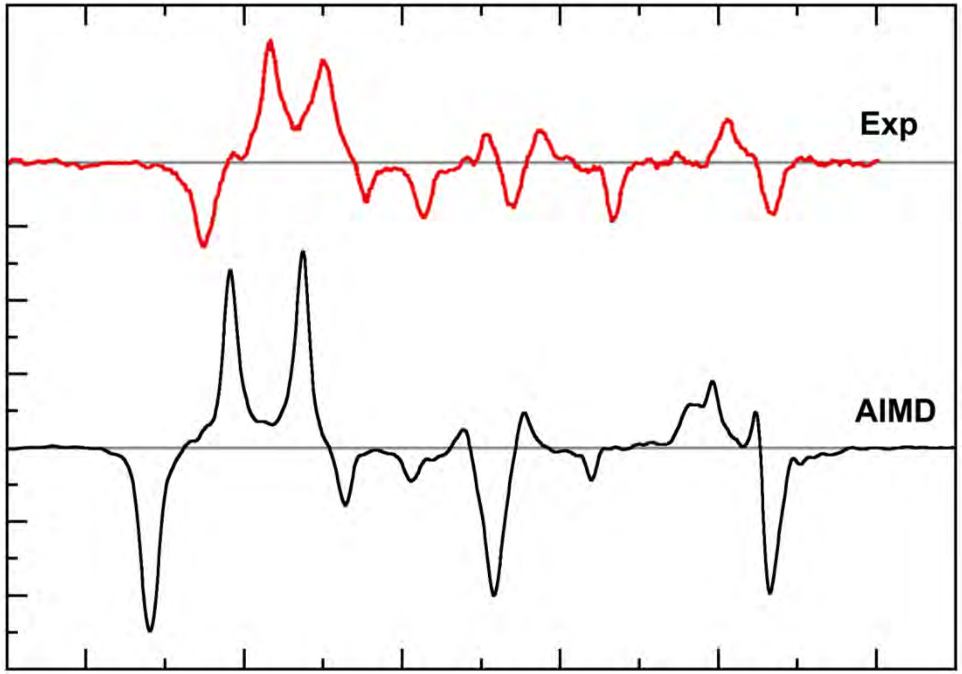
Predicting Vibrational Spectra of Periodic Bulk Phase Systems



Martin Brehm
Martin-Luther-Universität Halle–Wittenberg
Martin_Brehm@gmx.de

Outline

1.) Short Introduction to TRAVIS

2.) Computing Vibrational Spectra

3.) Compression of Volumetric Data

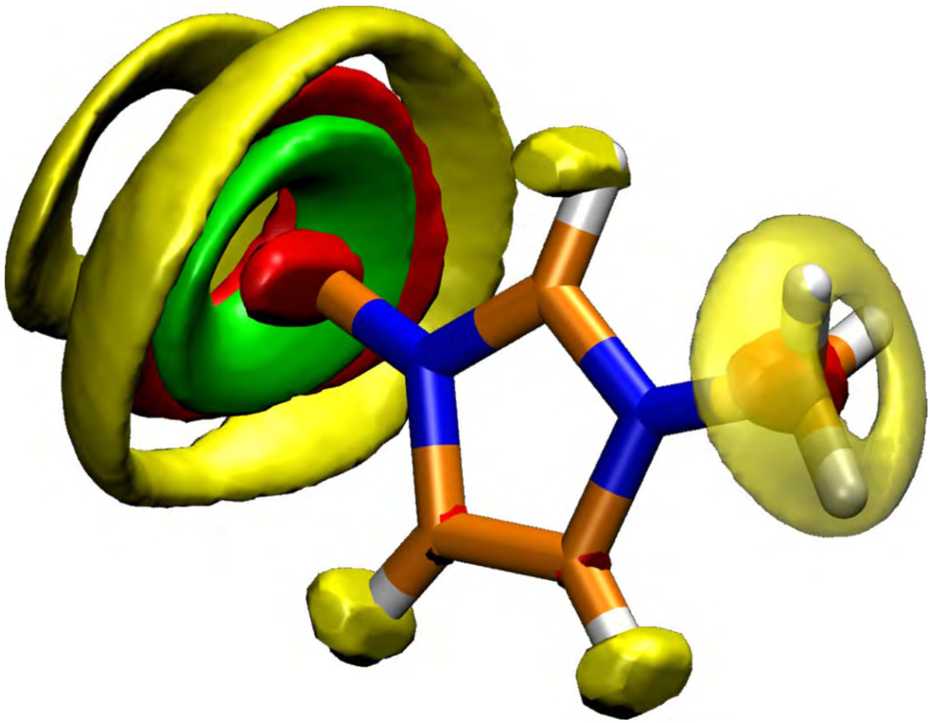
4.) Practical Workflow for Spectra

5.) What we will do in the Exercise

TRAVIS

A free Analyzer and Visualizer

for MC and MD Trajectories



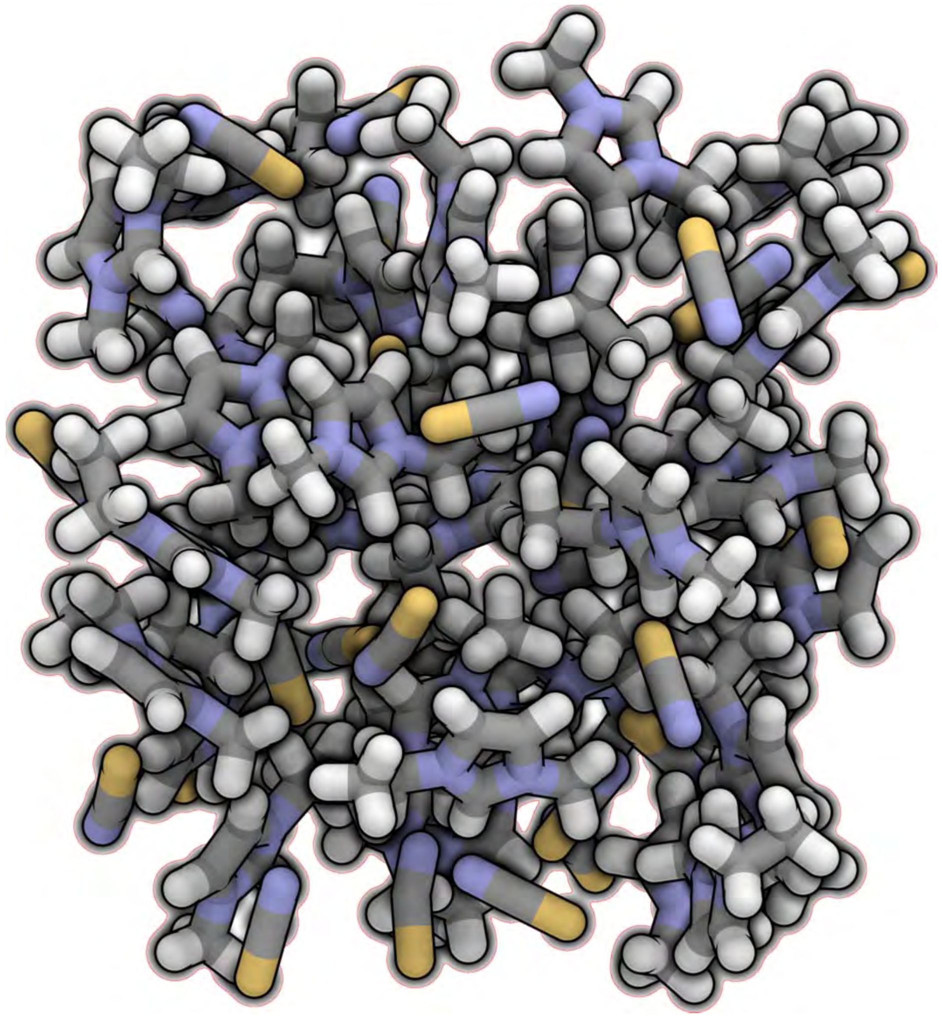
http://www.travis-analyzer.de

Analyzing Trajectories

- Direct result of all MD/MC simulations is a trajectory
- Contains positions and velocities of all atoms at each time

→ is a path through *6N*-dimensional space

„Nice to look at, but cannot be evaluated directly."



Mappings for the reduction of dimensionality are required.

Introducing TRAVIS

- Program package
  for doing these analyses

- Open-source freeware;
  licensed under GNU GPL 3

- ≈ 220 000 lines of C++ code

- Platform independent (Windows / Linux / Mac)

- Published in 2011, cited more than 230 times since then:

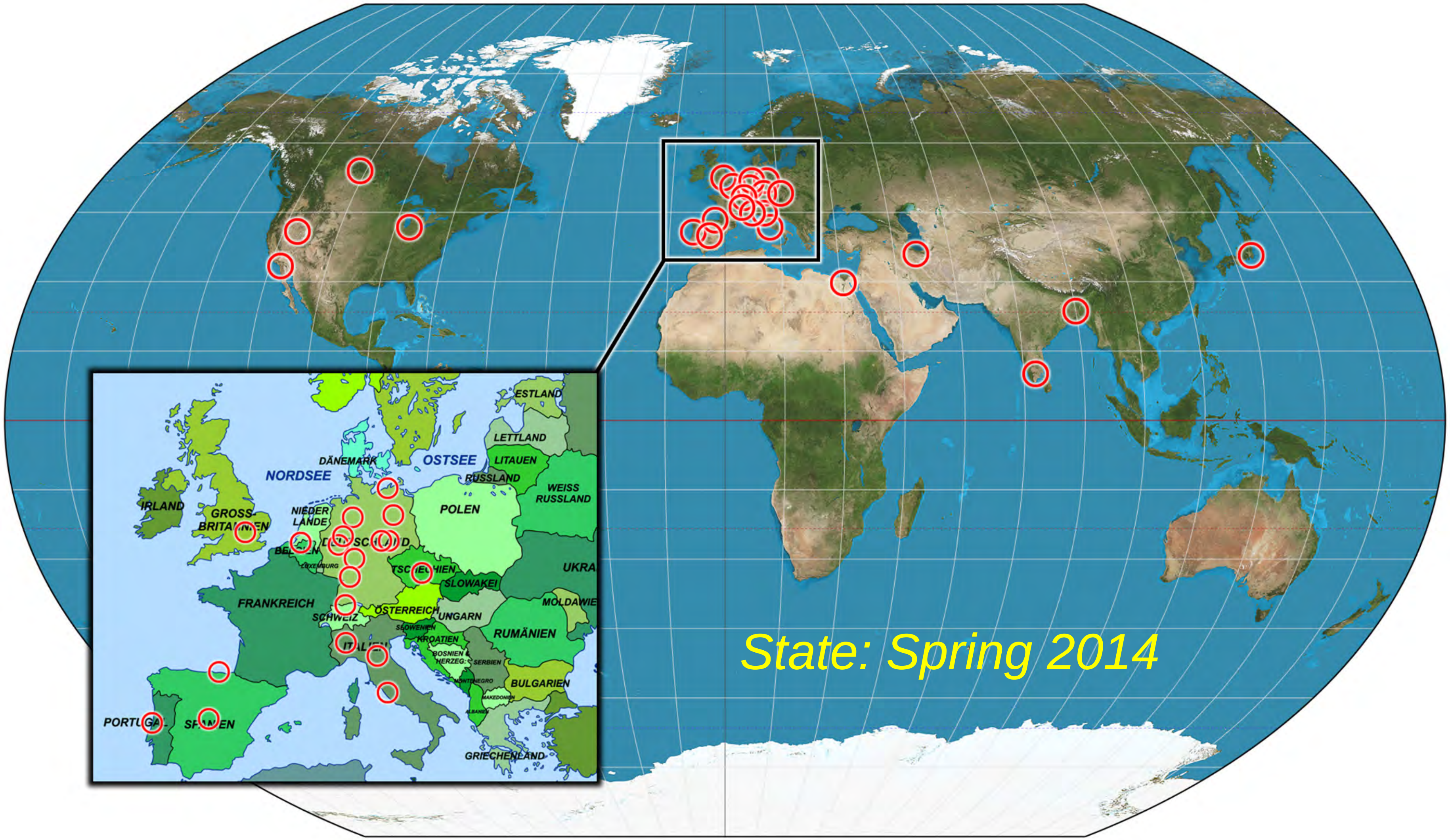
  Martin Brehm and Barbara Kirchner: „TRAVIS - A Free Analyzer and Visualizer for Monte Carlo and Molecular Dynamics Trajectories“
  J. Chem. Inf. Model. **2011**, *51 (8)*, pp 2007–2023 .

http://www.travis-analyzer.de

Introducing TRAVIS

Several dozen working groups around the world use TRAVIS

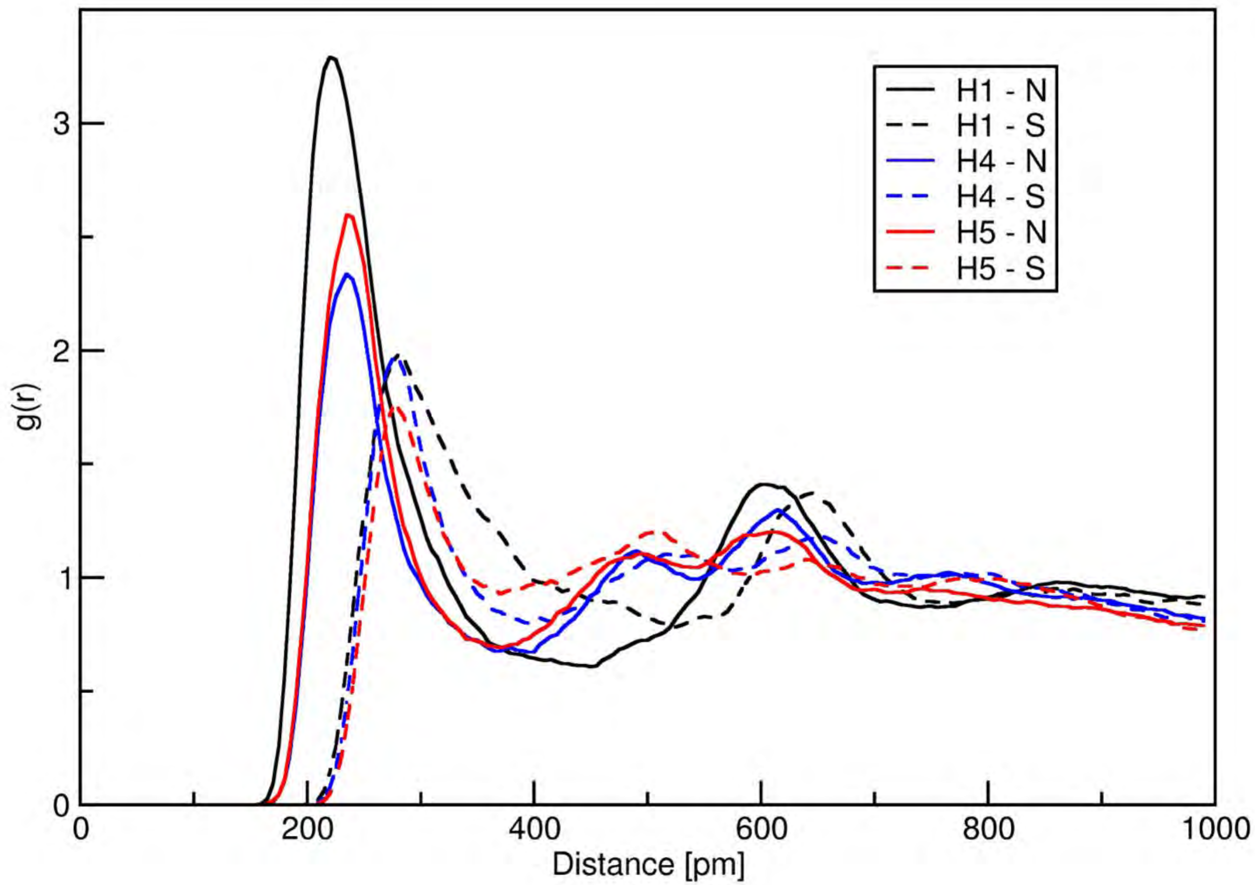
(I only know of the groups which had problems ☺)



General Features

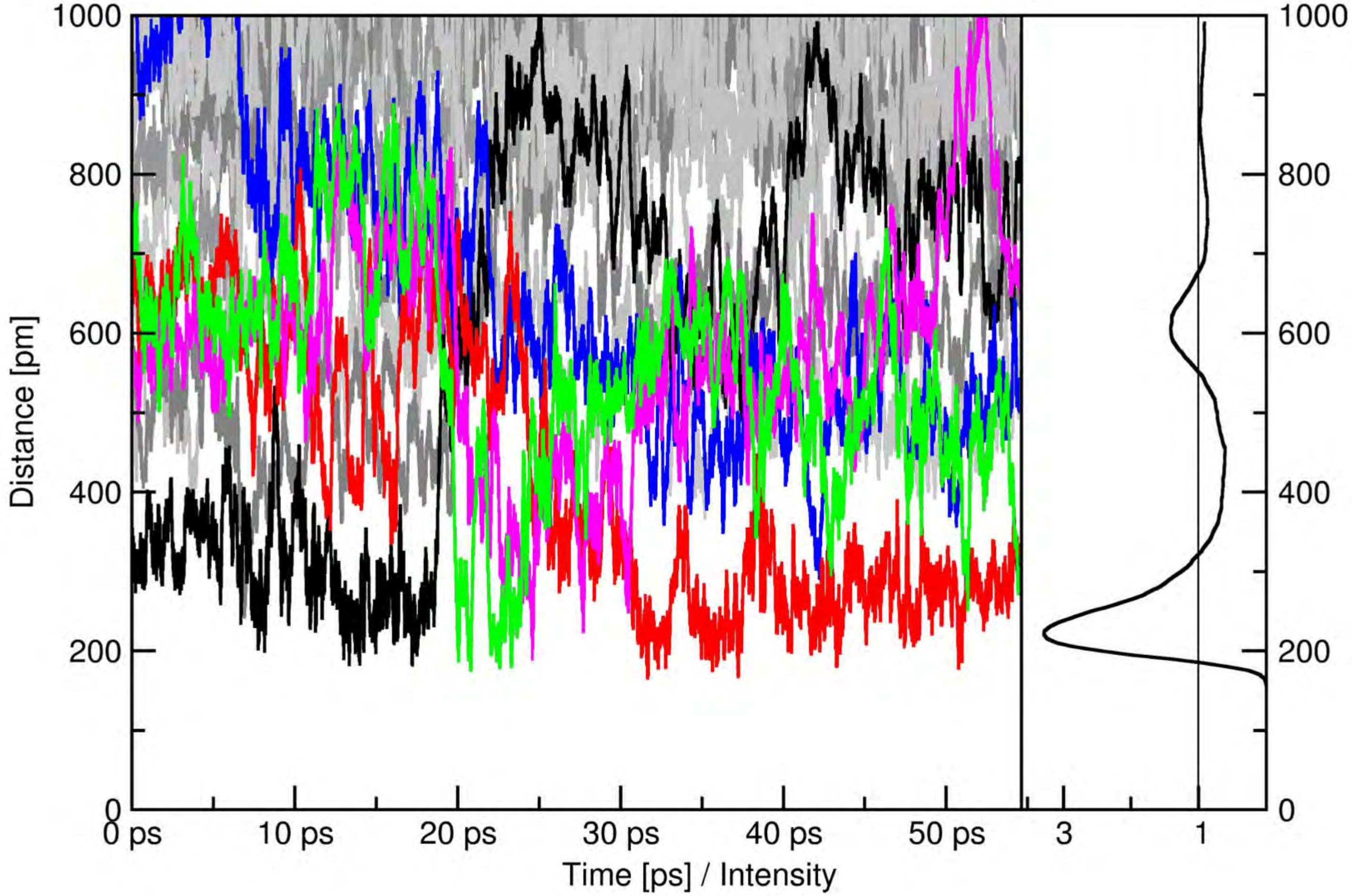
- Interactive text mode user interface (asks questions), but also scripting support
- Reads many popular trajectory file formats (xyz, pdb, mol2, AMBER, LAMMPS, DLPOLY)
- No limits on system size (works well with > 10^5 atoms)
- Support for periodic boundaries and changing cell vector (*e.g.*, from NPT simulations)
- Automatic molecule recognition (recognizes also molecules that are broken by wrapping)
- Atom labels based on purely topological algorithm

Structural Analyses



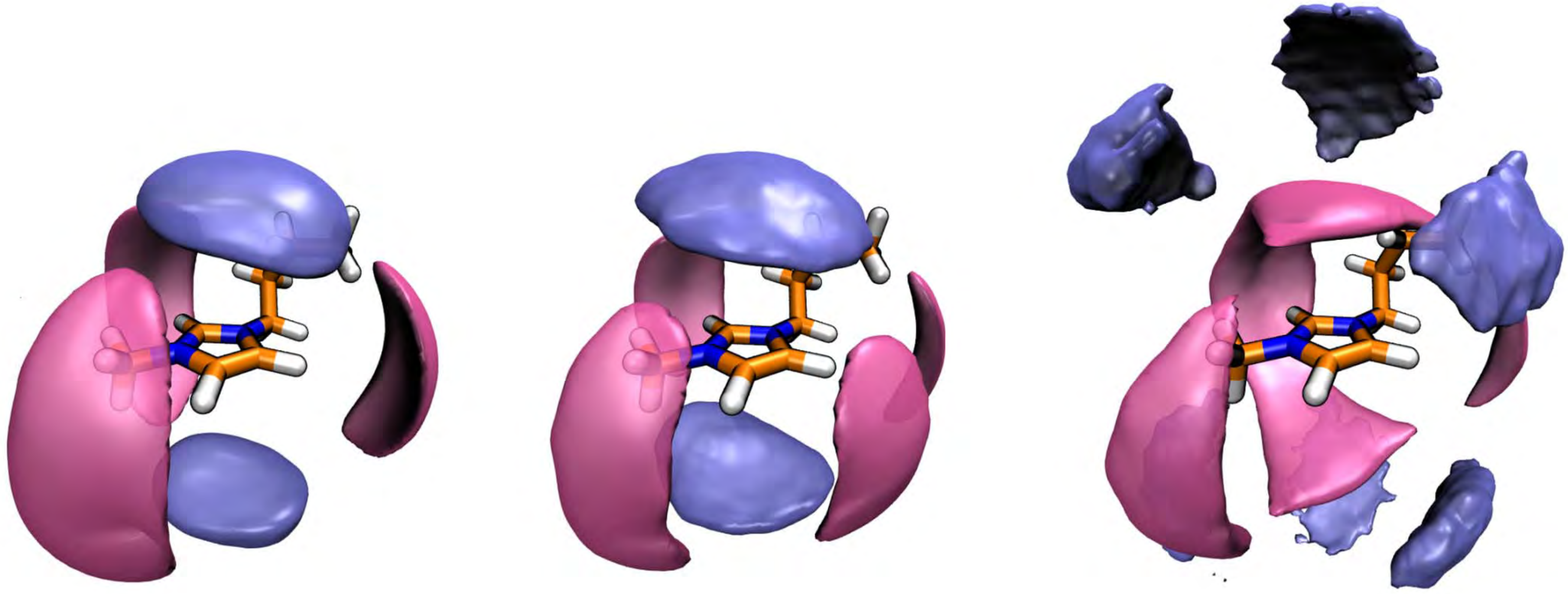
Radial Pair Distribution Functions

Structural Analyses



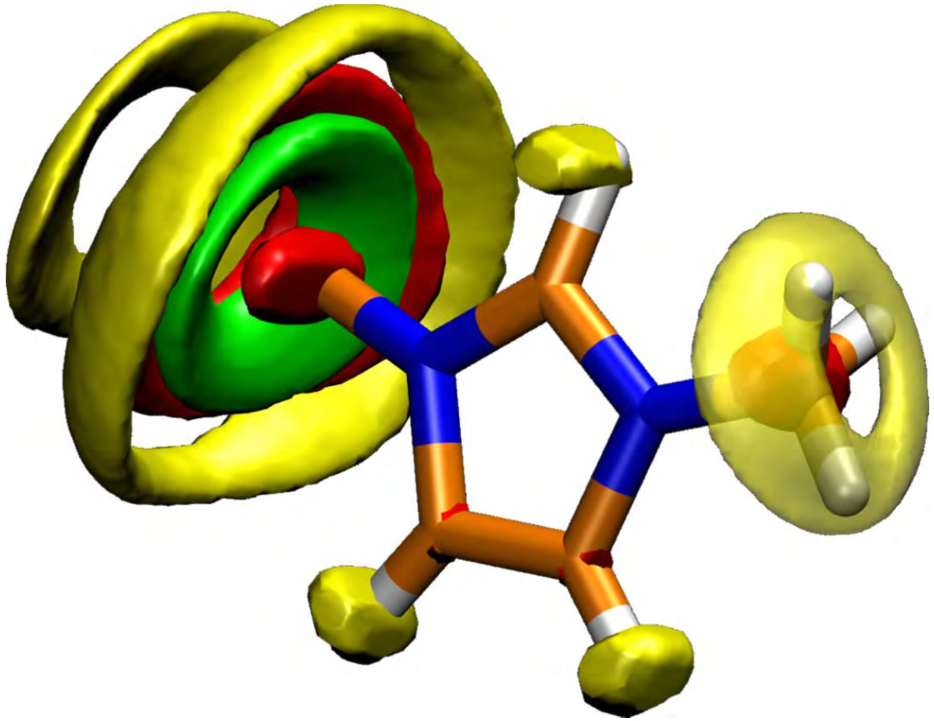
Temporal Distance Development and distribution

