible bug fixes.

Version Control System (VCS)

= records history of all changes in source code

Why?

Collaboration

- Merging: combine changes from different persons.
- Review code before merging.
- When was a bug introduced (bisection)
- Blame people for their ugly code. :)

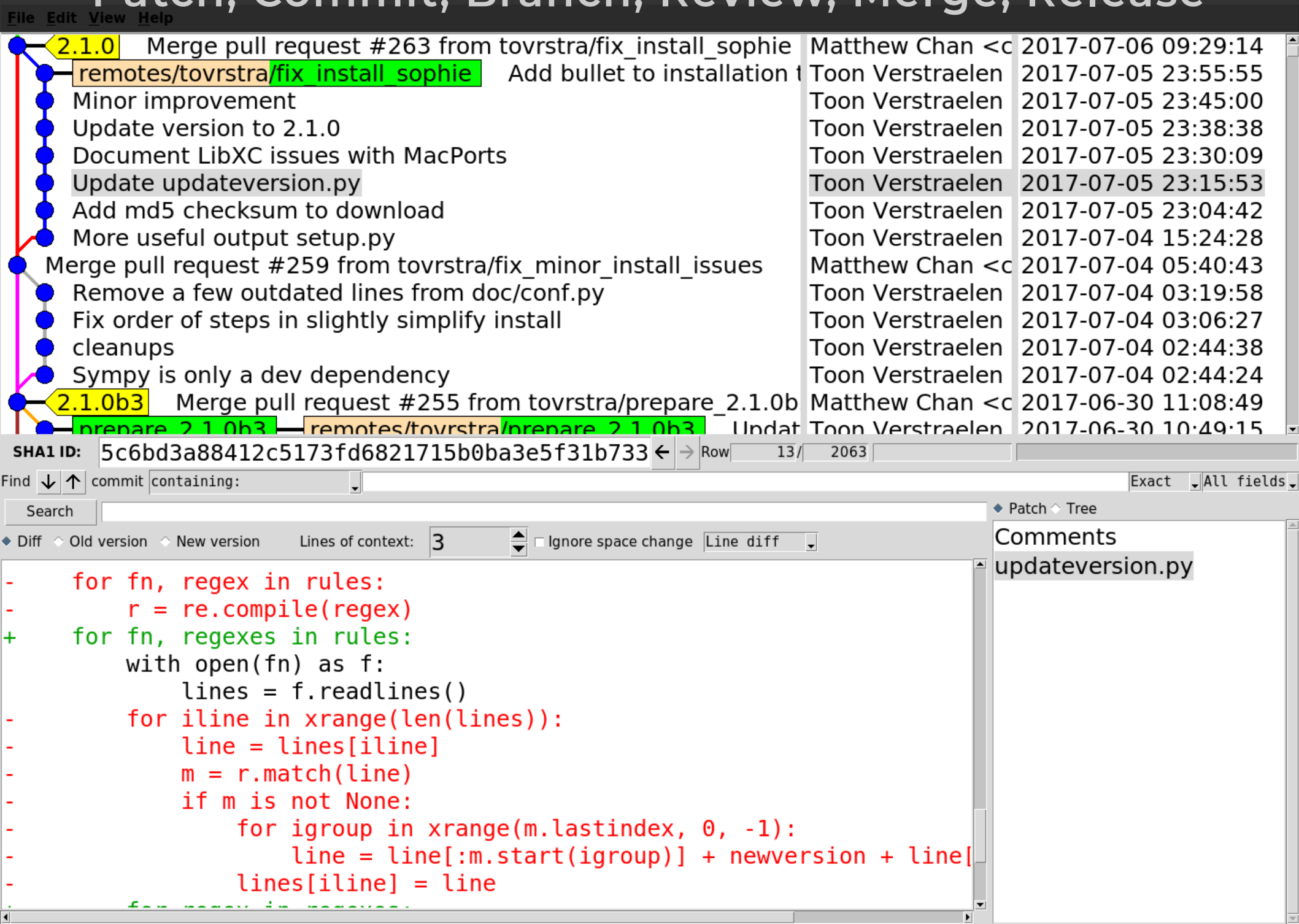
Access to all versions

Backup

A patch (file)

```
diff --git a/horton/grid/cext.pyx b/horton/grid/cext.pyx
index e4615275..47c607fc 100644
--- a/horton/grid/cext.pyx
+++ b/horton/grid/cext.pyx
@@ -55,7 +55,7 @@
     'PowerExtrapolation', 'PotentialExtrapolation', 'tridiagsym_solve',
     'CubicSpline', 'compute_cubic_spline_int_weights',
     # evaluate
-    'index_wrap', 'eval_spline_grid', 'eval_decomposition_grid',
+    'eval_spline_grid', 'eval_decomposition_grid',
     # ode2
     'hermite_overlap2', 'hermite_overlap3', 'hermite_node',
     'hermite_product2', 'build_ode2',
@@ -477,10 +477,6 @@


-def index_wrap(long i, long high):
-    return evaluate.index_wrap(i, high)
-
-
 def eval_spline_grid(CubicSpline spline not None,
                      np.ndarray[double, ndim=1] center not None,
                      np.ndarray[double, ndim=1] output not None,
```

Patch, Commit, Branch, Review, Merge, Release



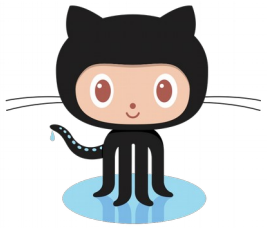
Git & Github

Git = probably the best VCS software

https://git-scm.com/

- Steep learning curve, but worth it.
- Lots of online tutorials.