Structural Analyses



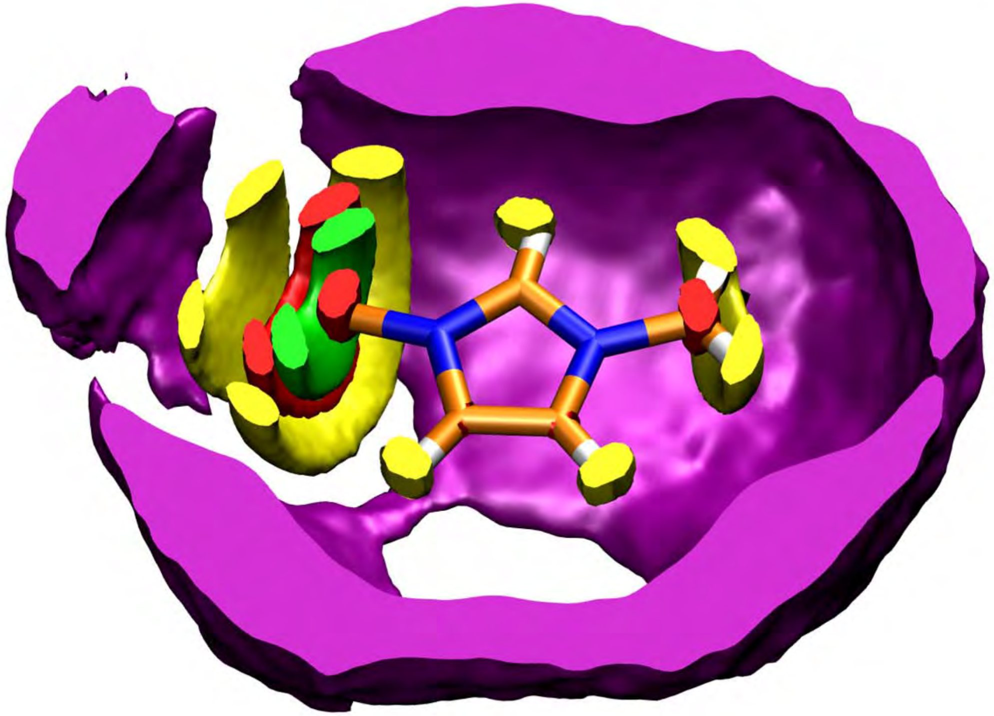
Spatial Distribution Functions

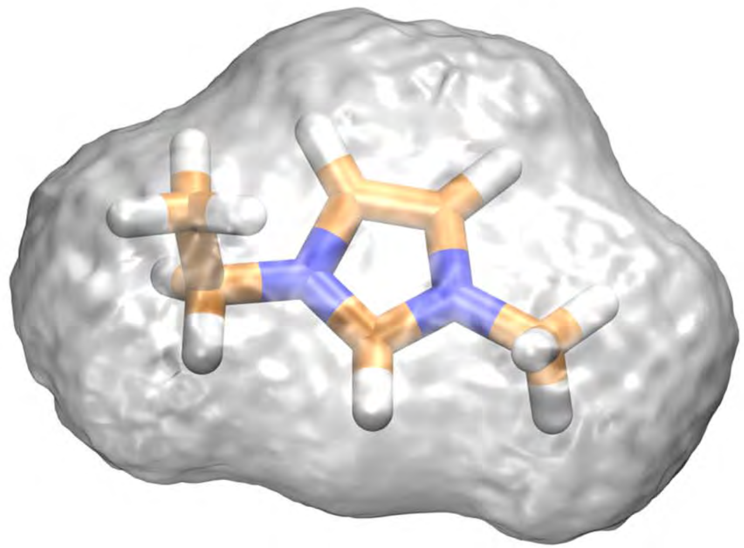
Structural Analyses



Spatial Distribution Functions

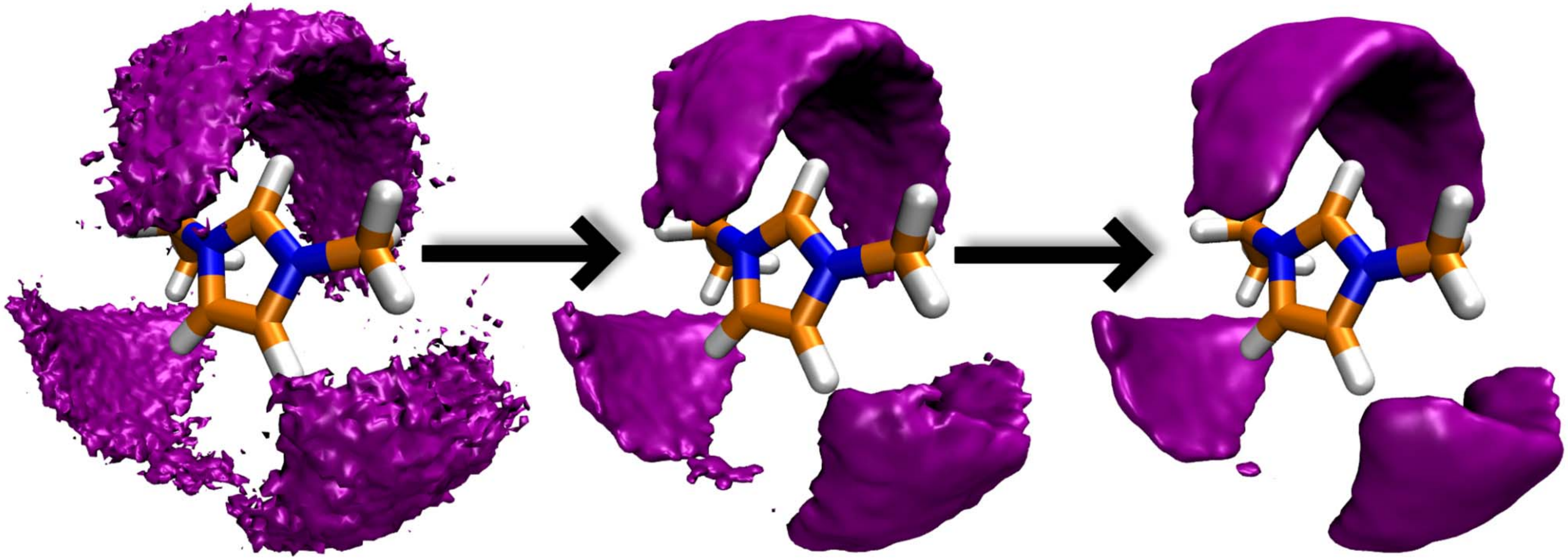
Structural Analyses





Spatial Distribution Functions

Structural Analyses

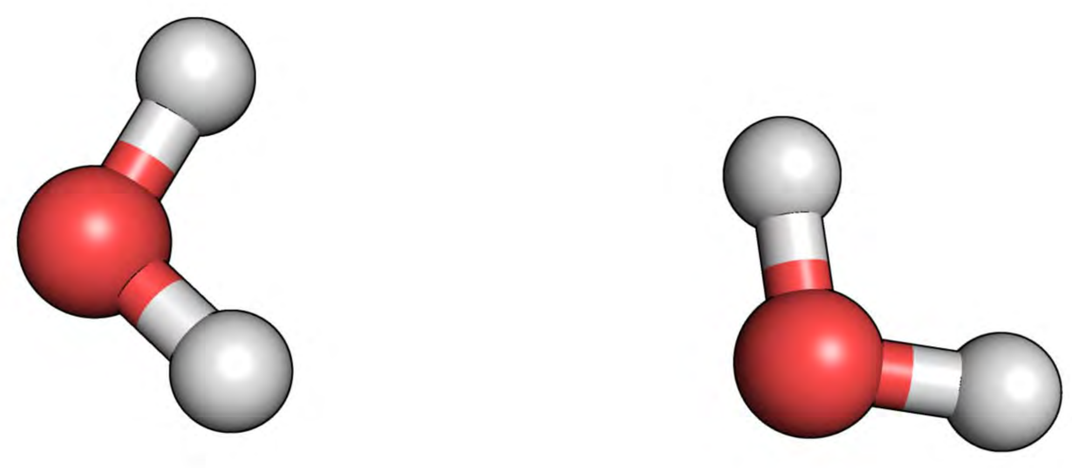


Smoothing of Spatial Distribution Functions

Combined Distribution Functions

- One example for a new feature that did not appear in literature before

- Consider these 2 water molecules

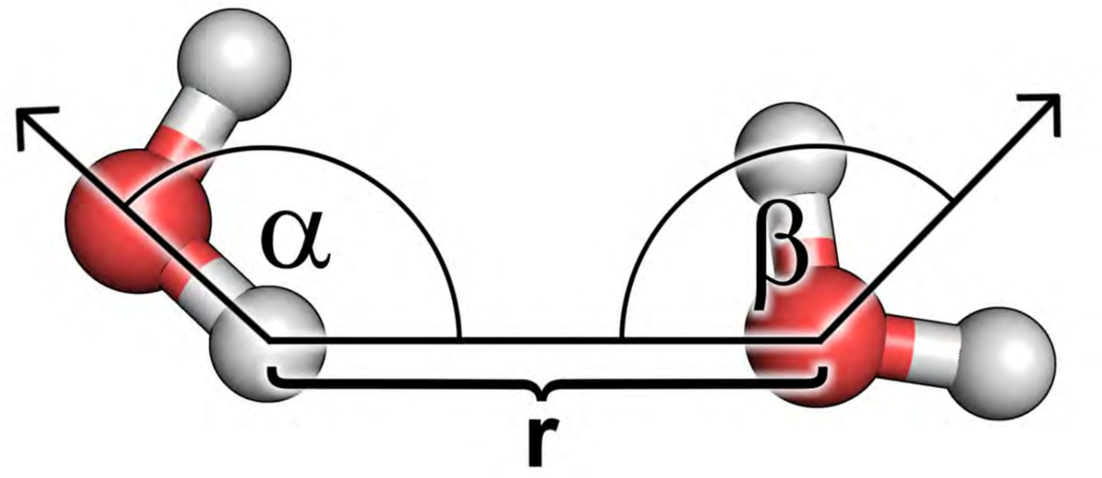


- Define a distance and two angles

Combined Distribution Functions

- One example for a new feature that did not appear in literature before

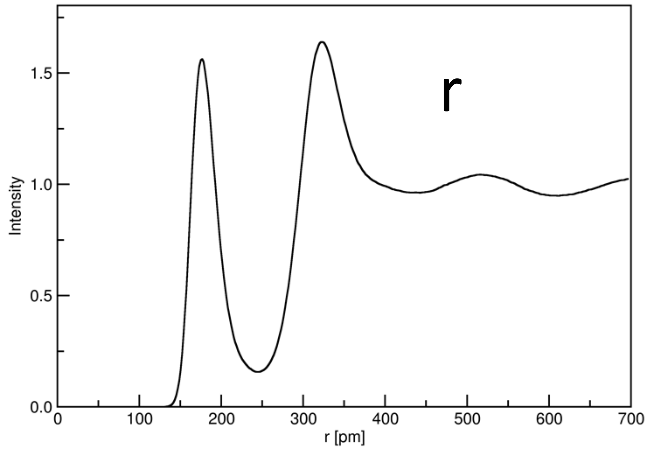
- Consider these 2 water molecules

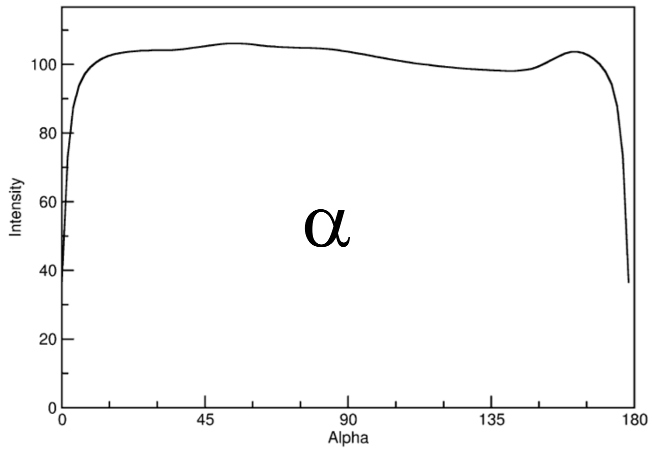


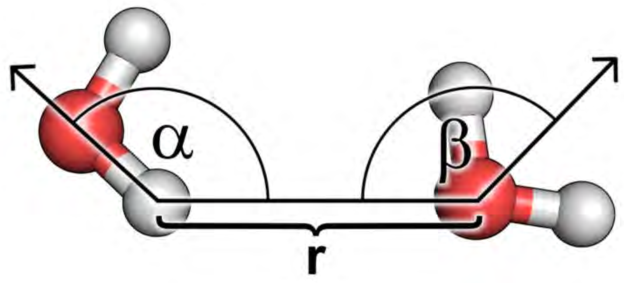
- Define a distance and two angles

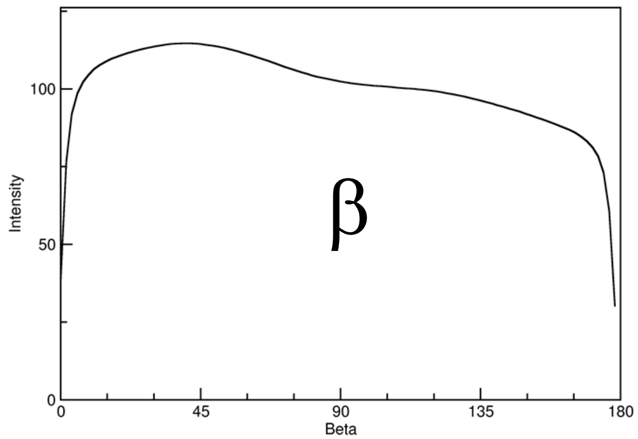
Combined Distribution Functions

- Plot distribution functions for these 3 quantities









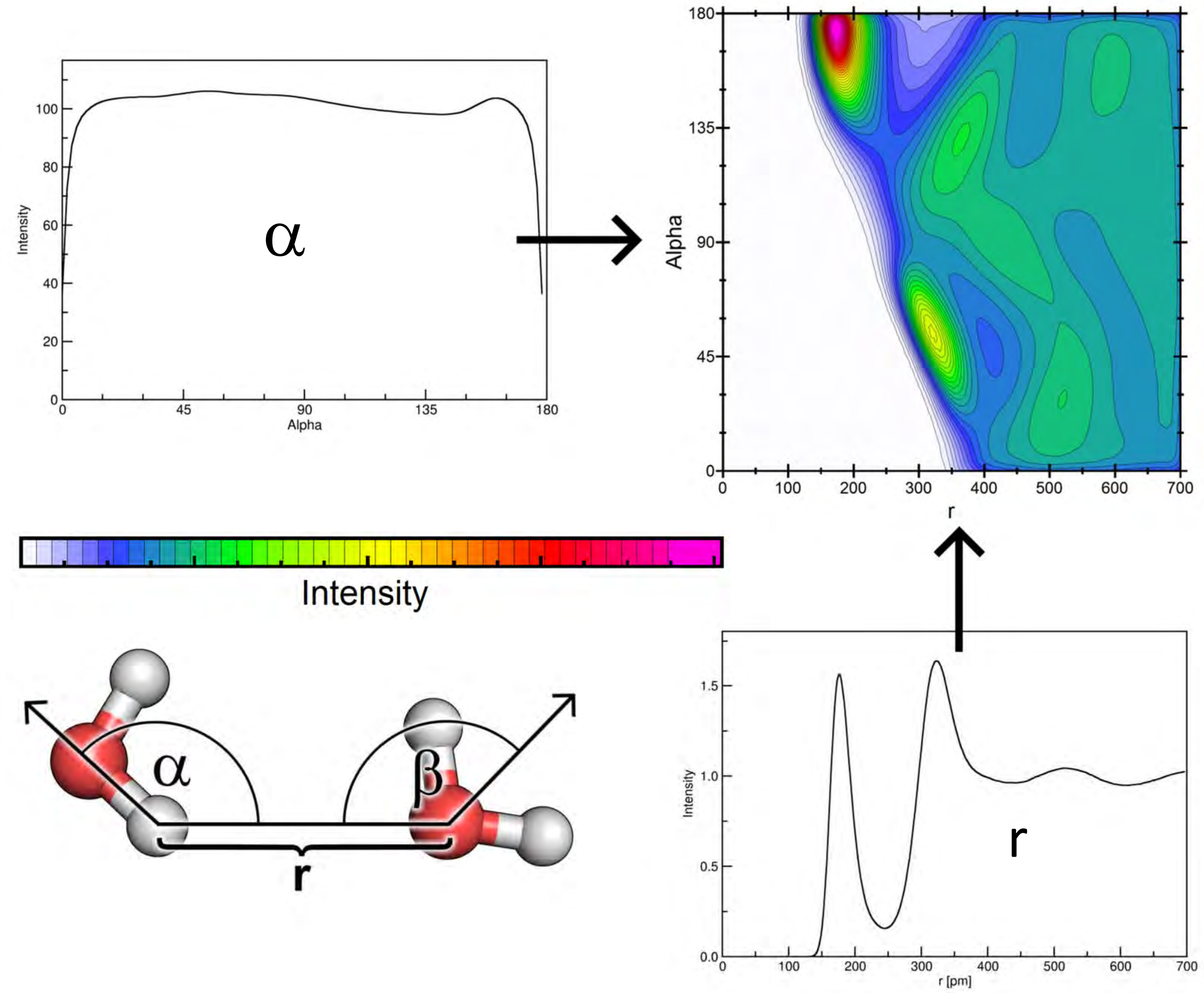
Combined Distribution Functions

- So far nothing new

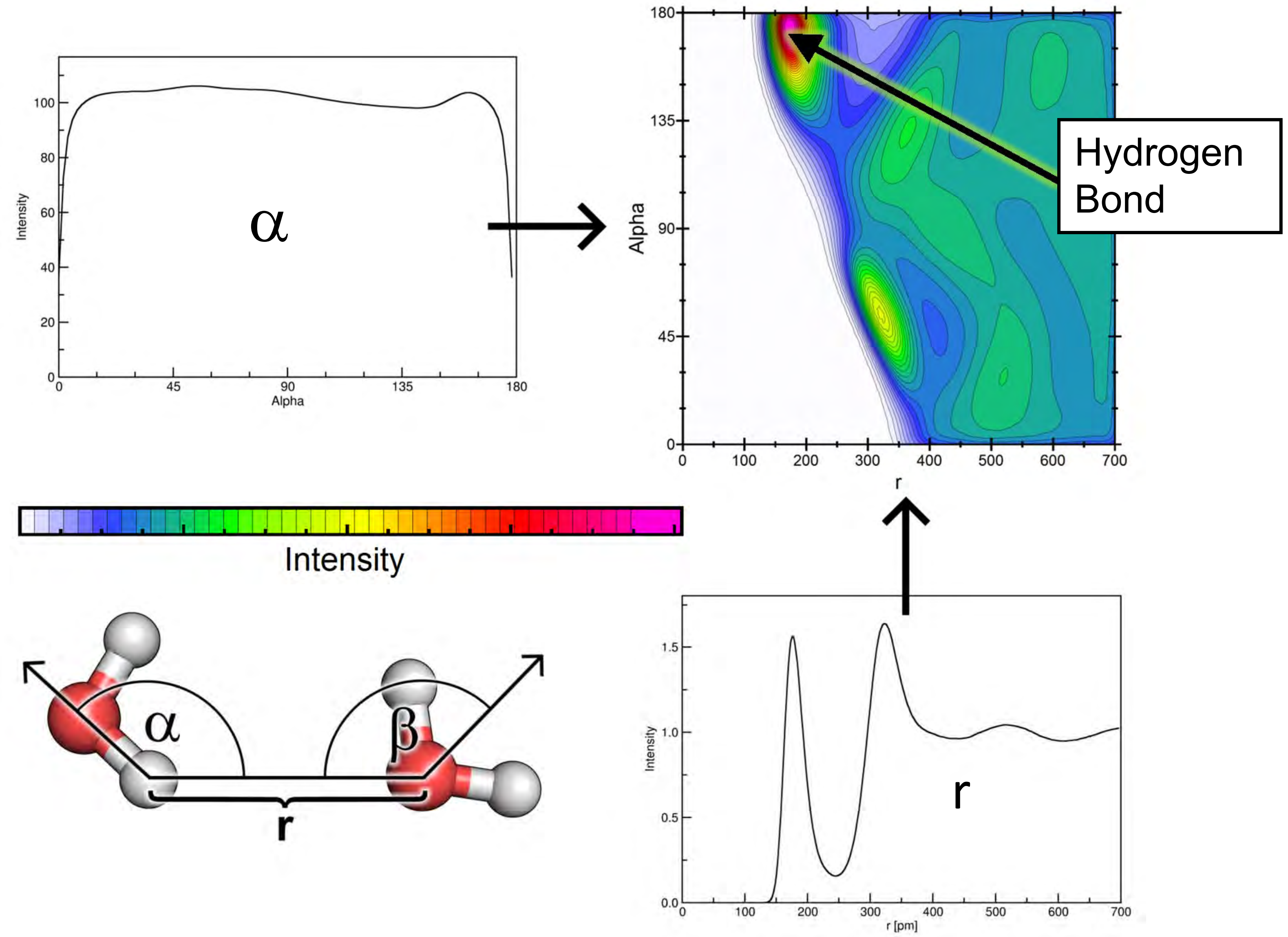
- Dependence of these quantities on each other
  is left out (but very important)

- Idea: Combine certain scalar quantities to yield
  Combined Distribution Functions (CDFs)

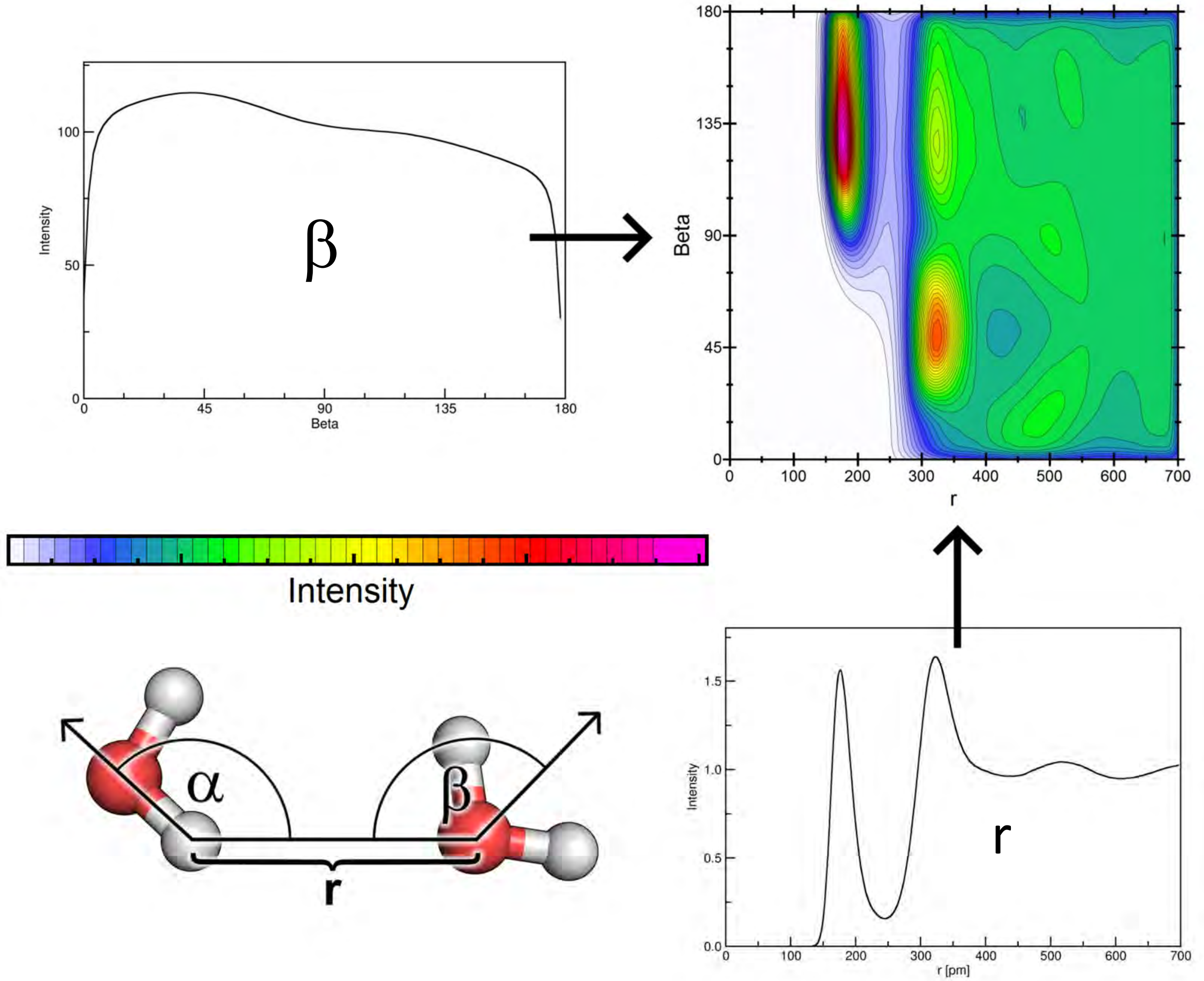
Combined Distribution Functions



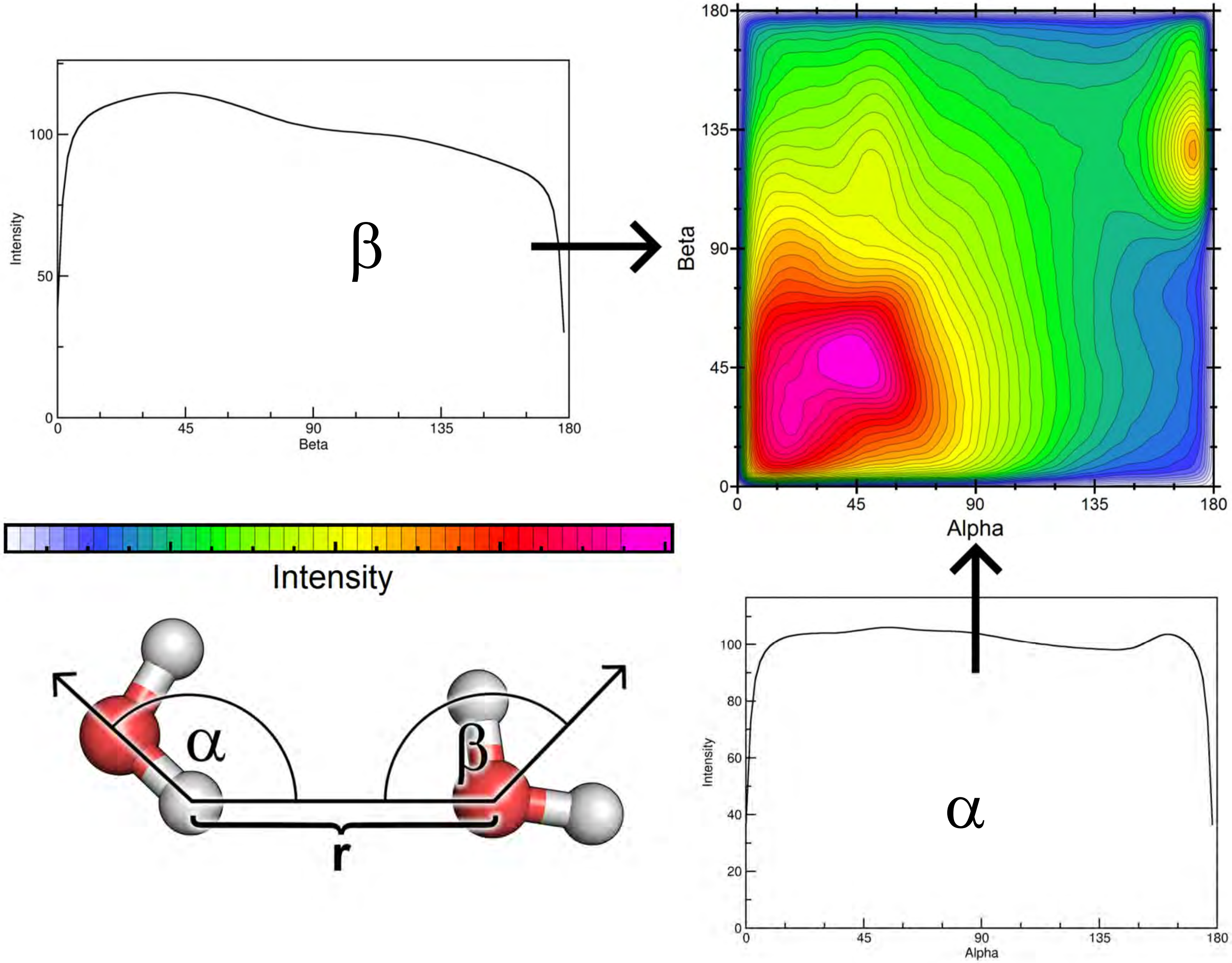
Combined Distribution Functions



Combined Distribution Functions



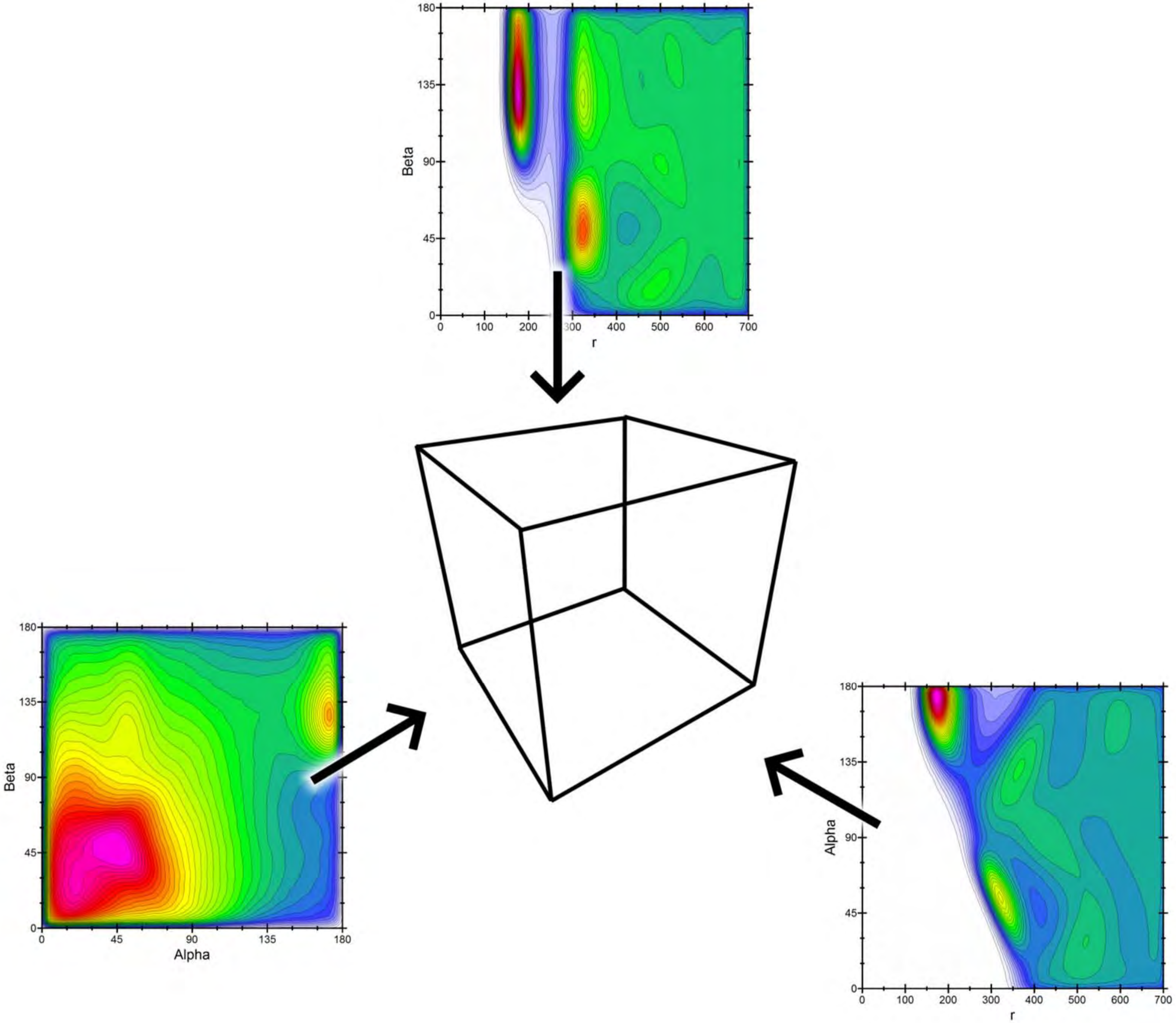
Combined Distribution Functions



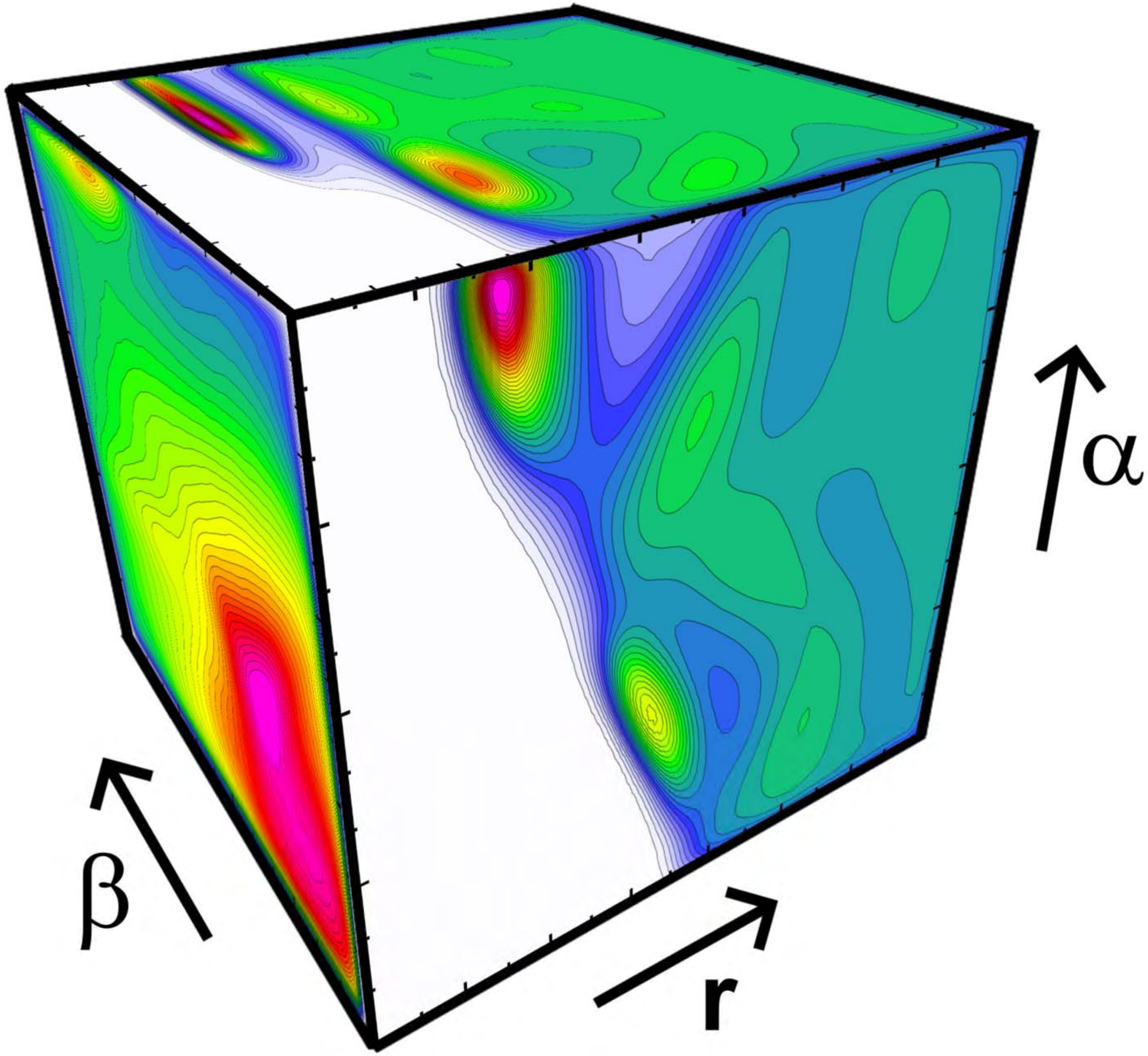
Combined Distribution Functions

- Now we have a 2D distribution
- Much more information can be read out
- What about higher-dimensional histograms? ☺

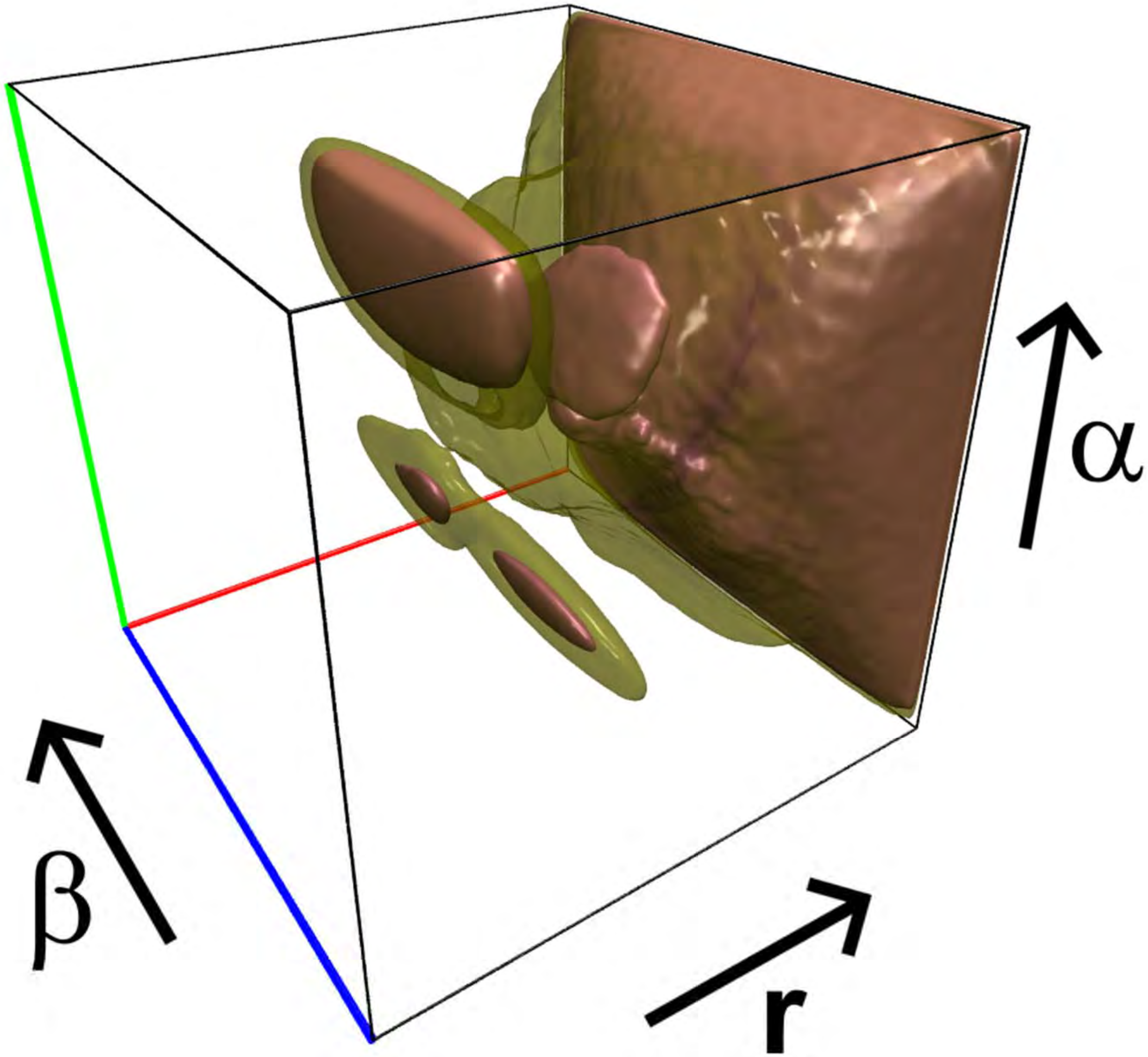
Combined Distribution Functions



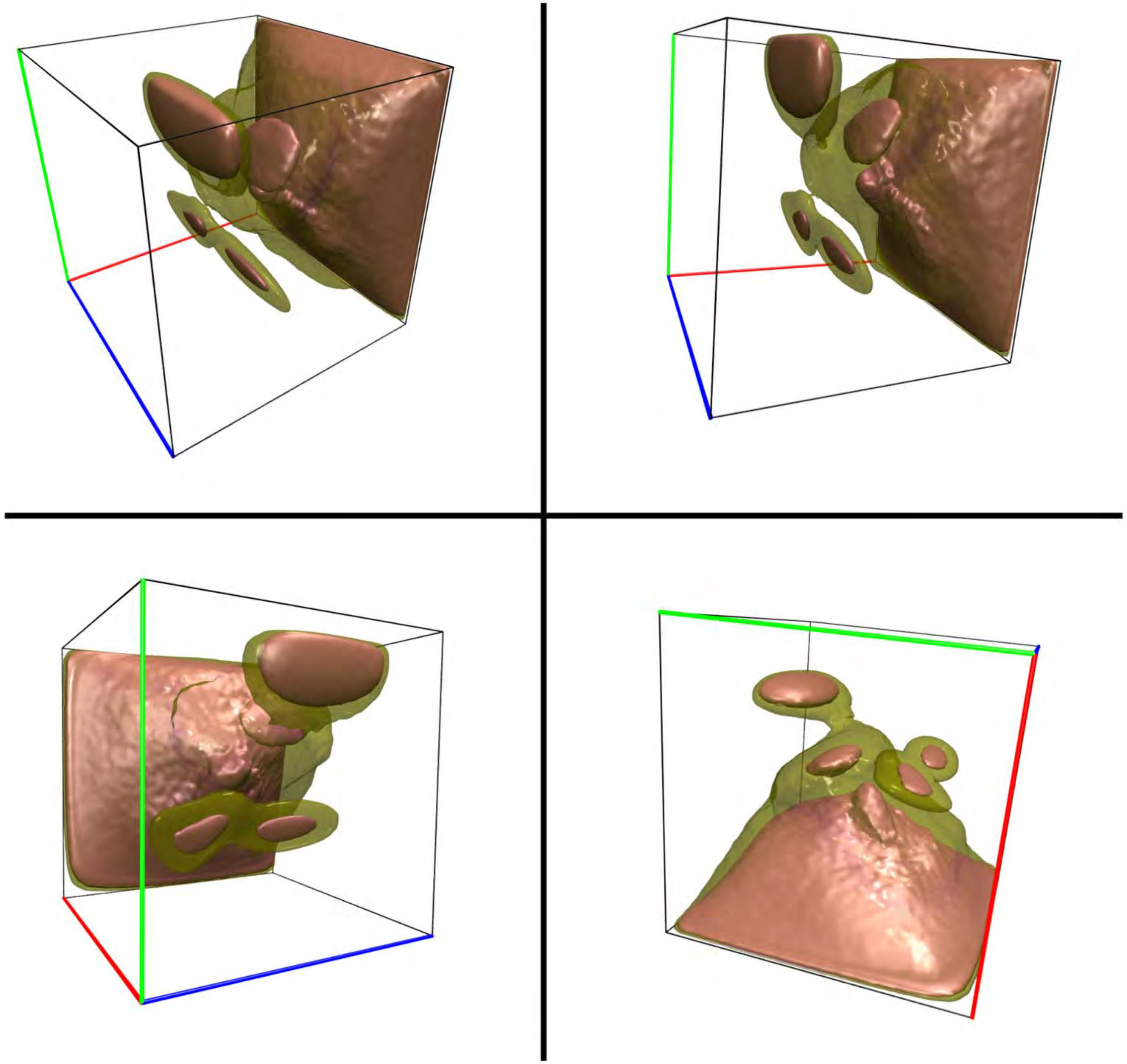
Combined Distribution Functions



Combined Distribution Functions



Combined Distribution Functions



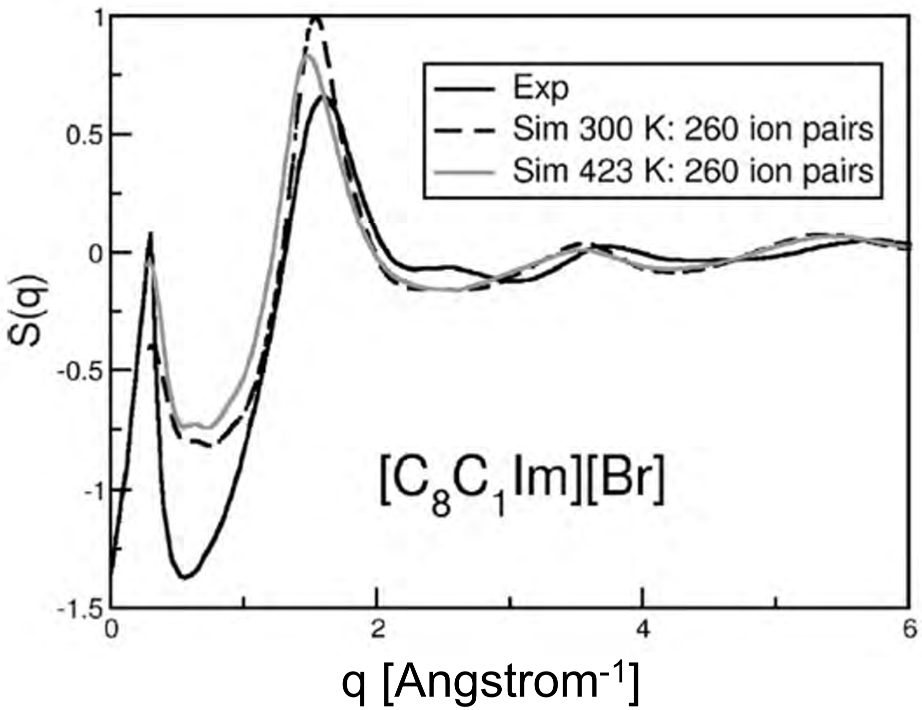
Combined Distribution Functions

- What can be combined?
  - Any distance between two atoms in the system
  - Any angle between three atoms (or two vectors)
  - Any dihedral angle (between 4 atoms or 3 vectors)
  - Absolute velocity of atoms
  - Velocity / force vectors
  - Dipole moments / vectors of molecules

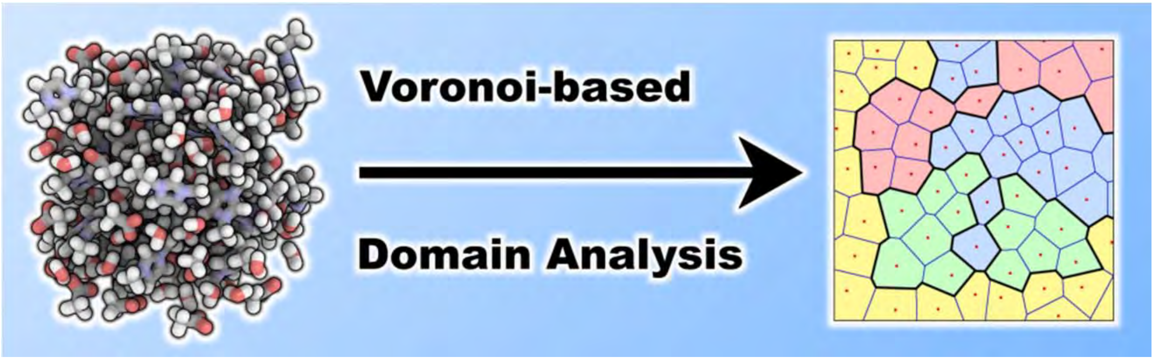
- Combinations can be of any dimensionality
  (shown here only 2D and 3D)

This gives trillions of different combinations!

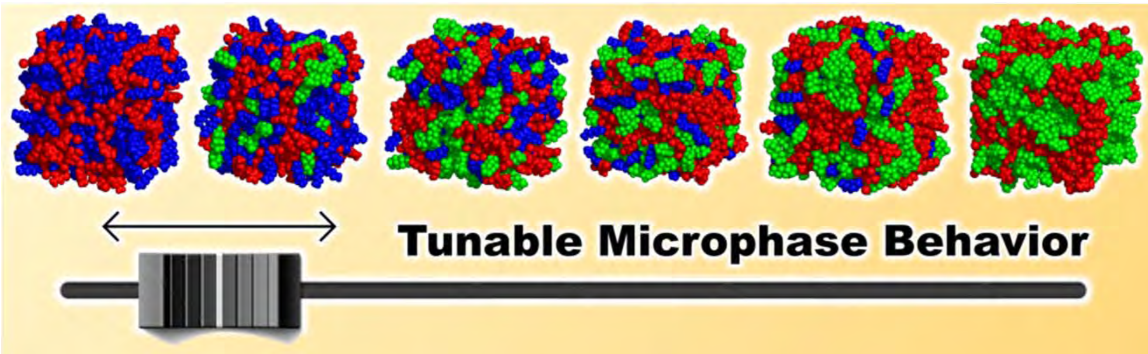
Structure Factors (Neutron / X ray)



Voronoi-based Domain Analysis

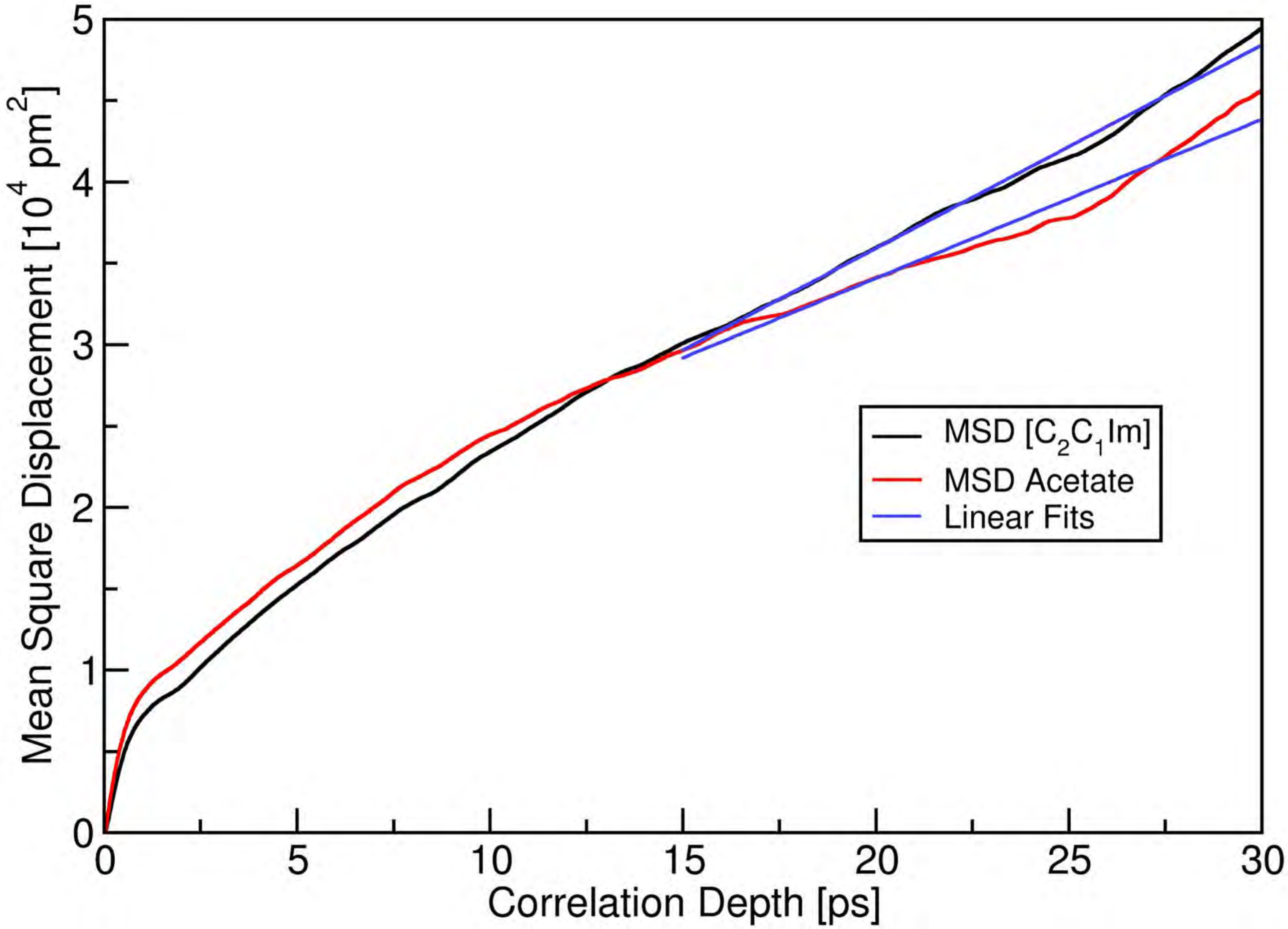


M. Brehm, H. Weber, M. Thomas, O. Holloczki, B. Kirchner: "Domain Analysis in Nanostructured Liquids: A Post-Molecular Dynamics Study at the Example of Ionic Liquids", *ChemPhysChem* **2015**, *16*, pp 3271-3277.