Github = Git hosting



https://github.com/

- Hosts git repositories
- Extra's: issue tracker, pull requests, web hosting

Continuous integration (CI)

= automatically analyze every commit on Github:

- Unit tests + coverage analysis
- Coding style (pylint, pycodestyle, ...)
- Test package build & install
- ...

Very neat, involved setup.

https://travis-ci.org/

Example: https://github.com/theochem/iodata/pull/44

Write user documentation

README.md

- Links to other documentation
- Quick install instructions
- Contact & License information

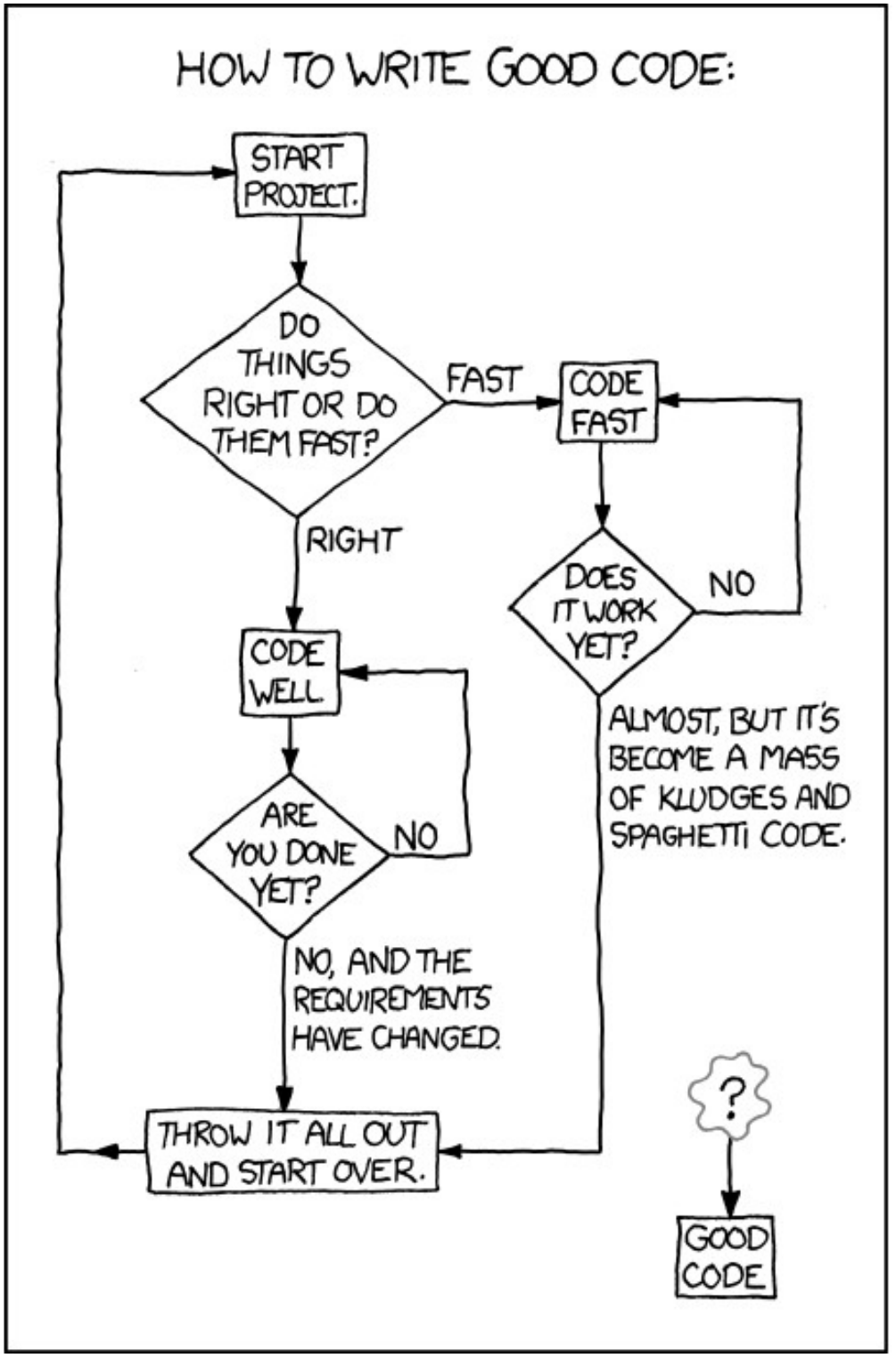
Website (use Sphinx; http://www.sphinx-doc.org)

- Background
- Tutorials
- API reference

Hands-on: fix a simple bug

Fix Scipy documentation:

https://github.com/scipy/scipy/issues/7168



SCRUM

Keep it simple.