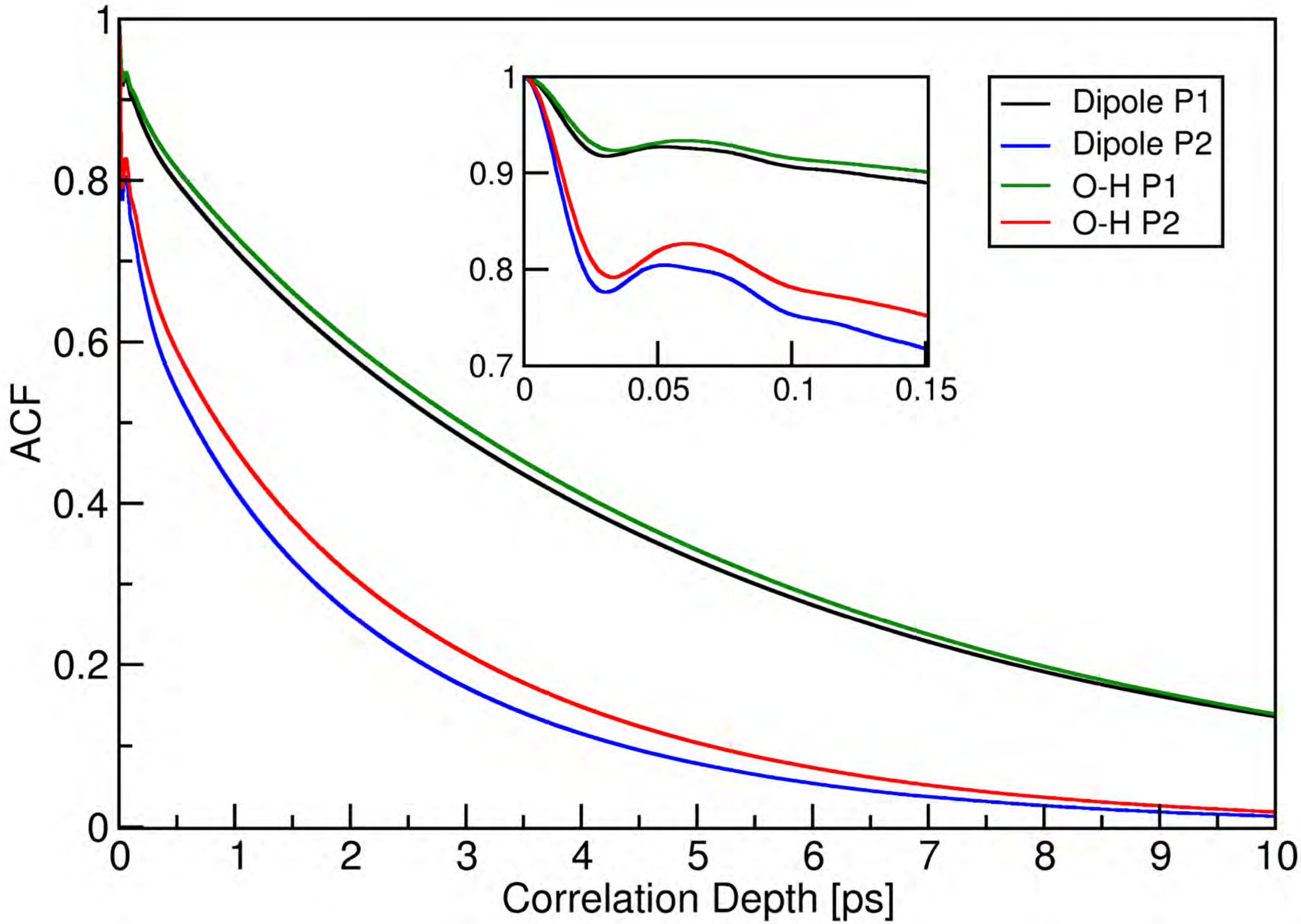
O. Holloczki, M. Macchiagodena, H. Weber, M. Thomas, M. Brehm, A. Stark, O. Russina, A. Triolo, B. Kirchner: "Triphilic Ionic-Liquid Mixtures: Fluorinated and Non-fluorinated Aprotic Ionic-Liquid Mixtures", *ChemPhysChem* **2015**, *16*, pp 3325-3333.

Dynamical Analyses



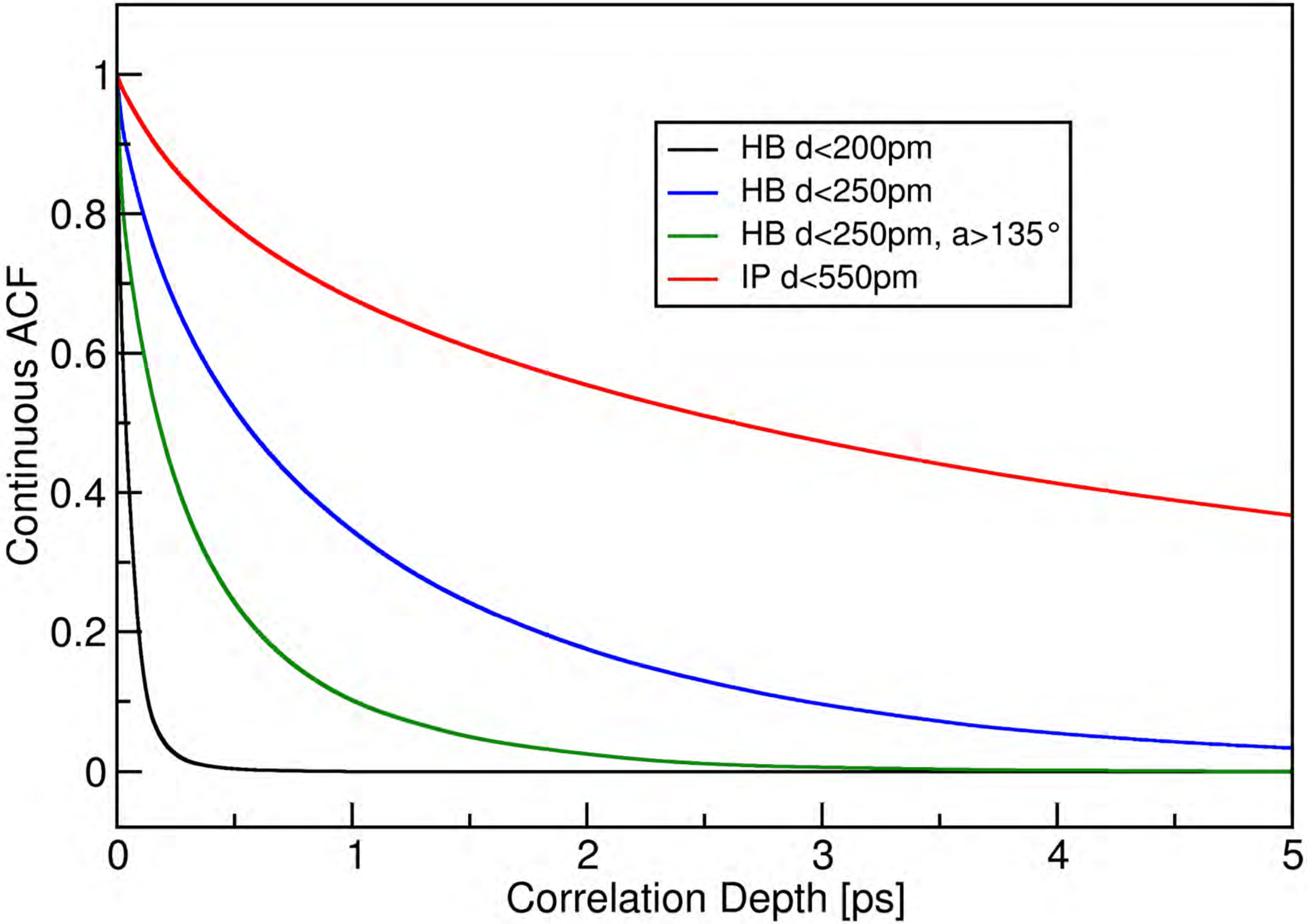
Mean Square Displacement & Diffusion Coefficients

Dynamical Analyses



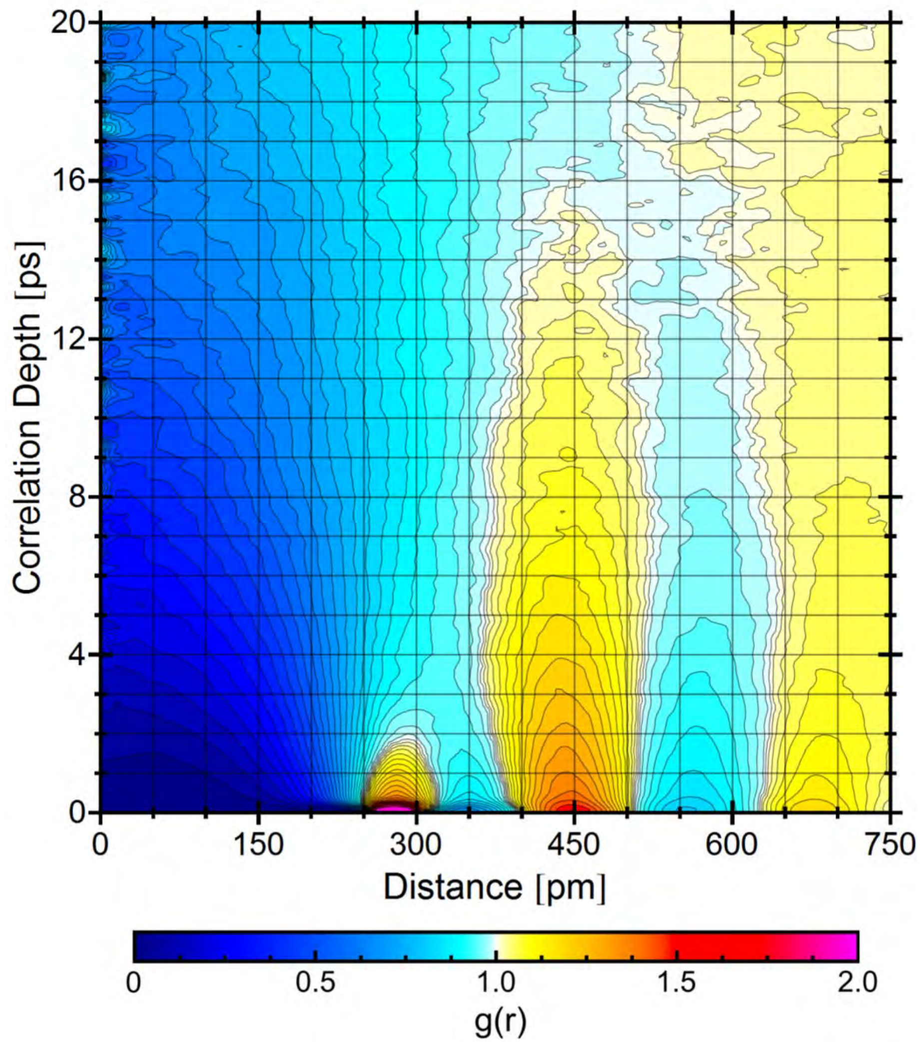
Vector Reorientation Dynamics

Dynamical Analyses



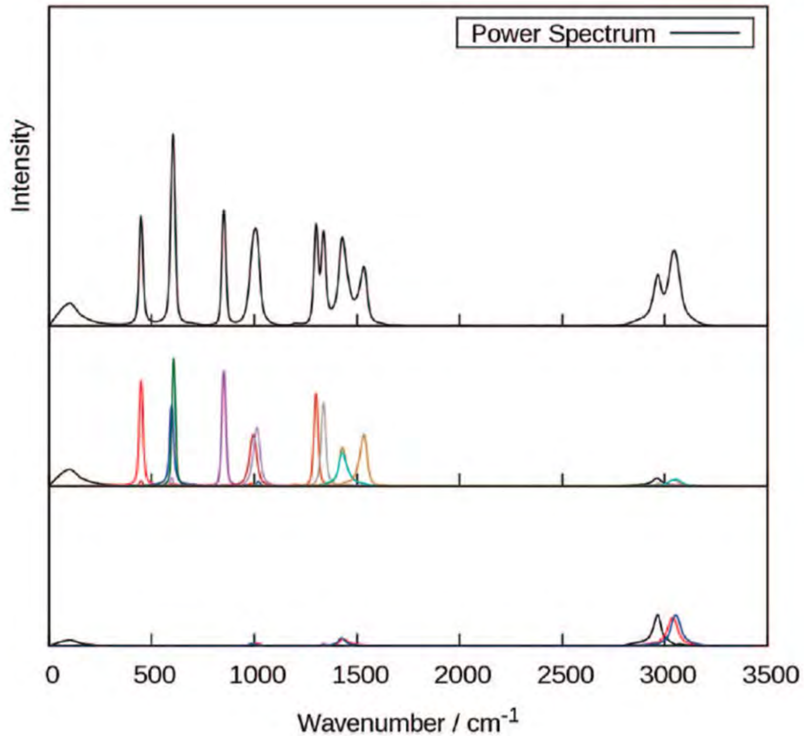
Lifetime of Aggregates (e.g., hydrogen bonds)

Dynamical Analyses

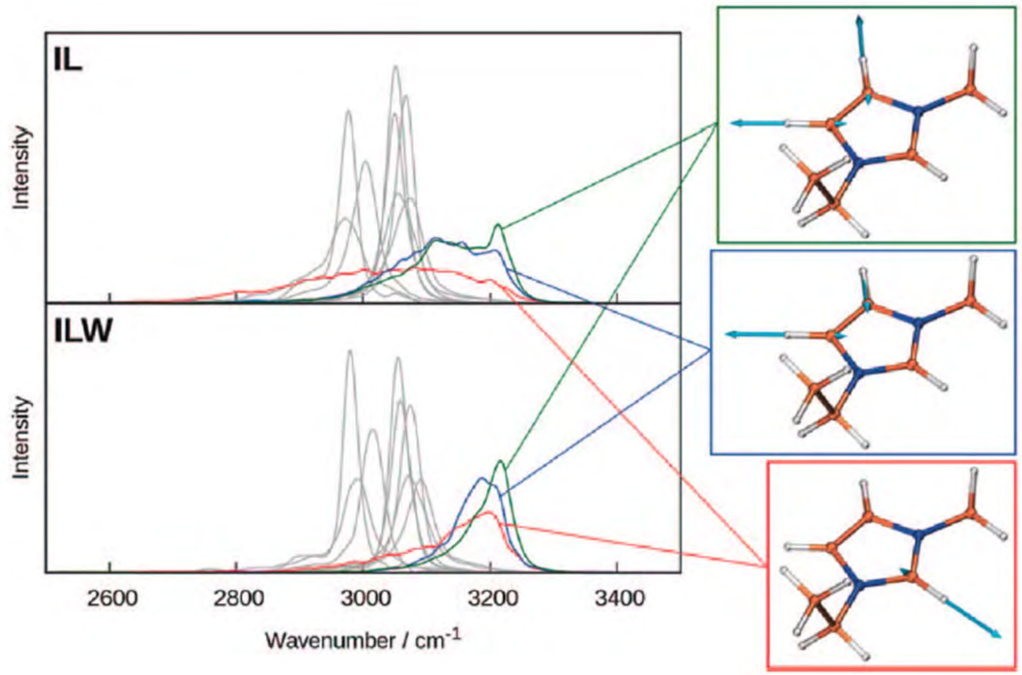


Van Howe Correlation Function & Dynamic Structure Factor

Spectroscopic Analyses

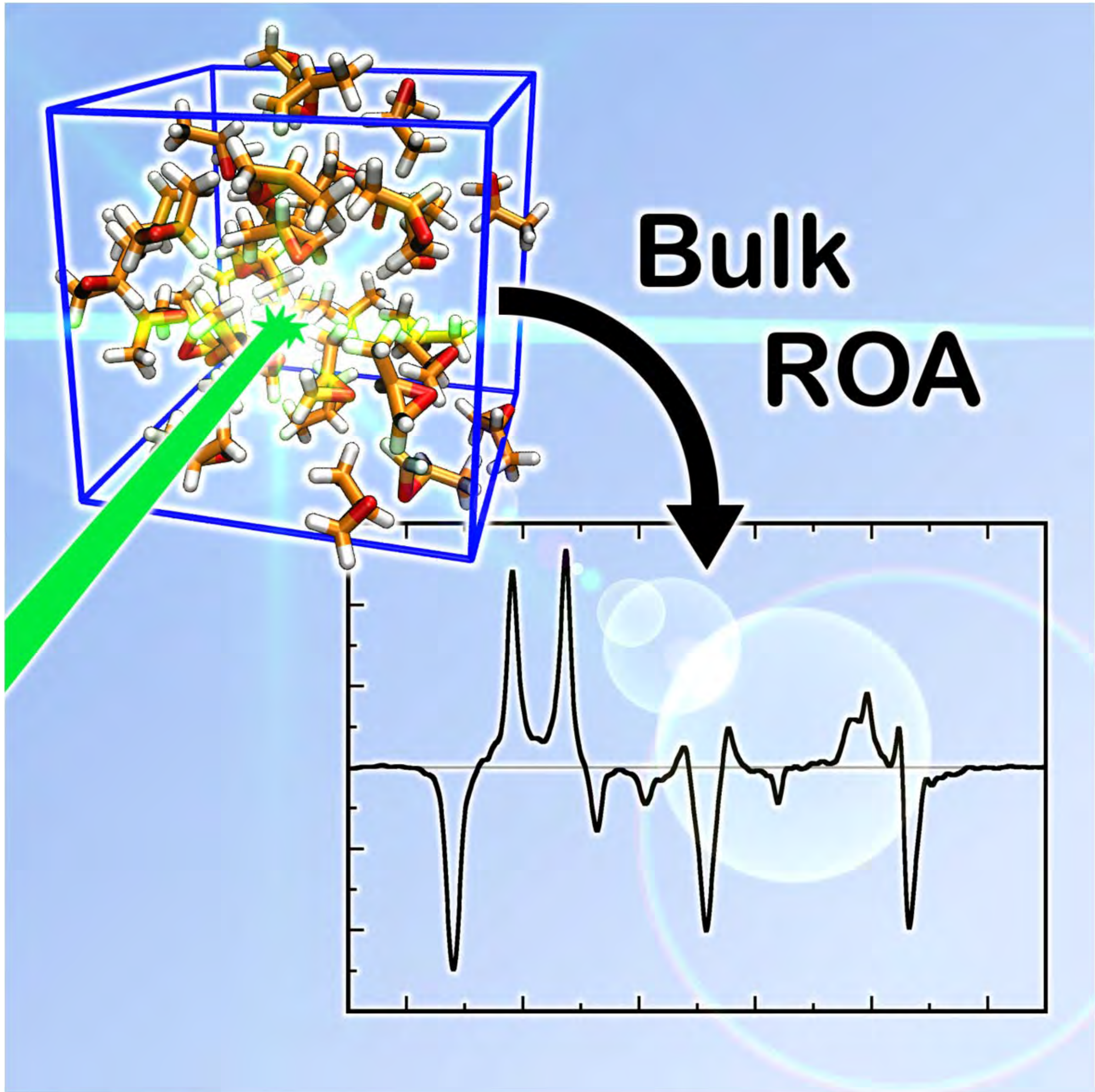


Normal Modes from bulk phase AIMD



M. Thomas, M. Brehm, O. Holloczki,
Z. Kelemen, L. Nyulaszi, T. Pasinszki, B. Kirchner,
J. Chem. Phys. **2014**, *141*, 024510.

2. Computing Vibrational Spectra



Vibrational Spectroscopy

Standard	Chiral
Infrared (IR)	Vibrational Circular Dichroism (VCD)
Raman	Raman Optical Activity (ROA)

Vibrational Spectroscopy

Standard	Chiral
Infrared (IR)	Vibrational Circular Dichroism (VCD)
Raman	Raman Optical Activity (ROA)

Two ways of Improvement:

1. Make calculations on small molecules more accurate
2. Perform calculations on larger/complex systems

Vibrational Spectroscopy

Standard	Chiral
Infrared (IR)	Vibrational Circular Dichroism (VCD)
Raman	Raman Optical Activity (ROA)

Two ways of Improvement:

1. Make calculations on small molecules more accurate
2. **Perform calculations on larger/complex systems**

Vibrational Spectroscopy

Standard	Chiral
Infrared (IR)	Vibrational Circular Dichroism (VCD)
Raman	Raman Optical Activity (ROA)

99% of computed spectra: Double-harmonic approximation

- Only 1 molecule in vacuum *(no solvent effects)*
- Requires minimum structure *(no conformational flexibility)*
- No anharmonic effects *(bad for hydrogen bonds, etc.)*
- No band shapes *(discrete line spectrum)*

Vibrational Spectroscopy

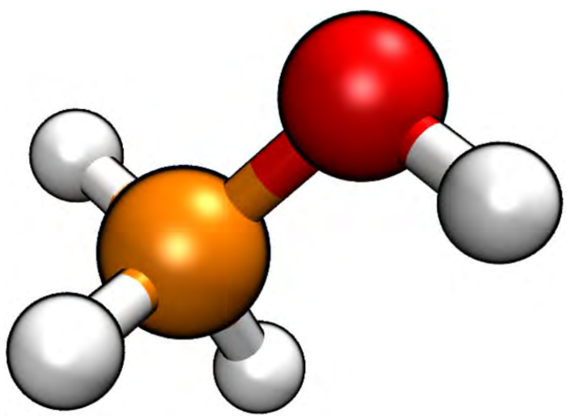
More powerful approach: Spectra from AIMD

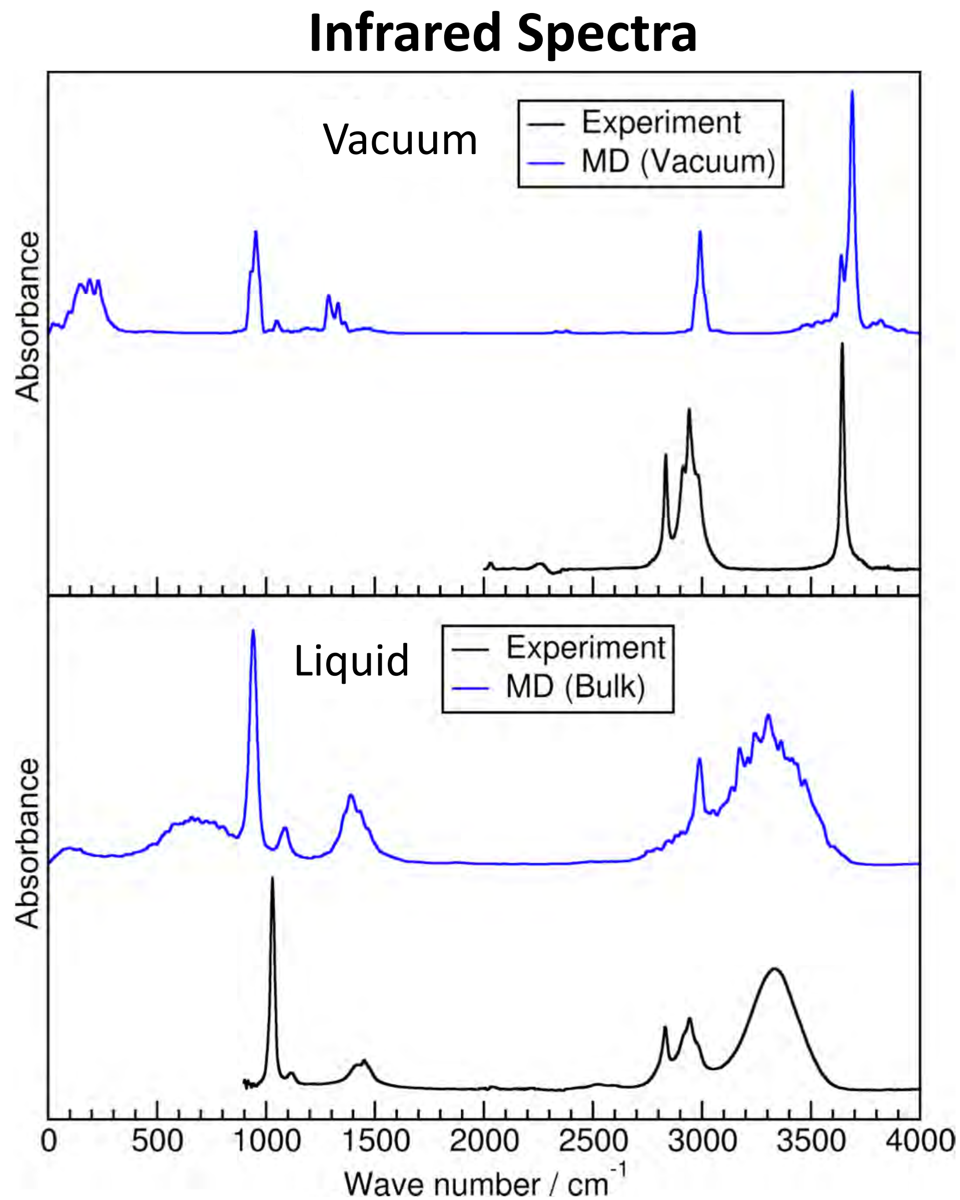
- Periodic bulk phase *(full treatment of solvent effects)*
- Phase space sampling *(full conformational flexibility)*
- Many anharmonic effects covered
  (overtones, combination bands, …); see PhD thesis of M. Thomas
- Nicely reproduces band shapes *(hydrogen bonds, …)*

First published shortly before year 2000 ***(feasibility of AIMD!)***

Spectra are obtained as the Fourier transform of certain
time correlation functions along the trajectory.

Application: Methanol

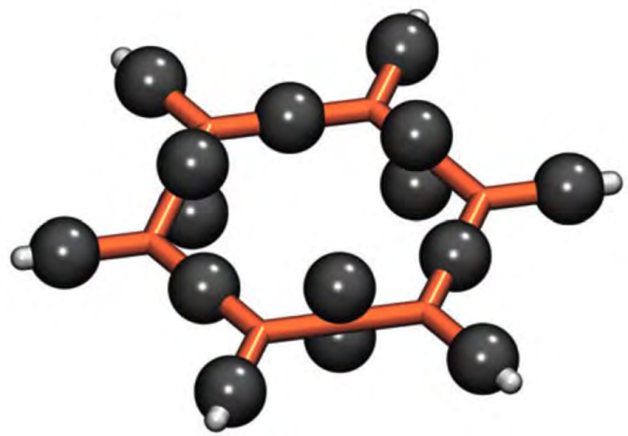




M. Thomas, M. Brehm, R. Fligg, P. Vöhringer, B. Kirchner, *Phys. Chem. Chem. Phys.* **2013**, *15*, pp 6608-6622.

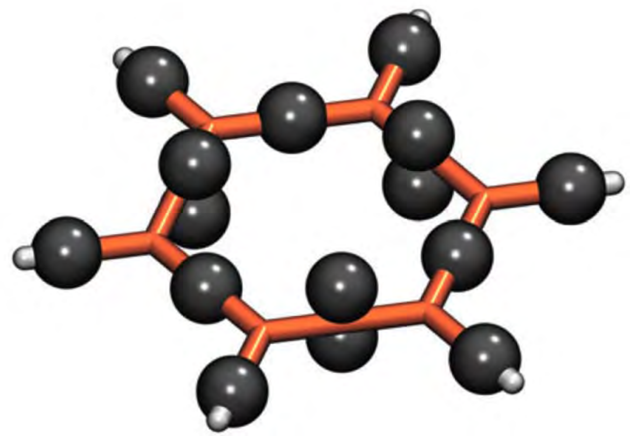
Wannier Centers

- Molecular orbitals are not unique; every unitary transformation yields another set of valid MOs
- Idea: Find set of MOs with minimal (spatial) spread
- Those are called Wannier orbitals
- The center of mass of each Wannier orbital is called Wannier center
- Suggested by G. Wannier for solid-state systems in 1937



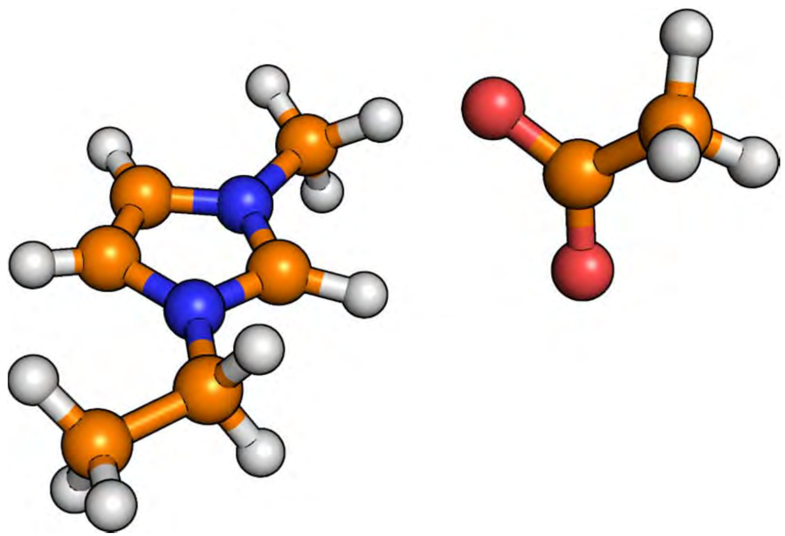
Wannier Centers

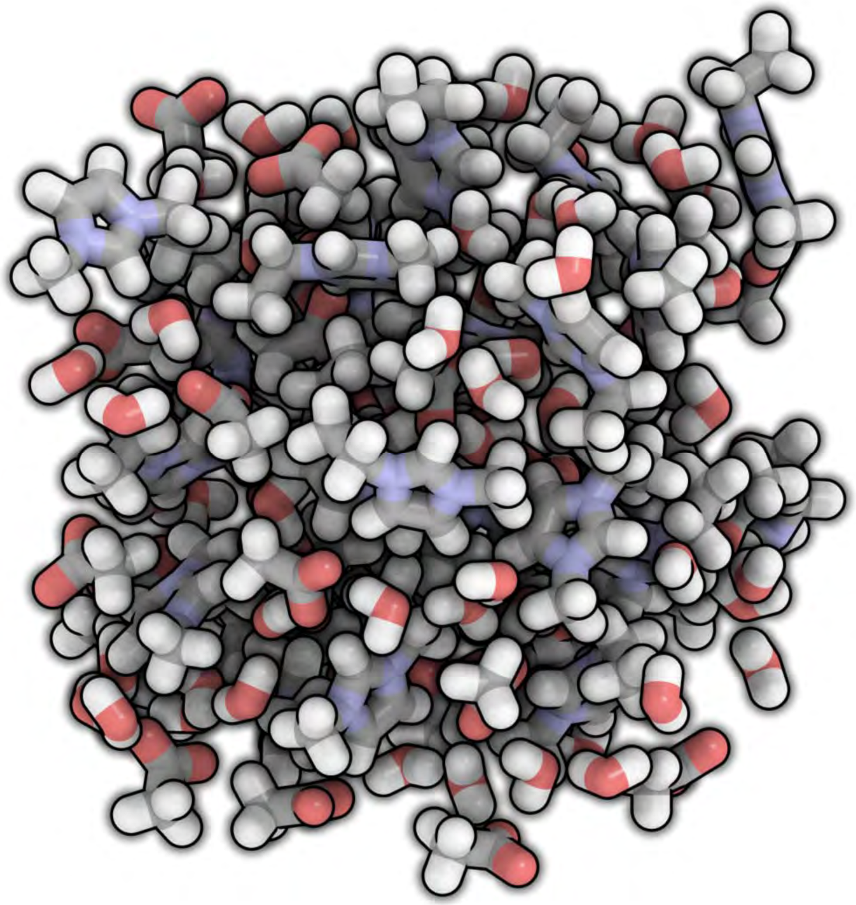
- Can be used to compute molecular dipole moments
- For only 1 molecule in vacuum, dipole from Wannier centers exactly matches „true“ QM dipole (Berry phase)
- Standard algorithm uses Jacobi diagonalization to construct suitable unitary transformation; rather slow...
- CP2k offers a very modern and efficient method, called „Crazy Angle algorithm“ ☺
- Virtually all IR / Raman spectra from AIMD rely on Wannier centers for molecular dipole moments



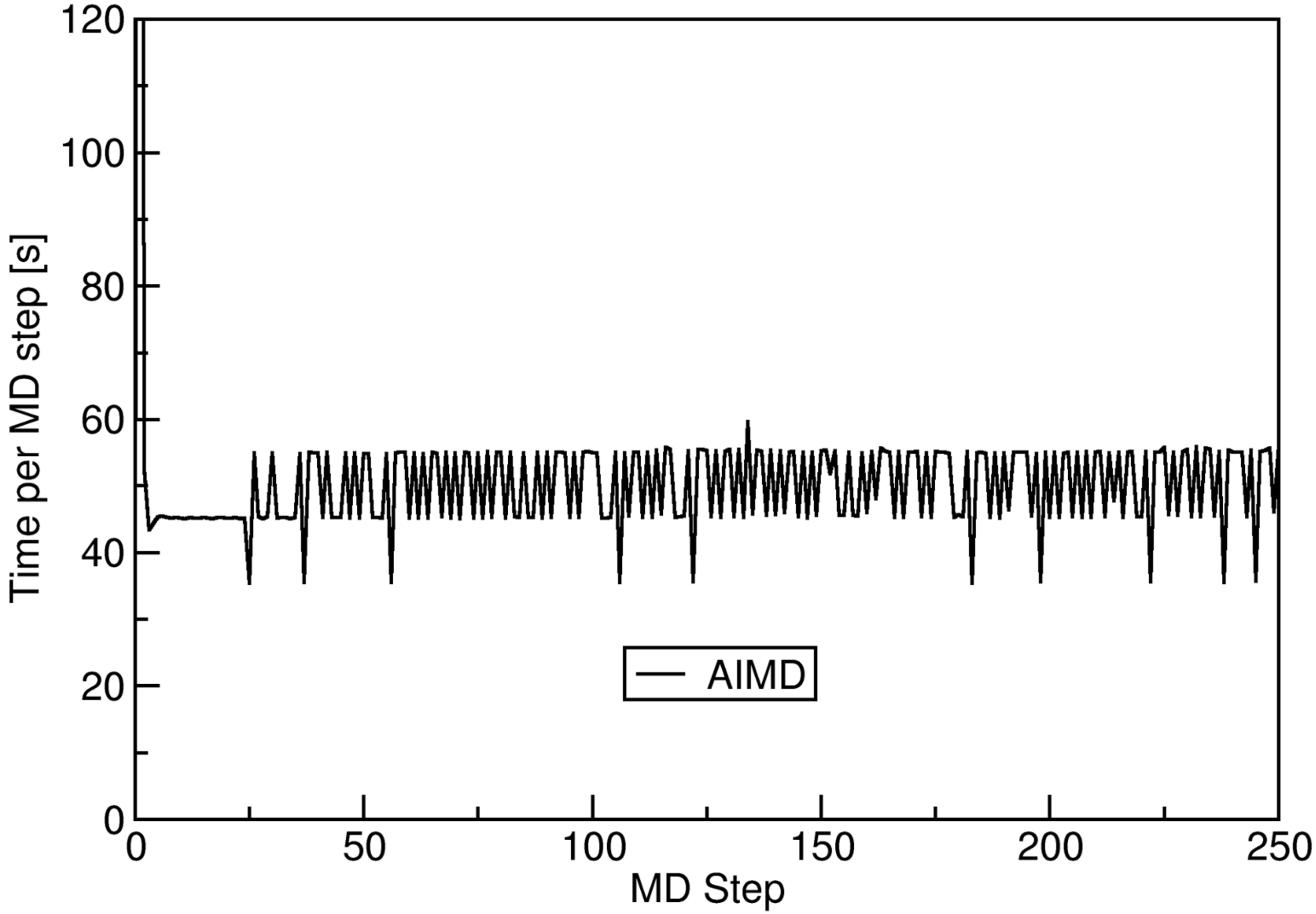
Wannier Centers: Example

Bulk phase [EMIm][OAc], liquid, 936 atoms,
cell 21 x 21 x 21 Å, 350 K, BLYP-D3, DZVP-MOLOPT-SR,
2500 electrons, 8100 basis functions,
1 node (16 cores) Intel Xeon E5-2620 v4

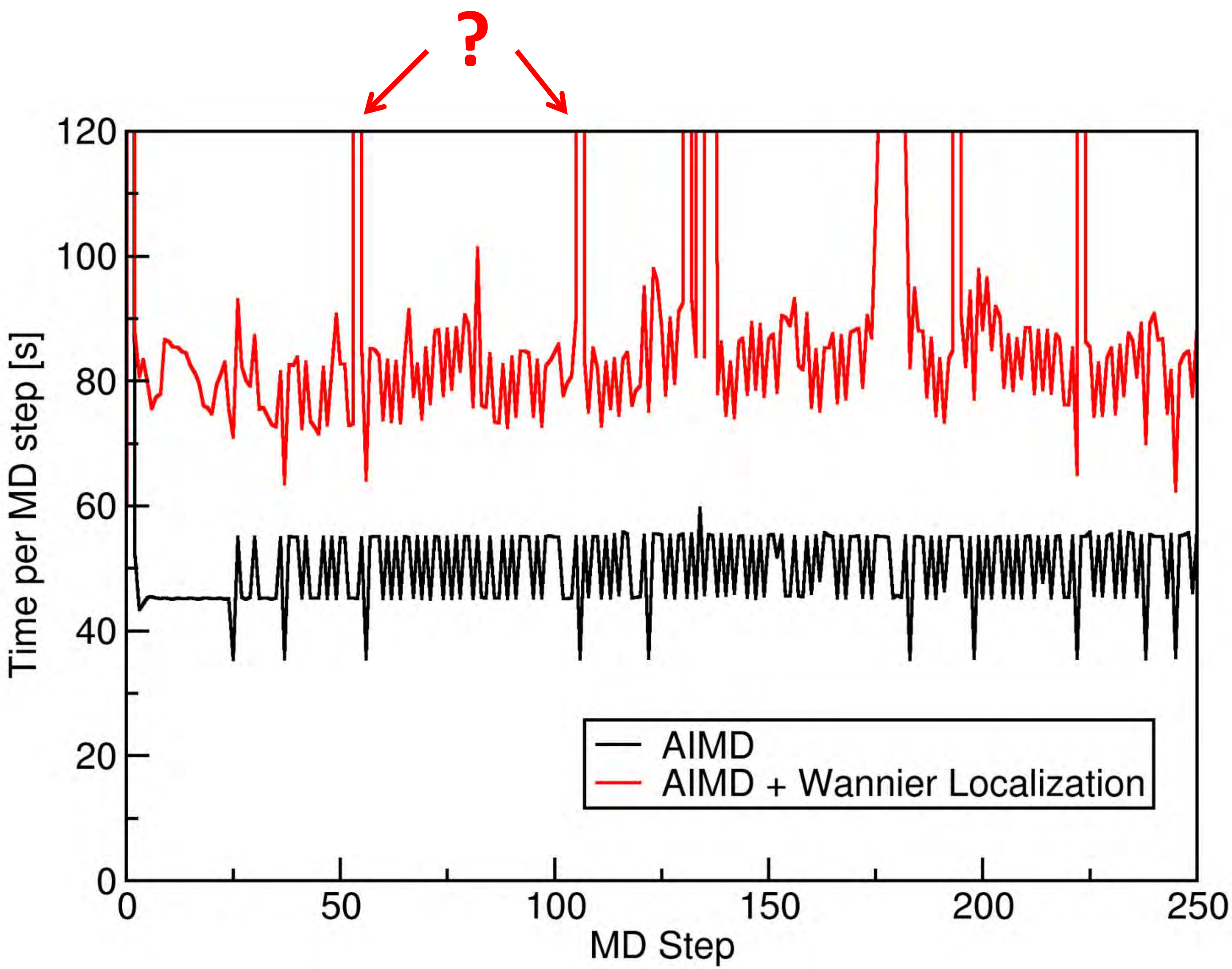




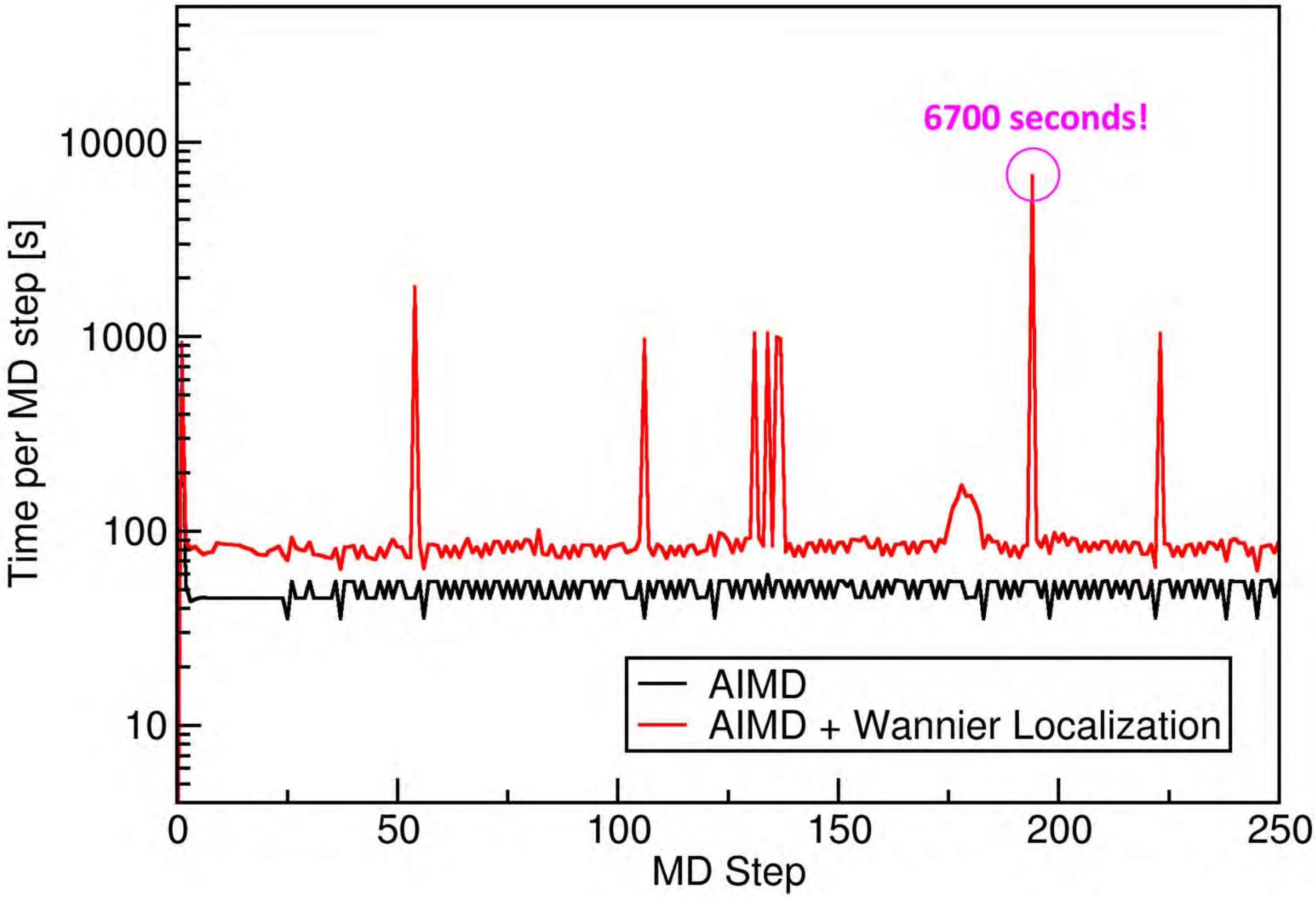
Wannier Centers: Example



Wannier Centers: Example



Wannier Centers: Example



Wannier Centers

```
LOCALIZATION| Computing localization properties for OCCUPIED ORBITALS. Spin:  1
   Spread Functional     sum_in -w_i ln(|z_in|^2)    sum_in w_i(1-|z_in|^2)
   Initial Spread (Berry) :            992039.6748362964   149090.8476625035
CRAZY|   Iter               value         gradient   Max. eval       limit
CRAZY|      1 759416.300978268      0.1314E+05  0.2295E+01  0.2000E+00
CRAZY|      2 1825582.25180607      0.7230E+04  0.5686E+01  0.2000E+00
CRAZY|      3 3504667.50291648      0.4232E+04  0.9659E+01  0.2000E+00
CRAZY|      4 5213901.57181955      0.3002E+03  0.1881E+02  0.2000E+00
 ...
CRAZY|    497 5872400.29576750      0.2857E+03  0.2367E+03  0.2000E+00
CRAZY|    498 5872334.68198040      0.3178E+03  0.2374E+03  0.2000E+00
CRAZY|    499 5872451.22848857      0.3350E+03  0.2378E+03  0.2000E+00
CRAZY|    500 5871893.26651500      0.2990E+03  0.2360E+03  0.2000E+00
   Crazy Wannier localization not converged after    500
   iterations, switching to jacobi rotations
   Localization by iterative distributed Jacobi rotation
                     Iteration            Functional         Tolerance   Time
                           100      -74827.4008233787       0.1916E+04   3.361
                           200      -74882.2665054244       0.2077E+03   3.361
                           300      -74895.0539325070       0.4188E+01   3.362
                           ...
                          1200      -74895.0559064523       0.7365E-03   3.355
                          1300      -74895.0559064549       0.3717E-03   3.353
                          1400      -74895.0559064543       0.1876E-03   3.355
   Localization  for spin   1 converged in   1493 iterations
   Spread Functional     sum_in -w_i ln(|z_in|^2)    sum_in w_i(1-|z_in|^2)
   Total Spread (Berry) :             -22638.0247818508   -74895.0559064557
```

Wannier Centers

```
LOCALIZATION| Computing localization properties for OCCUPIED ORBITALS. Spin:  1
   Spread Functional     sum_in -w_i ln(|z_in|^2)    sum_in w_i(1-|z_in|^2)
   Initial Spread (Berry) :             992039.6748362964   149090.8476625035
CRAZY|   Iter           value          gradient   Max. eval       limit
CRAZY|      1 759416.300978268      0.1314E+05  0.2295E+01  0.2000E+00
CRAZY|      2 1825582.25180607      0.7230E+04  0.5686E+01  0.2000E+00
CRAZY|      3 3504667.50291648      0.4232E+04  0.9659E+01  0.2000E+00
CRAZY|      4 5213901.57181955      0.3002E+03  0.1881E+02  0.2000E+00
 ...
CRAZY|    497 5872400.29576750      0.2857E+03  0.2367E+03  0.2000E+00
CRAZY|    498 5872334.68198040      0.3178E+03  0.2374E+03  0.2000E+00
CRAZY|    499 5872451.22848857      0.3350E+03  0.2378E+03  0.2000E+00
CRAZY|    500 5871893.26651500      0.2990E+03  0.2360E+03  0.2000E+00
   Crazy Wannier localization not converged after    500
   iterations, switching to jacobi rotations
   Localization by iterative distri
                    Iteration                                  e   Time
                          100                                  4   3.361
                          200                                  3   3.361
                          300      -74895.0539325070       0.4188E+01   3.362
                          ...
                         1200      -74895.0559064523       0.7365E-03   3.355
                         1300      -74895.0559064549       0.3717E-03   3.353
                         1400      -74895.0559064543       0.1876E-03   3.355
   Localization  for spin   1 converged in   1493 iterations
   Spread Functional     sum_in -w_i ln(|z_in|^2)    sum_in w_i(1-|z_in|^2)
   Total Spread (Berry) :              -22638.0247818508   -74895.0559064557
```

112 Minutes!

Wannier Centers

Average frame times:

Standard AIMD: 47.9 seconds
AIMD + Wannier: 139.3 seconds

→ Wannier localization takes 91.4 seconds on average!

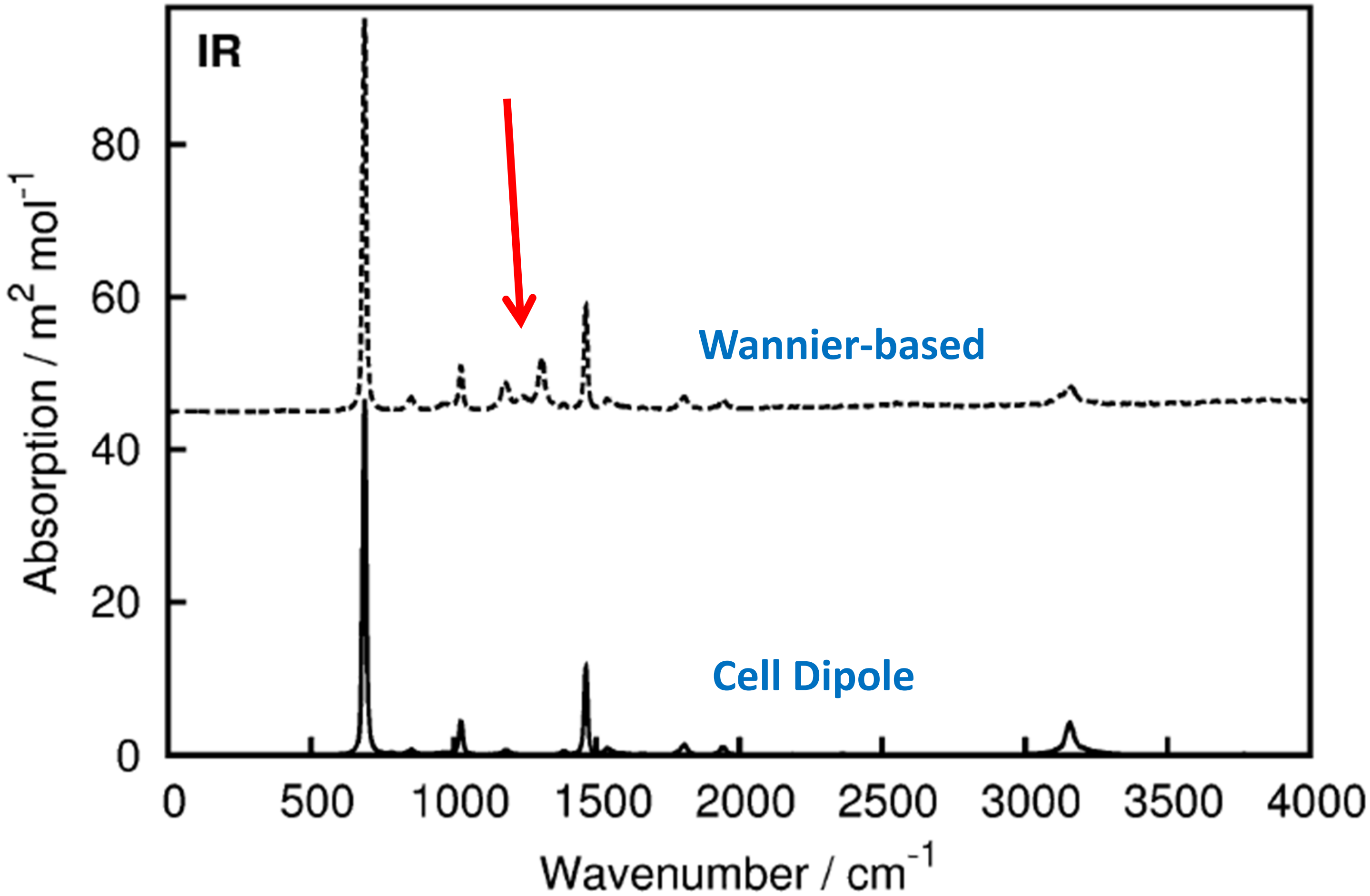
65 % of total computer time goes into localization only...

Even if CRAZY **<u>would</u>** converge in every step,
localization would take ≈ 30 seconds per frame.

It can even happen that JACOBI does not converge;
then no Wannier centers at all are available for that frame...

Wannier Centers

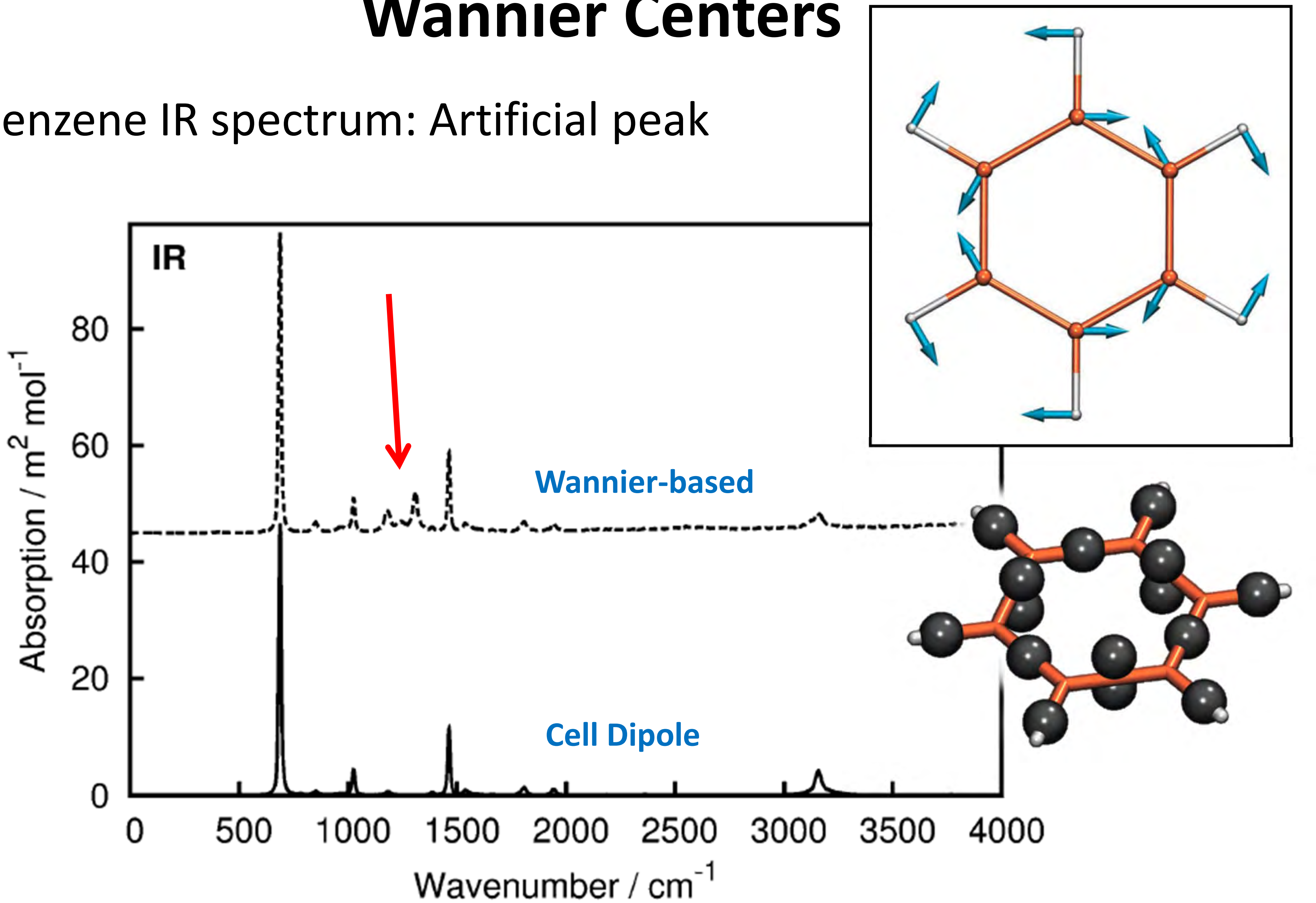
Benzene IR spectrum: Artificial peak



M. Thomas, M. Brehm, B. Kirchner, *Phys. Chem. Chem. Phys.* **2015**, *17*, pp 3207-3213.

Wannier Centers

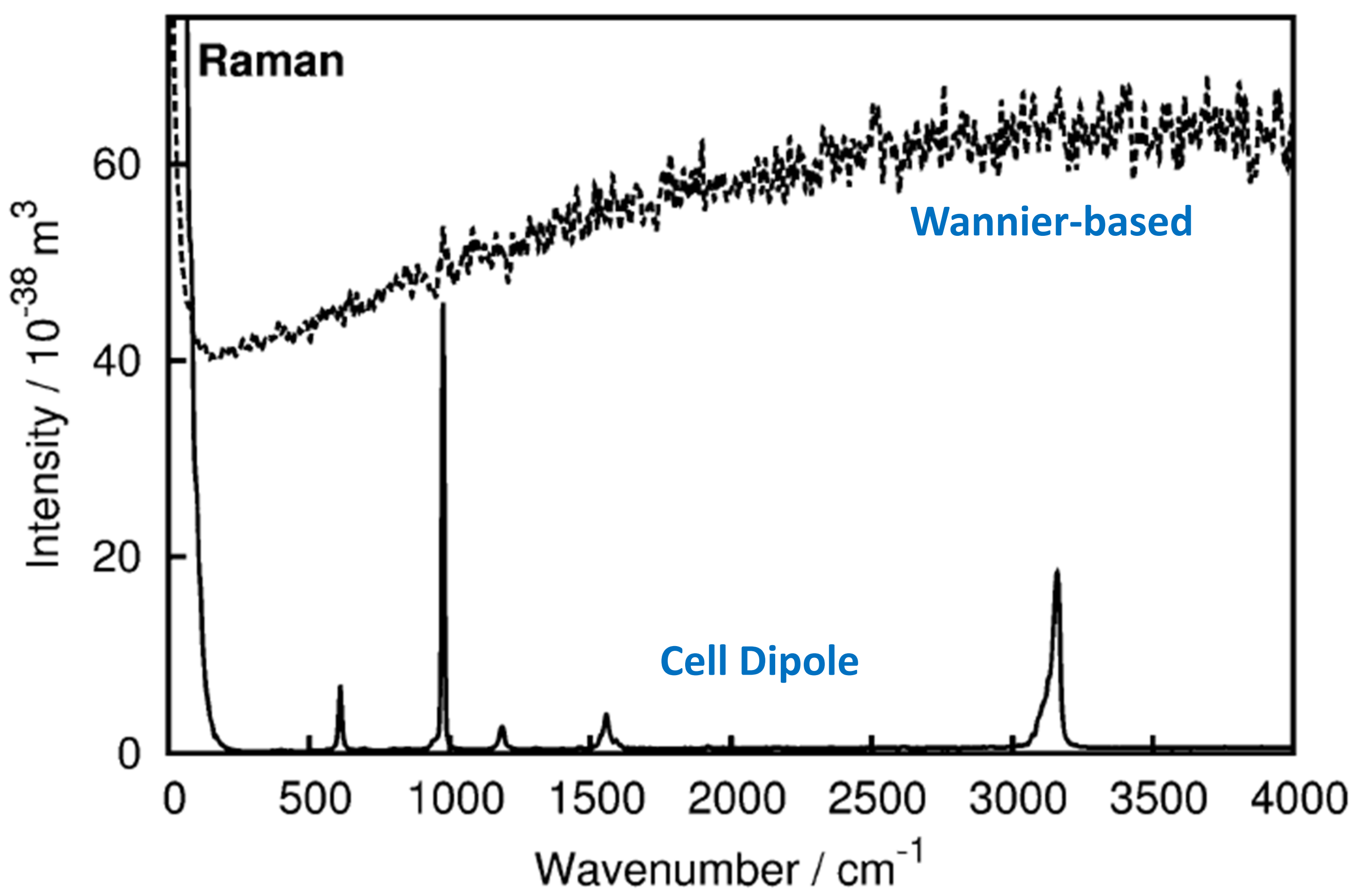
Benzene IR spectrum: Artificial peak



M. Thomas, M. Brehm, B. Kirchner, *Phys. Chem. Chem. Phys.* **2015**, *17*, pp 3207-3213.

Wannier Centers

Benzene Raman: No spectrum at all, only noise…



M. Thomas, M. Brehm, B. Kirchner, *Phys. Chem. Chem. Phys.* **2015**, *17*, pp 3207-3213.

Wannier Centers

Most IR / Raman spectra from AIMD use Wannier centers.

Disadvantages:

- Huge computational overhead *(for systems with ≈ 1000 atoms: around 65 % of the total CPU time!)*
- Not guaranteed to converge at all
- Severe problems in aromatic systems *(artificial bands; polarizability not accessible)*
- Only works for electric dipole; can‘t reproduce quadrupole (required for ROA) → won‘t work for ROA anyway

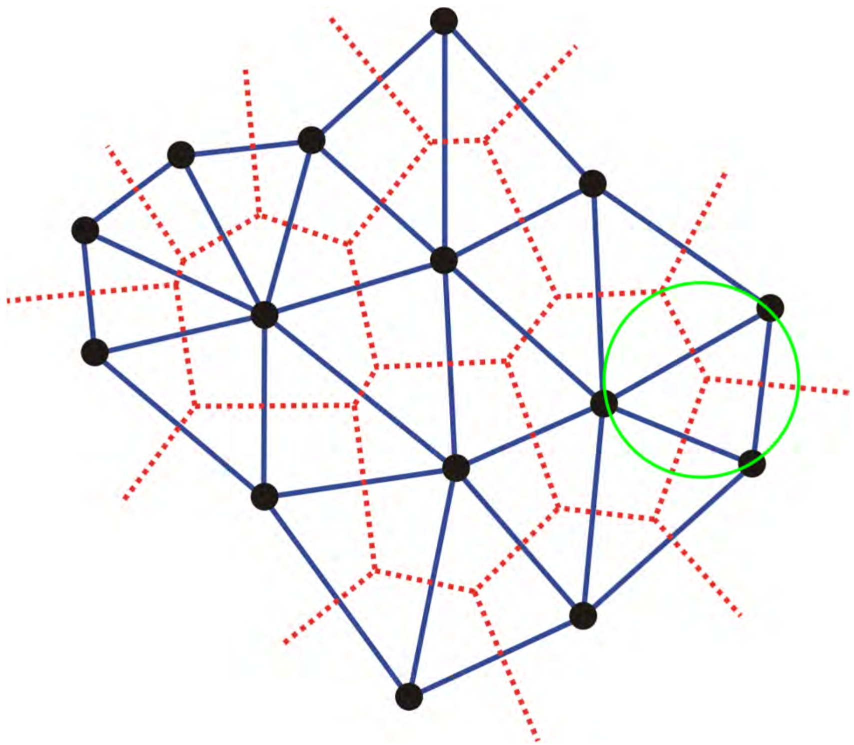
Voronoi Integration

Our idea: Completely drop Wannier localization;
Integrate molecular dipole via Voronoi instead.

Voronoi Integration

Our idea: Completely drop Wannier localization;
Integrate molecular dipole via Voronoi instead.

Voronoi Tessellation (G. Voronoi, 1908):



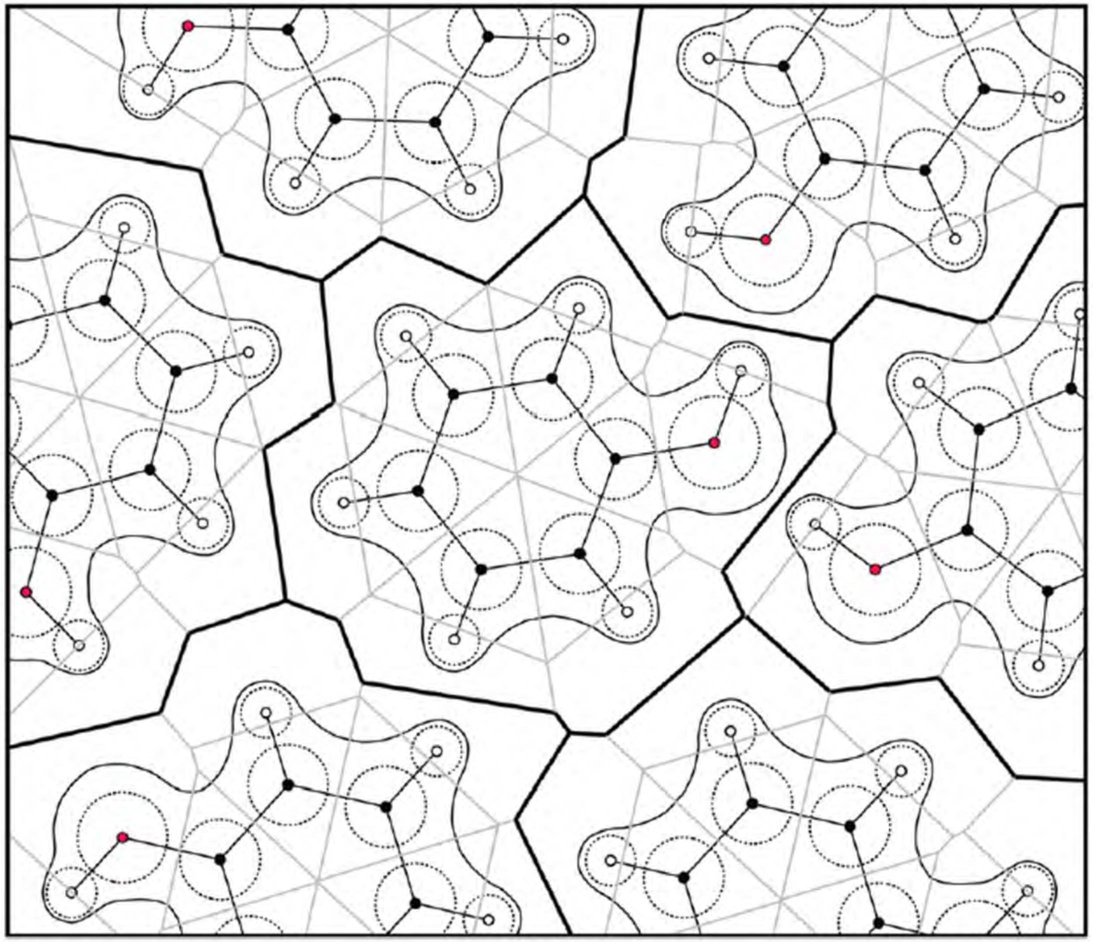
Voronoi Integration

Our idea: Completely drop Wannier localization;
Integrate molecular dipole via Voronoi instead.

Also works in 3D → show video

Voronoi Integration

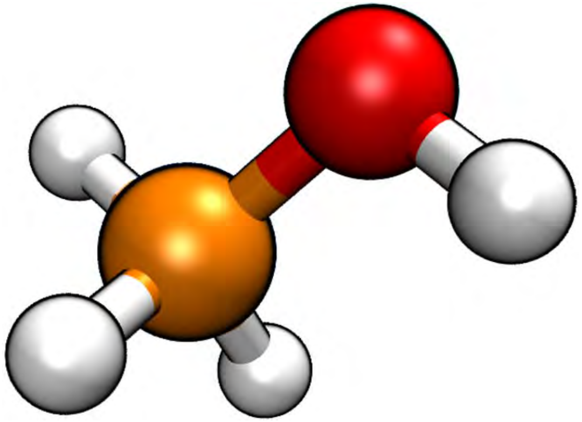
Our idea: Completely drop Wannier localization;
Integrate molecular dipole via Voronoi instead.

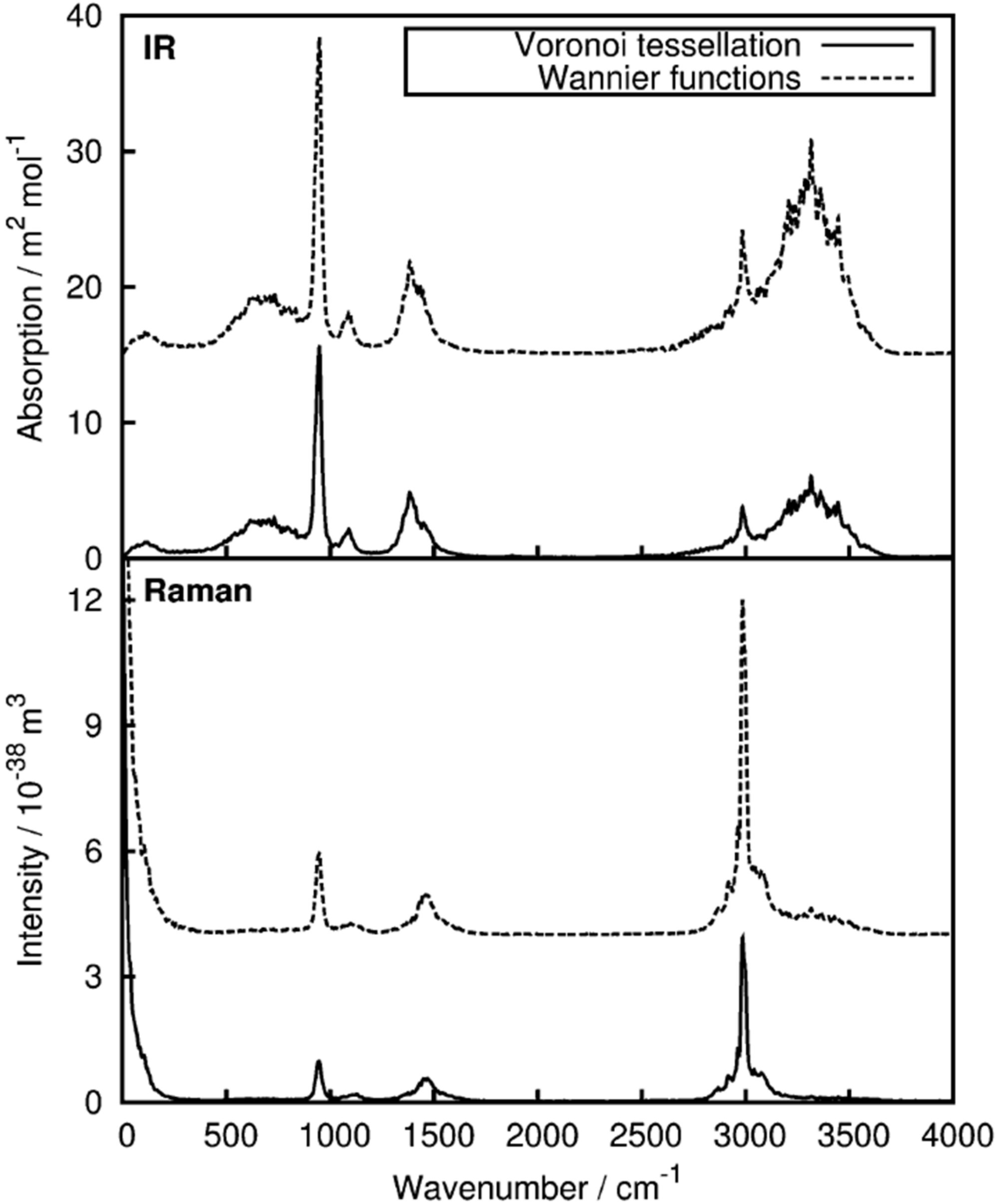


Voronoi Integration

For bulk methanol:

No difference





Voronoi Integration

Average Timings

Wannier-based:

```
 47.9 s AIMD
 91.4 s Localization
139.3 s per frame
```

Voronoi Integration:

```
 47.9 s AIMD
 10.0 s Write CUBE (≈100 MiB)
  2.0 s Voronoi Integration
 59.9 s per frame
```

Saves more than a factor of 2 in total CPU time!

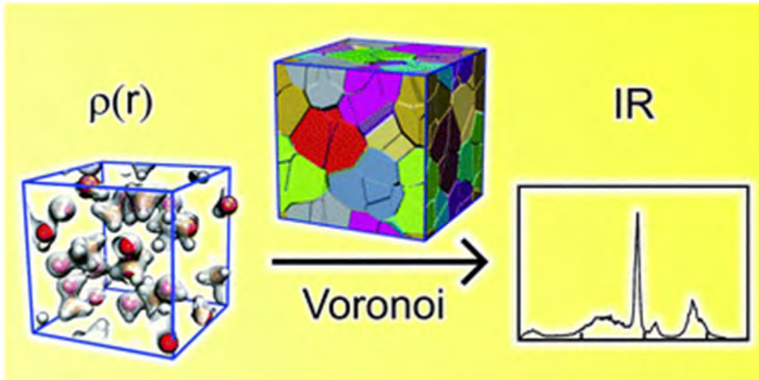
Voronoi Integration

Our idea: Completely drop Wannier localization;
Integrate molecular Dipole via Voronoi instead

- Spectra look almost identical
- Saves lots of computational time (Factor > 2)
- All problems due to Wannier localization are gone
- Works also for higher multipole moments

Published:

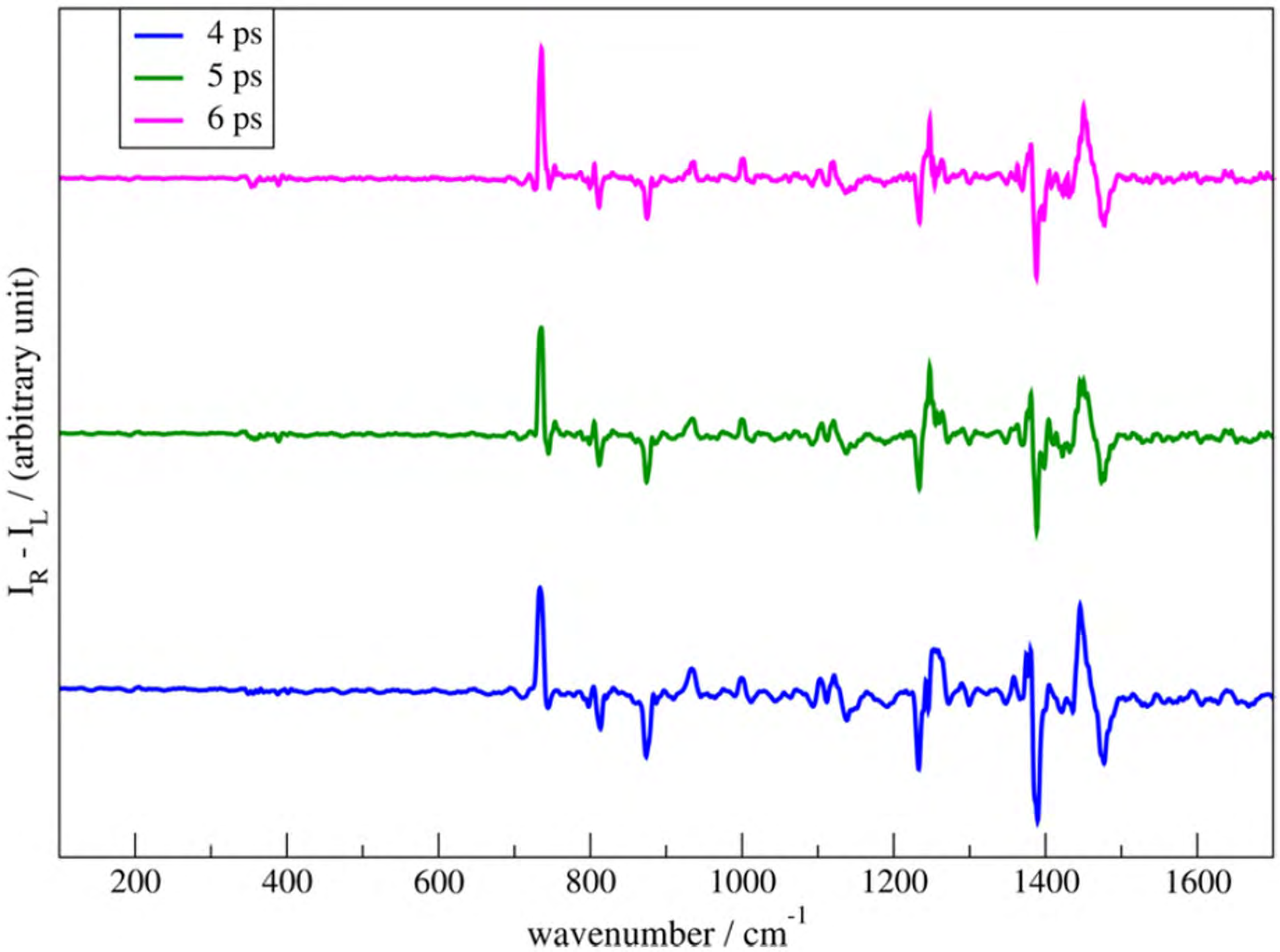
M. Thomas, M. Brehm, B. Kirchner: "Voronoi dipole moments for the simulation of bulk phase vibrational spectra",
Phys. Chem. Chem. Phys. **2015**, *17*, pp 3207-3213.



Vibrational Spectra from AIMD

- **Infrared:** Since ≈ 1997 → **„standard method“**.
- **Raman:** Since ≈ 2002 → **„standard method“**.
- **VCD:** First bulk phase spectrum from *Thomas et al.* in Jan 2016.
  Shortly after: *Scherrer et al.* in Aug 2016.
  Both methods work nicely → **„solved“**.
- **ROA:** Article from *S. Luber* in Feb 2017.
  „the first ROA spectrum from AIMD“.

  But only 1 molecule in vacuum (non-periodic);
  author states that application to periodic bulk systems
  remains open for the future…



S. Luber, *J. Chem. Theory Comput.* **2017**, *13 (3)*, pp 1254–1262.

Vibrational Spectra from AIMD

- **Infrared:** Since ≈ 1997 → **„standard method"**.
- **Raman:** Since ≈ 2002 → **„standard method"**.
- **VCD:** First bulk phase spectrum from *Thomas et al.* in Jan 2016.
  Shortly after: *Scherrer et al.* in Aug 2016.
  Both methods work nicely → **„solved"**.
- **ROA:** Article from *S. Luber* in Feb 2017.
  „the first ROA spectrum from AIMD".

  But only 1 molecule in vacuum (non-periodic);
  author states that application to periodic bulk systems remains open for the future...

IR, Raman, VCD spectra of periodic bulk phase systems can be routinely computed. **ROA is still missing.**

Magnetic Moments

Why is it so hard to obtain VCD and ROA from AIMD?

Because both require the ***magnetic dipole moment***.

Magnetic moments result from electric current.

This is a time-dependent phenomenon.

However, AIMD applies the Born–Oppenheimer approximation.

→ Static limit, no time-dependence in QC, no electric current ☹

How to overcome this dilemma?

Magnetic Moments

(a) Use Perturbation Theory

- Comes in different formulations (e.g. NVPT).
- Applied by *Luber* for VCD and ROA spectra, and by *Scherrer et al.* for VCD spectra
- Idea is known for a long time, but hard to implement/compute…
- Problems, e.g., with non-local pseudopotentials
- Currently works for LDA/GGA DFT only (to the best of my knowledge)

Magnetic Moments

(b) Use a purely classical approach

M. Thomas, B. Kirchner, *J. Phys. Chem. Lett.* **7**, 2016, 509–513.

Presented by *Thomas et al.* in Jan 2016.

Apply some grave simplifications:

- Treat system as if it were classical (macroscopic).
- No eddy currents can flow.
- Currents flow only where there is some electron density.

Start with the continuity equation:

$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t} + \nabla \cdot \mathbf{j}(\mathbf{r}, t) = 0$$

Approximate current as a simple product:

$$\mathbf{j}(\mathbf{r}, t) = -\rho(\mathbf{r}, t) \nabla \alpha(\mathbf{r}, t)$$

Then obtain the following PDE:

$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t} = \nabla \rho(\mathbf{r}, t) \cdot \nabla \alpha(\mathbf{r}, t) + \rho(\mathbf{r}, t) \Delta \alpha(\mathbf{r}, t)$$

Magnetic Moments

(b) Use a purely classical approach

M. Thomas, B. Kirchner, *J. Phys. Chem. Lett.* **7**, 2016, 509–513.

Presented by *Thomas et al.* in Jan 2016.

Apply some grave simplifications:

- Treat system as if it were classical (macroscopic).
- No eddy currents can flow.
- Currents flow only where there is some electron density.

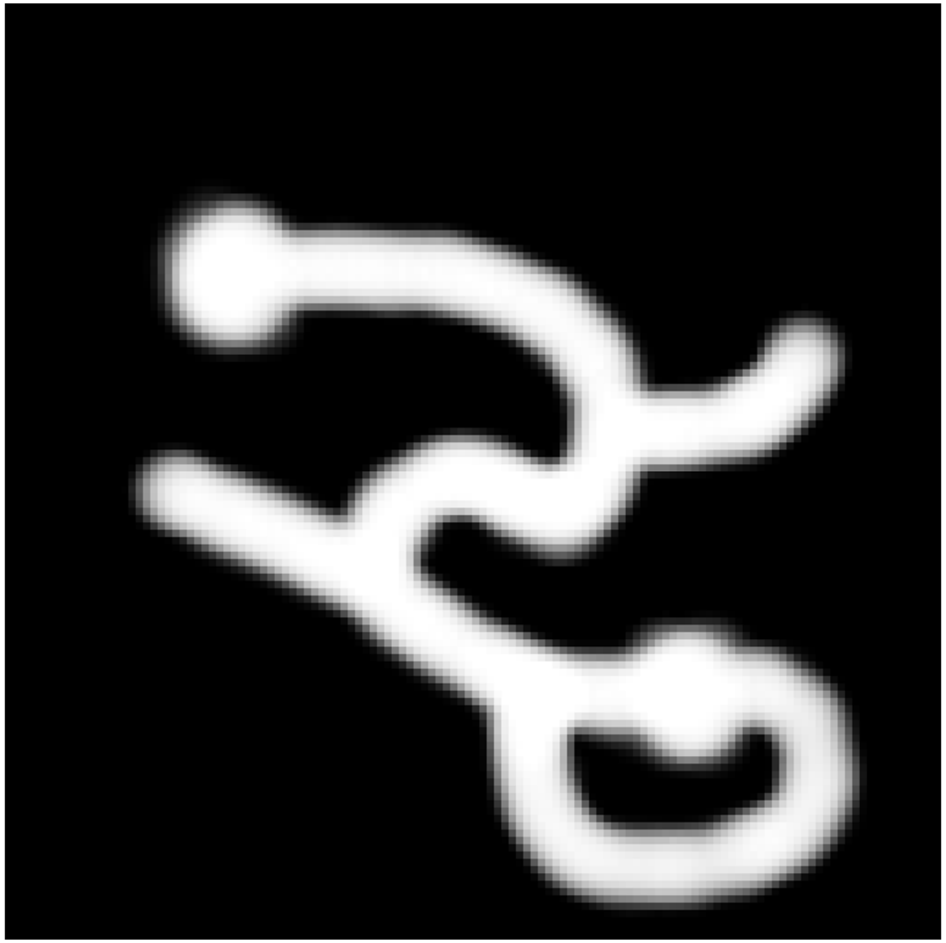
The PDE is discretized on a grid as a (huge) linear system of equations (10^6 variables, 10^6 equations, sparse coefficient matrix).

This system is iteratively solved via the preconditioned bijoncugate gradient stabilized („BiCGStab“) method.

Magnetic Moments

(b) Use a purely classical approach

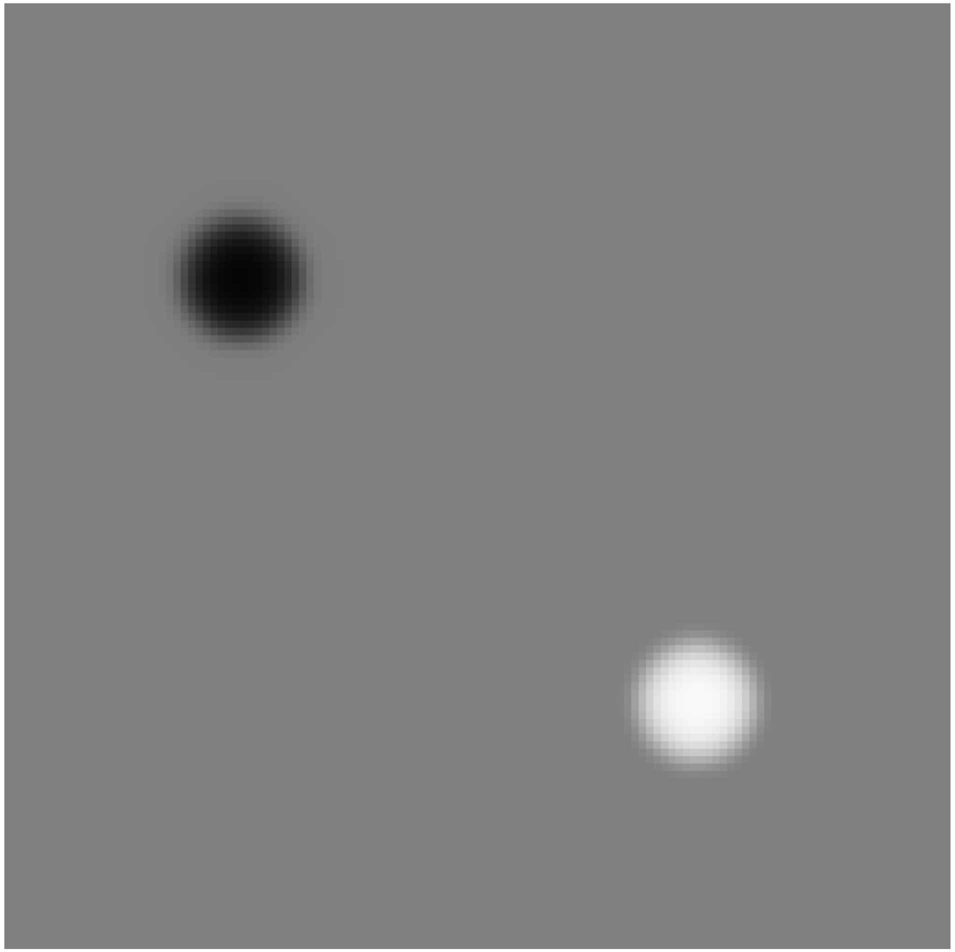
$$\rho(\mathbf{r}, t)$$



Magnetic Moments

(b) Use a purely classical approach

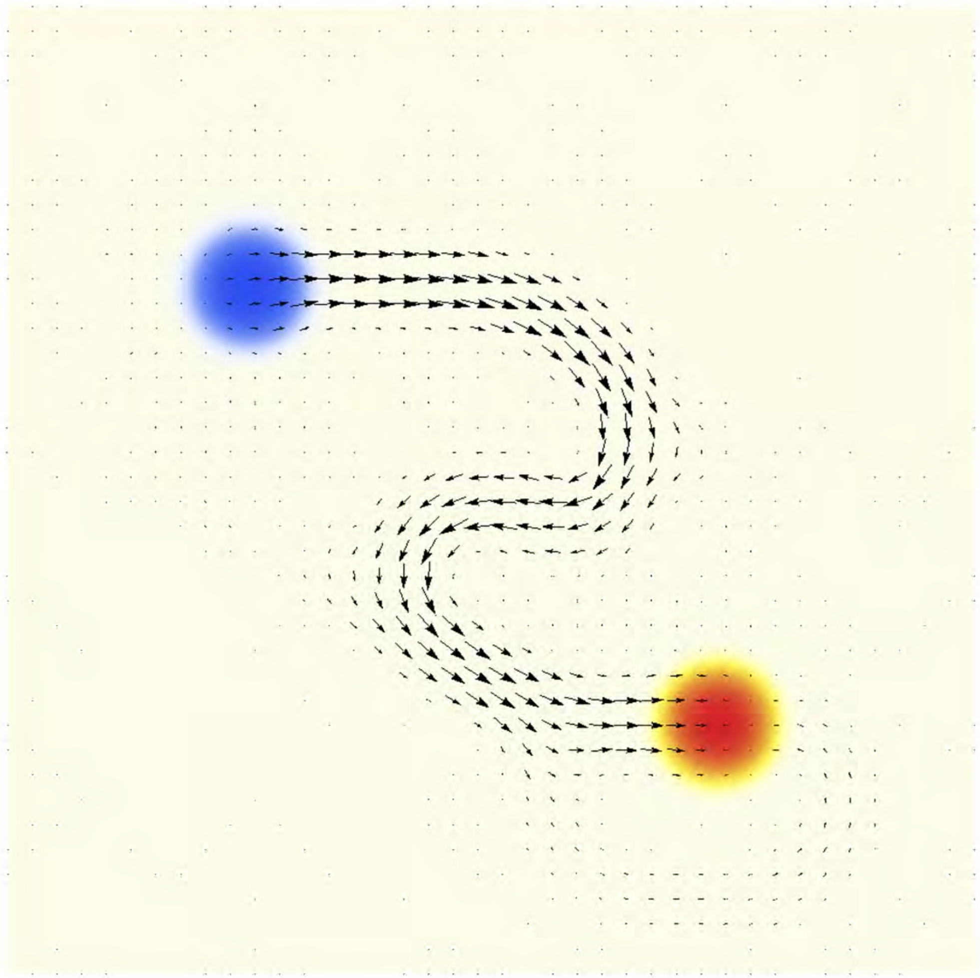
$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t}$$



Magnetic Moments

(b) Use a purely classical approach

$$\mathbf{j}(\mathbf{r},\ t)$$



Magnetic Moments

M. Thomas, B. Kirchner, *J. Phys. Chem. Lett.* **7**, 2016, 509–513.

(b) Use a purely classical approach

Presented by *Thomas et al.* in Jan 2016.

Apply some grave simplifications:

- Treat system as if it were classical (macroscopic).
- No eddy currents can flow.
- Currents flow only where there is some electron density.

Based on these assumptions, obtain current from two successive electron density snapshots on a grid.

Have to solve a partial differential equation (PDE) on a 3D grid.

Magnetic Moments

(b) Use a purely classical approach

Advantage: Only requires electron density on grid as input.

- Works with *all* electron structure methods
  that give total electron density *(MP2, CCSD, …).*

- No modification in QC code required – can be combined
  with *any* QC software that writes CUBE files (*e.g.*, CP2k).

- Solving the PDE is fast (≈ 10 seconds per frame);
  total computational time is only that of a standard AIMD.

→ Much faster than NVPT, very good scaling, can be applied
to periodic systems with > 1000 atoms (in case of DFT).

Magnetic Moments

(b) Use a purely classical approach

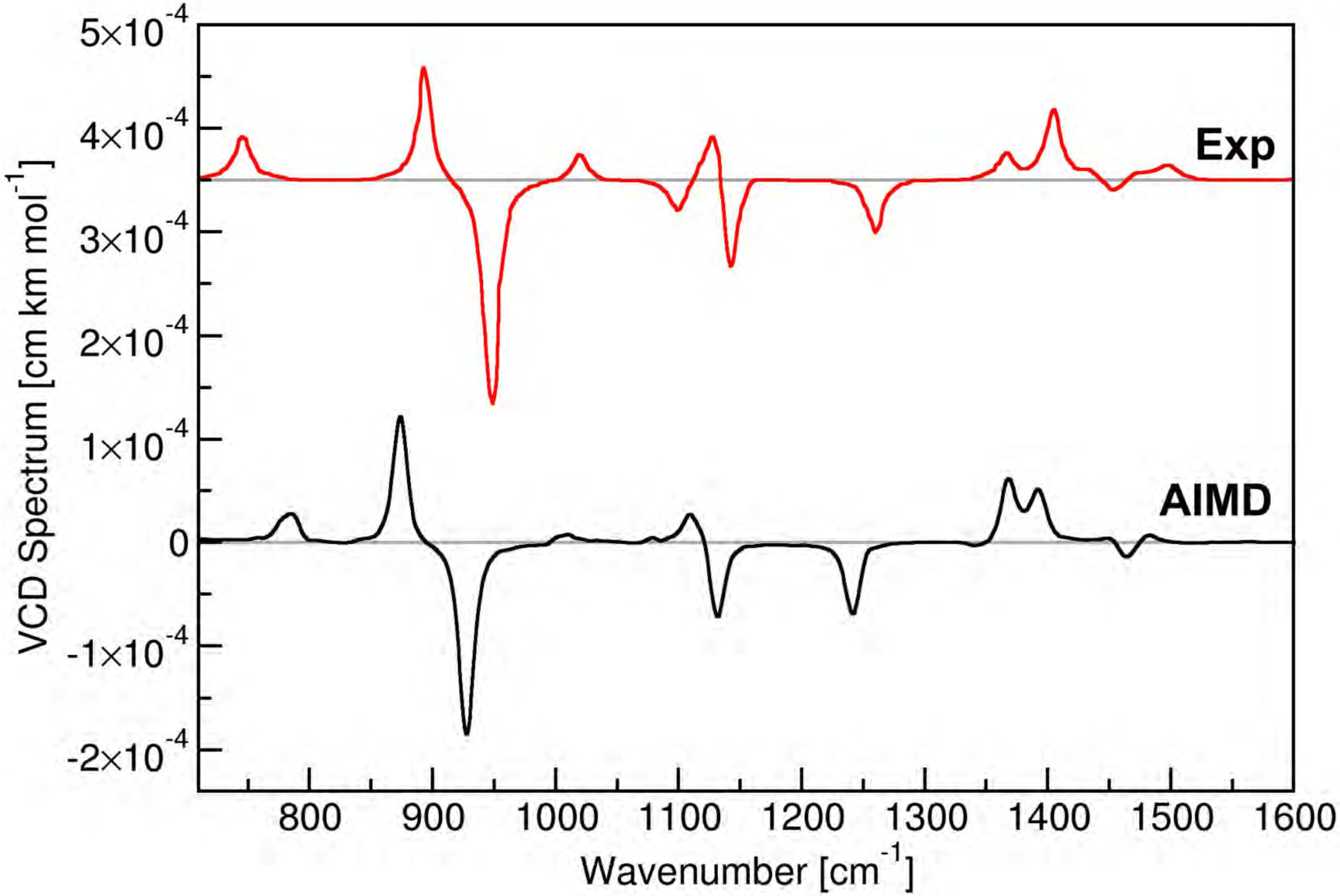
Disadvantage: Ridiculous simplification.

„There is no physical reason why the results of this approach should make any sense.“

However, it worked very nicely for prediction of VCD.

M. Thomas, B. Kirchner, *J. Phys. Chem. Lett.* **7**, 2016, 509–513.

Predicted VCD spectrum of *(R)*-propylene oxide



→ **Classical approach indeed works!**

Magnetic Moments

(b) Use a purely classical approach

Disadvantage: Ridiculous simplification.

„There is no physical reason why the results of this approach should make any sense.“

However, it worked very nicely for prediction of VCD.

→ We now have all ingredients for ROA *(since Jan 2016)*

Our Approach to ROA

Computation of ROA spectra requires three properties:

- Electric dipole – electric dipole polarizability
- Electric quadrupole – electric dipole polarizability
- **Magnetic dipole – electric dipole polarizability**

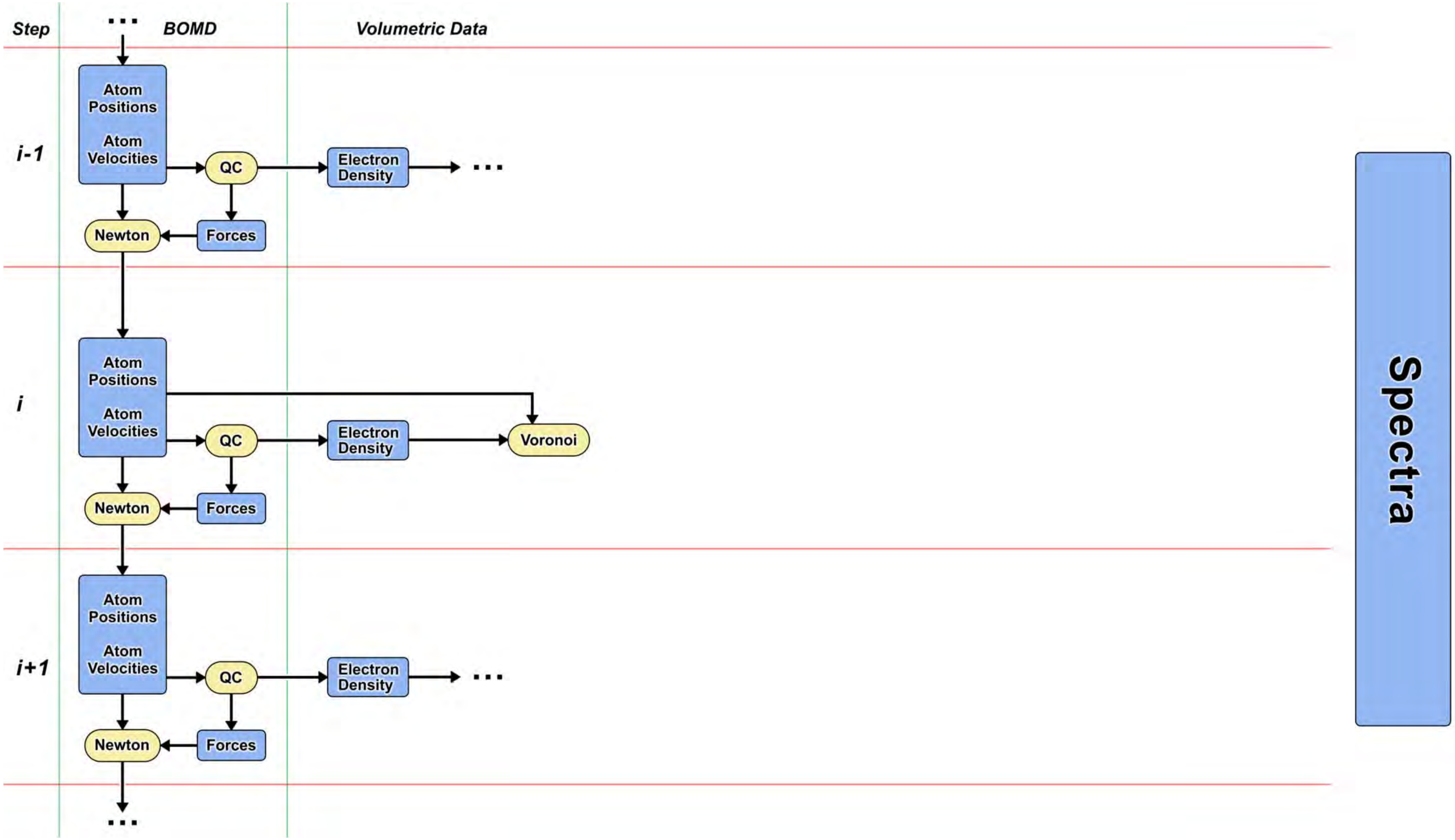
The spectrum can be obtained as FT of cross-correlations of those properties along the trajectory...

Sounds „easy“, but required ≈ 1 year of additional effort.

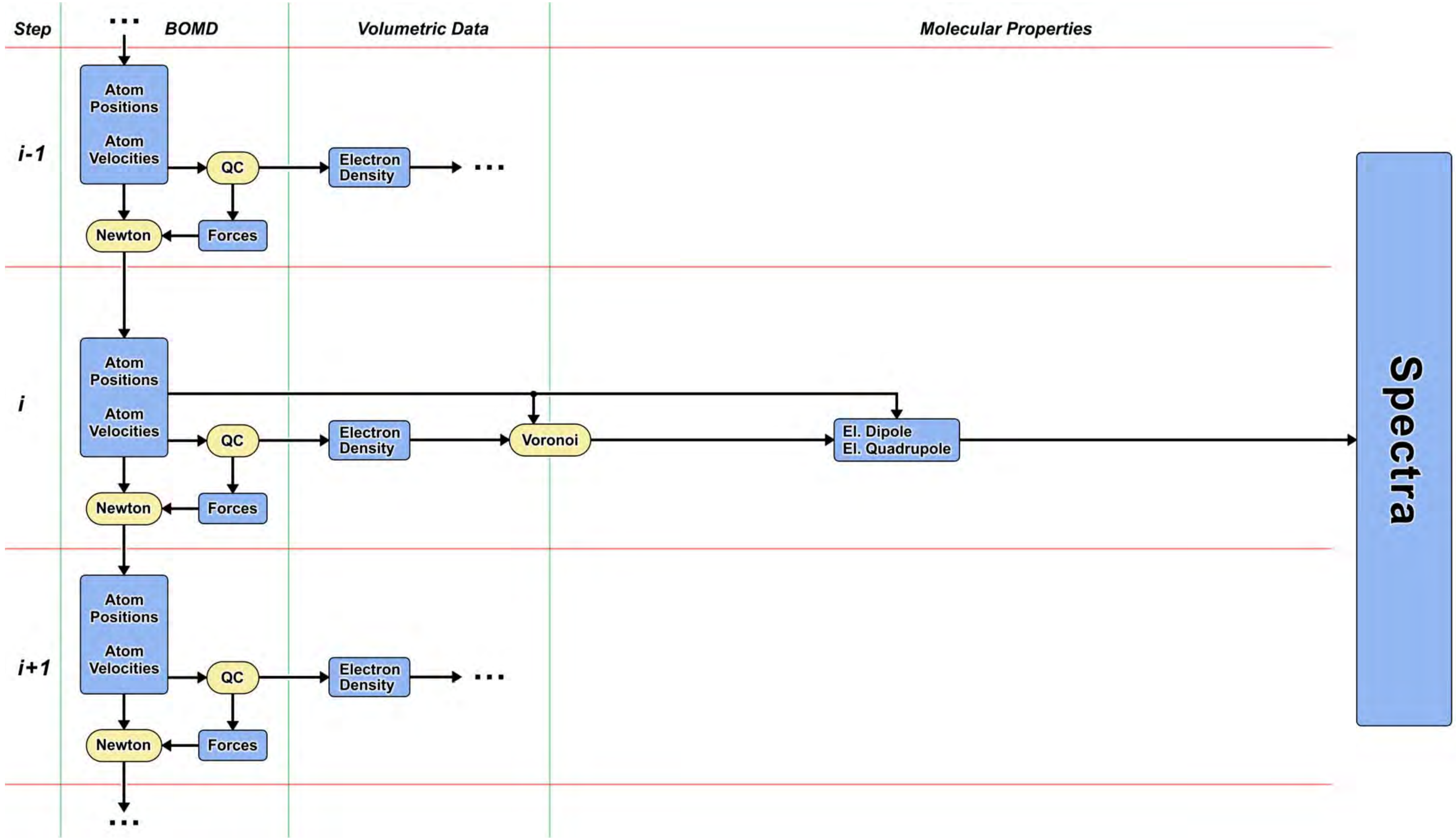
Our Approach to ROA

Our Approach to ROA

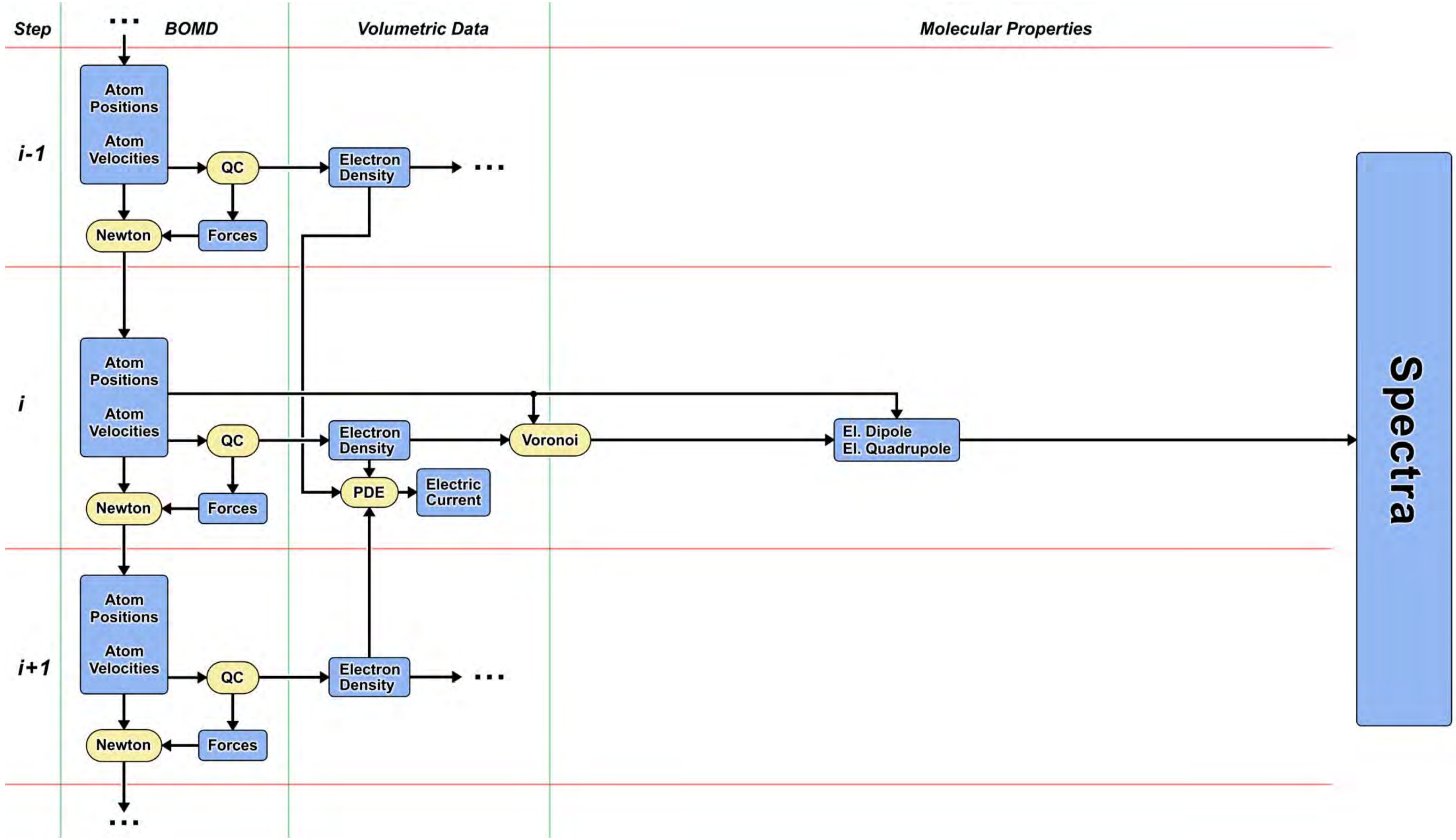
Our Approach to ROA



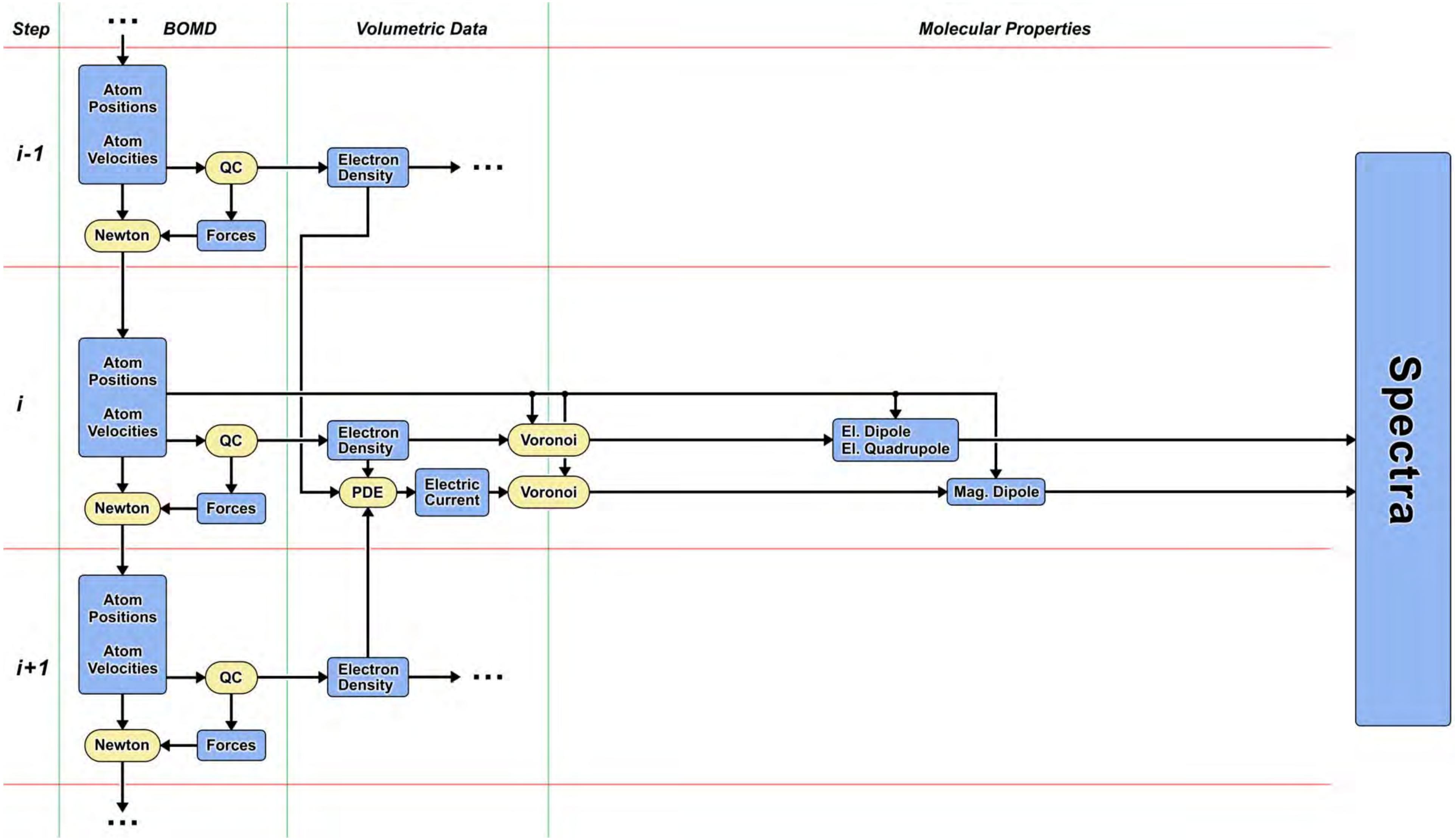
Our Approach to ROA



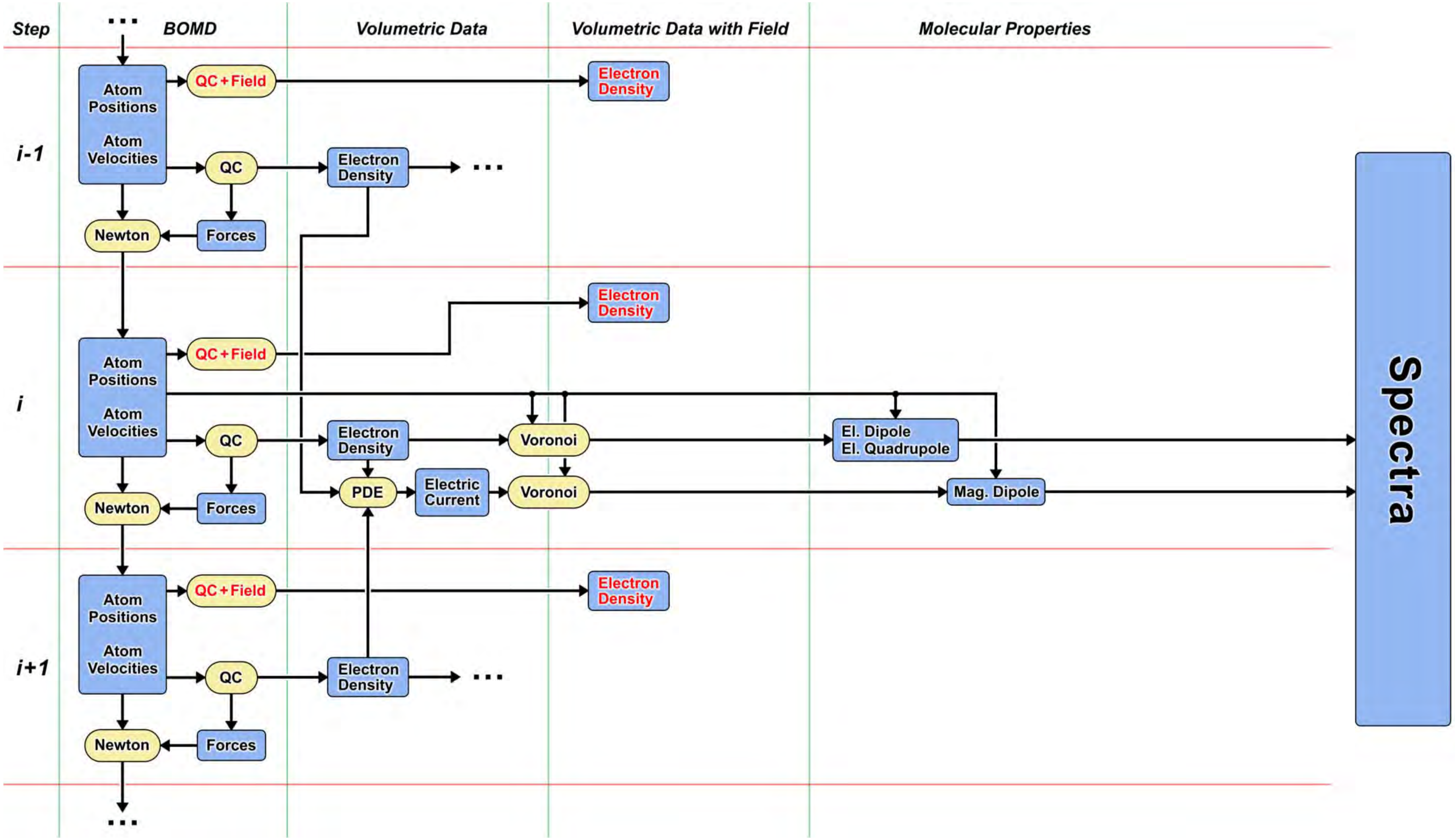
Our Approach to ROA



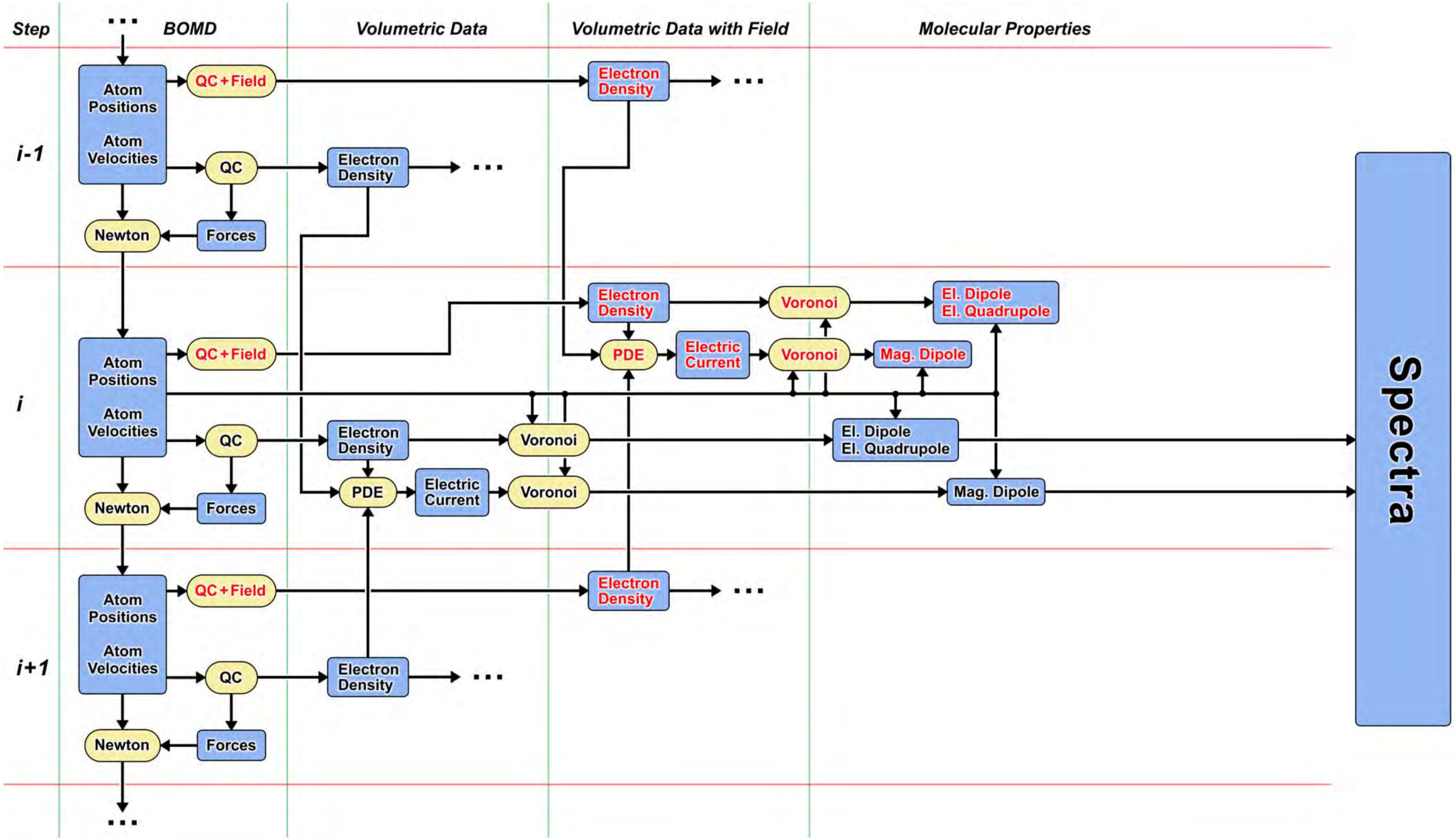
Our Approach to ROA



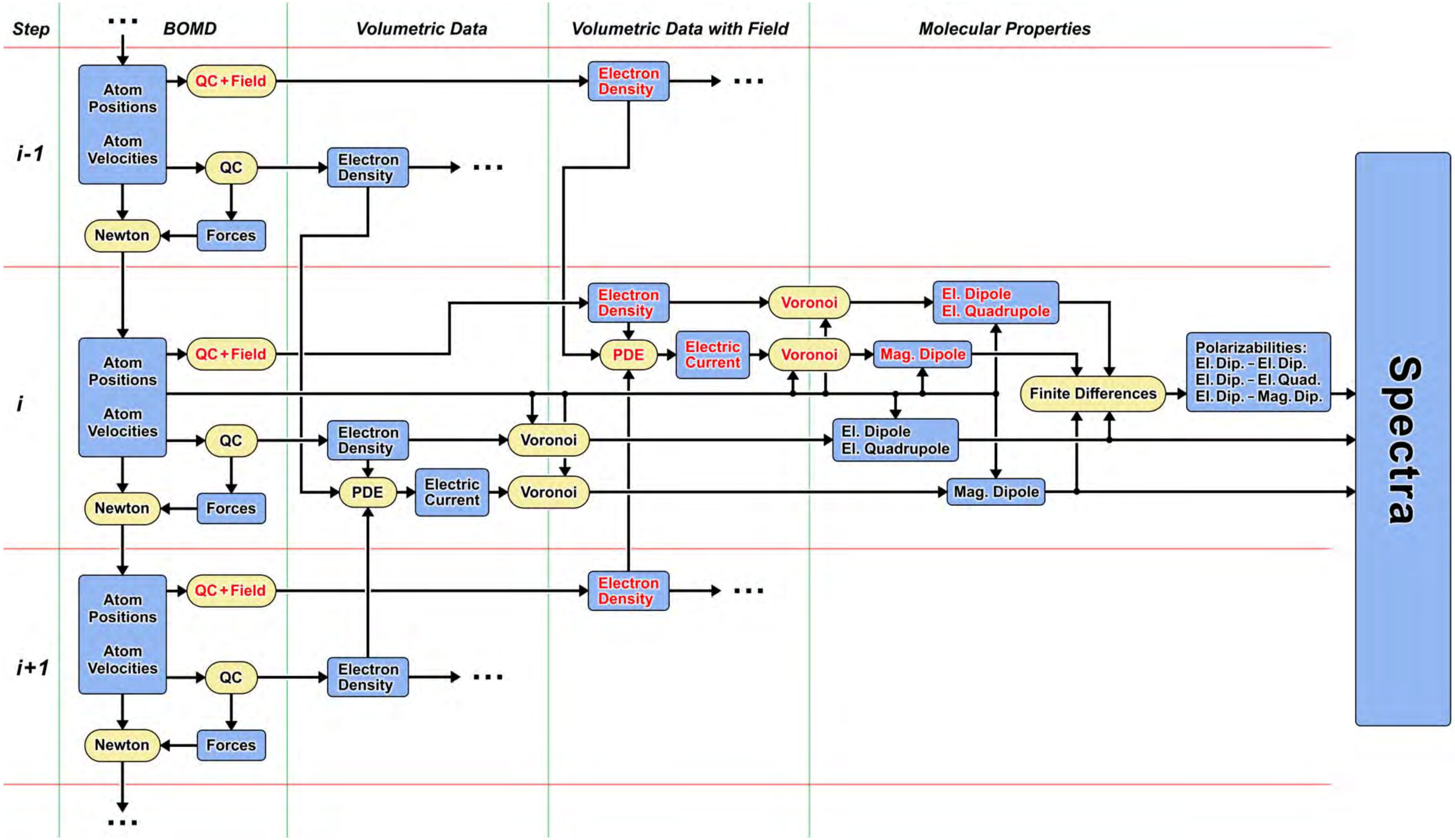
Our Approach to ROA



Our Approach to ROA



Our Approach to ROA



Our Approach to ROA

Molecular electric dipole / quadrupole from classical expressions:

$$\mathbf{p}^{\mathrm{Mol}} = \sum_{n=1}^{N_{\mathrm{Mol}}} q_n \mathbf{r}_n - \int_{\mathrm{Mol}} \rho(\mathbf{r})\,\mathbf{r}\,d^3\mathbf{r}$$

$$\mathbf{Q}_{ij}^{\mathrm{Mol}} = \sum_{n=1}^{N_{\mathrm{Mol}}} q_n \left(3\mathbf{r}_{n,i}\mathbf{r}_{n,j} - \|\mathbf{r}_n\|^2 \delta_{ij}\right) - \int_{\mathrm{Mol}} \rho(\mathbf{r}) \left(3\mathbf{r}_i\mathbf{r}_j - \|\mathbf{r}\|^2 \delta_{ij}\right) d^3\mathbf{r}$$

Electric current from purely classical PDE on grid.

Molecular magnetic dipole from classical expression:

$$\mathbf{m}^{\mathrm{Mol}} = \frac{1}{2}\sum_{n=1}^{N_{\mathrm{Mol}}} q_n \left(\mathbf{r}_n \times \mathbf{v}_n\right) - \frac{1}{2}\int_{\mathrm{Mol}} \mathbf{r} \times \mathbf{j}(\mathbf{r})\; d^3\mathbf{r}$$

We always use the molecular center of mass as coordinate origin.

Our Approach to ROA

Polarizabilities from finite differences (external electric field):

$$\alpha^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{p}^{\mathrm{Mol}}$$

Electric dipole – Electric dipole Polarizability

$$\mathcal{A}^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{Q}^{\mathrm{Mol}}$$

Electric dipole – Electric quadrupole Polarizability

$$\mathcal{G}'^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{m}^{\mathrm{Mol}}$$

Electric dipole – Magnetic dipole Polarizability

Our Approach to ROA

Polarizabilities from finite differences (external electric field):

$$\alpha^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{p}^{\mathrm{Mol}}$$ *Electric dipole – Electric dipole Polarizability*

$$\mathcal{A}^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{Q}^{\mathrm{Mol}}$$ *Electric dipole – Electric quadrupole Polarizability*

$$\mathcal{G}'^{\mathrm{Mol}} = \frac{d}{d\mathbf{E}}\mathbf{m}^{\mathrm{Mol}}$$ *Electric dipole – Magnetic dipole Polarizability*

Obtain the required polarizabilities:

$$A^{\mathrm{Mol}} = \mathcal{A}^{\mathrm{Mol}}$$ *Electric quadrupole – Electric dipole Polarizability*

$$G'^{\mathrm{Mol}} = -\left(\mathcal{G}'^{\mathrm{Mol}}\right)^{T}$$ *Magnetic dipole – Electric dipole Polarizability*

Our Approach to ROA

Compute the ROA invariants via FT of cross-correlations:

$$aG'(\tilde{\nu}) = 2\pi c\tilde{\nu}_{\text{in}} \int_{-\infty}^{\infty} \left\langle \frac{\dot{\alpha}_{xx}(\tau) + \dot{\alpha}_{yy}(\tau) + \dot{\alpha}_{zz}(\tau)}{3} \frac{G'_{xx}(\tau+t) + G'_{yy}(\tau+t) + G'_{zz}(\tau+t)}{3} \right\rangle_{\tau} \cdot \exp(-2\pi \mathrm{i} c\tilde{\nu}t)\,\mathrm{d}t$$

$$\begin{aligned}\gamma^2_{G'}(\tilde{\nu}) = 2\pi c\tilde{\nu}_{\text{in}} \int_{-\infty}^{\infty} \Bigg[&\frac{1}{2} \left\langle (\dot{\alpha}_{xx}(\tau) - \dot{\alpha}_{yy}(\tau)) \left(G'_{xx}(\tau+t) - G'_{yy}(\tau+t)\right) \right\rangle_{\tau} \\ &+ \frac{1}{2} \left\langle (\dot{\alpha}_{yy}(\tau) - \dot{\alpha}_{zz}(\tau)) \left(G'_{yy}(\tau+t) - G'_{zz}(\tau+t)\right) \right\rangle_{\tau} \\ &+ \frac{1}{2} \left\langle (\dot{\alpha}_{zz}(\tau) - \dot{\alpha}_{xx}(\tau)) \left(G'_{zz}(\tau+t) - G'_{xx}(\tau+t)\right) \right\rangle_{\tau} \\ &+ \frac{3}{2} \left\langle \dot{\alpha}_{xy}(\tau) \left(G'_{xy}(\tau+t) + G'_{yx}(\tau+t)\right) \right\rangle_{\tau} + \frac{3}{2} \left\langle \dot{\alpha}_{yz}(\tau) \left(G'_{yz}(\tau+t) + G'_{zy}(\tau+t)\right) \right\rangle_{\tau} \\ &+ \frac{3}{2} \left\langle \dot{\alpha}_{zx}(\tau) \left(G'_{zx}(\tau+t) + G'_{xz}(\tau+t)\right) \right\rangle_{\tau} \Bigg] \exp(-2\pi \mathrm{i} c\tilde{\nu}t)\,\mathrm{d}t\end{aligned}$$

$$\begin{aligned}\gamma^2_{A}(\tilde{\nu}) = \pi c\tilde{\nu}_{\text{in}} \int_{-\infty}^{\infty} \Bigg[&\left\langle (\dot{\alpha}_{yy}(\tau) - \dot{\alpha}_{xx}(\tau))\, \dot{A}_{z,xy}(\tau+t) \right\rangle_{\tau} \\ &+ \left\langle (\dot{\alpha}_{xx}(\tau) - \dot{\alpha}_{zz}(\tau))\, \dot{A}_{y,zx}(\tau+t) \right\rangle_{\tau} + \left\langle (\dot{\alpha}_{zz}(\tau) - \dot{\alpha}_{yy}(\tau))\, \dot{A}_{x,yz}(\tau+t) \right\rangle_{\tau} \\ &+ \left\langle \dot{\alpha}_{xy}(\tau) \left(\dot{A}_{y,yz}(\tau+t) - \dot{A}_{z,yy}(\tau+t) + \dot{A}_{z,xx}(\tau+t) - \dot{A}_{x,xz}(\tau+t) \right) \right\rangle_{\tau} \\ &+ \left\langle \dot{\alpha}_{yz}(\tau) \left(\dot{A}_{z,zx}(\tau+t) - \dot{A}_{x,zz}(\tau+t) + \dot{A}_{x,yy}(\tau+t) - \dot{A}_{y,yx}(\tau+t) \right) \right\rangle_{\tau} \\ &+ \left\langle \dot{\alpha}_{zx}(\tau) \left(\dot{A}_{y,zz}(\tau+t) - \dot{A}_{z,zy}(\tau+t) + \dot{A}_{x,xy}(\tau+t) - \dot{A}_{y,xx}(\tau+t) \right) \right\rangle_{\tau} \Bigg] \\ &\cdot \exp(-2\pi \mathrm{i} c\tilde{\nu}t)\,\mathrm{d}t\end{aligned}$$

Our Approach to ROA

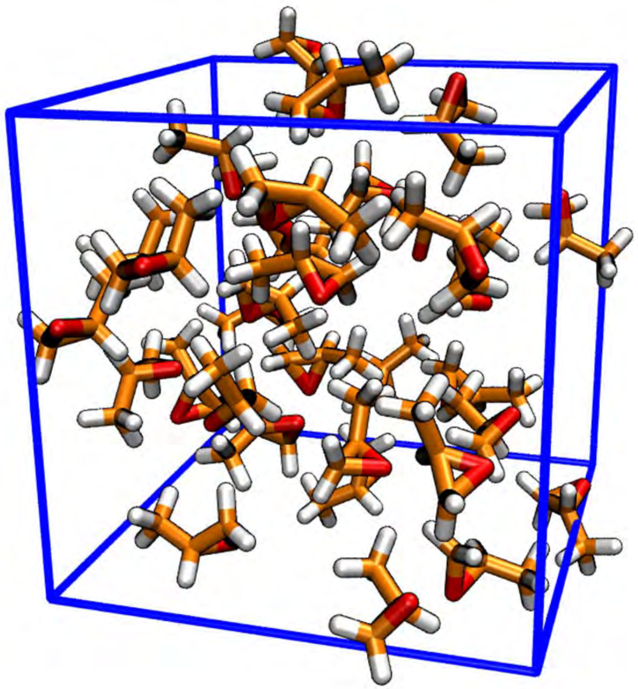
Assemble the ROA spectrum as linear combination of invariants:

$$\Delta I(\tilde{\nu}) = \frac{h}{8\varepsilon_0^2 c k_{\mathrm{B}} T} \cdot \frac{(\tilde{\nu}_{\mathrm{in}} - \tilde{\nu})^4}{\tilde{\nu}\left(1 - \exp\left(-\frac{hc\tilde{\nu}}{k_{\mathrm{B}}T}\right)\right)} \cdot \frac{1}{90}\left(X \cdot aG'(\tilde{\nu}) + Y \cdot \gamma_{G'}^2(\tilde{\nu}) + Z \cdot \gamma_A^2(\tilde{\nu})\right)$$

Coefficient values from literature:

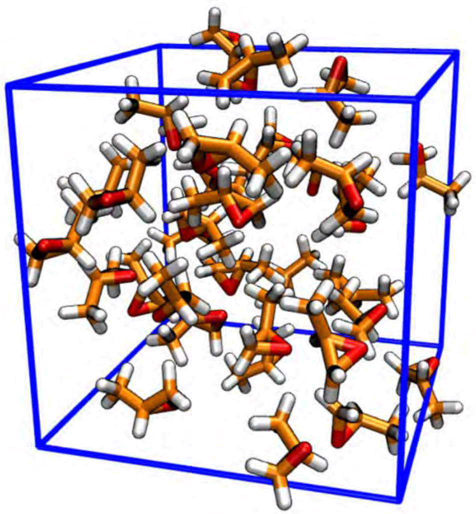
Scattering angle	Polarization	X	Y	Z
0°	$\Delta I^{\perp} = \Delta I^{\parallel}$	360	8	-8
0°	ΔI	720	16	-16
90°	$\Delta I^{\perp}$	180	28	4
90°	$\Delta I^{\parallel}$	0	24	-8
90°	ΔI	180	52	-4
180°	$\Delta I^{\perp} = \Delta I^{\parallel}$	0	48	16
180°	ΔI	0	96	32

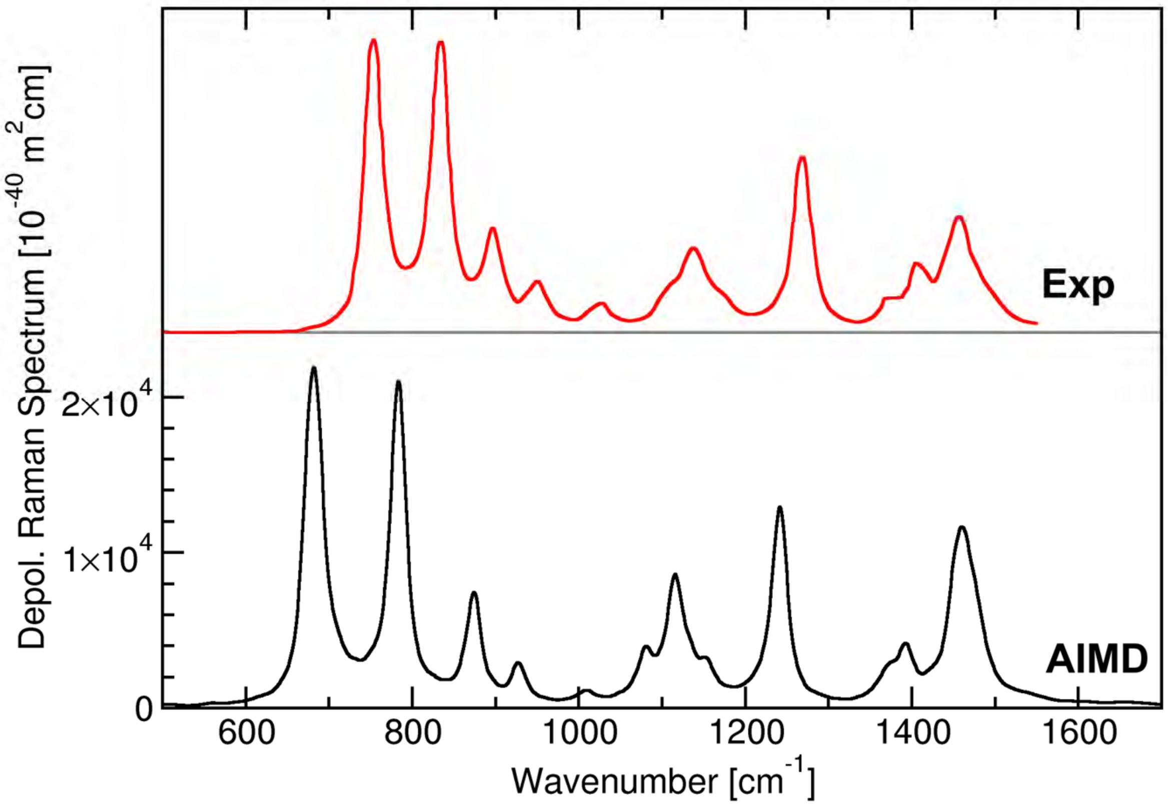
Application



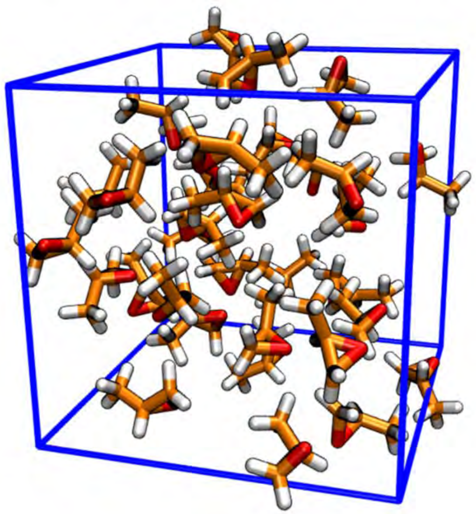
Liquid *(R)*-propylene oxide
32 molecules, 300 K
Standard AIMD (BLYP-D3)
65 000 steps (32.5 ps) production run

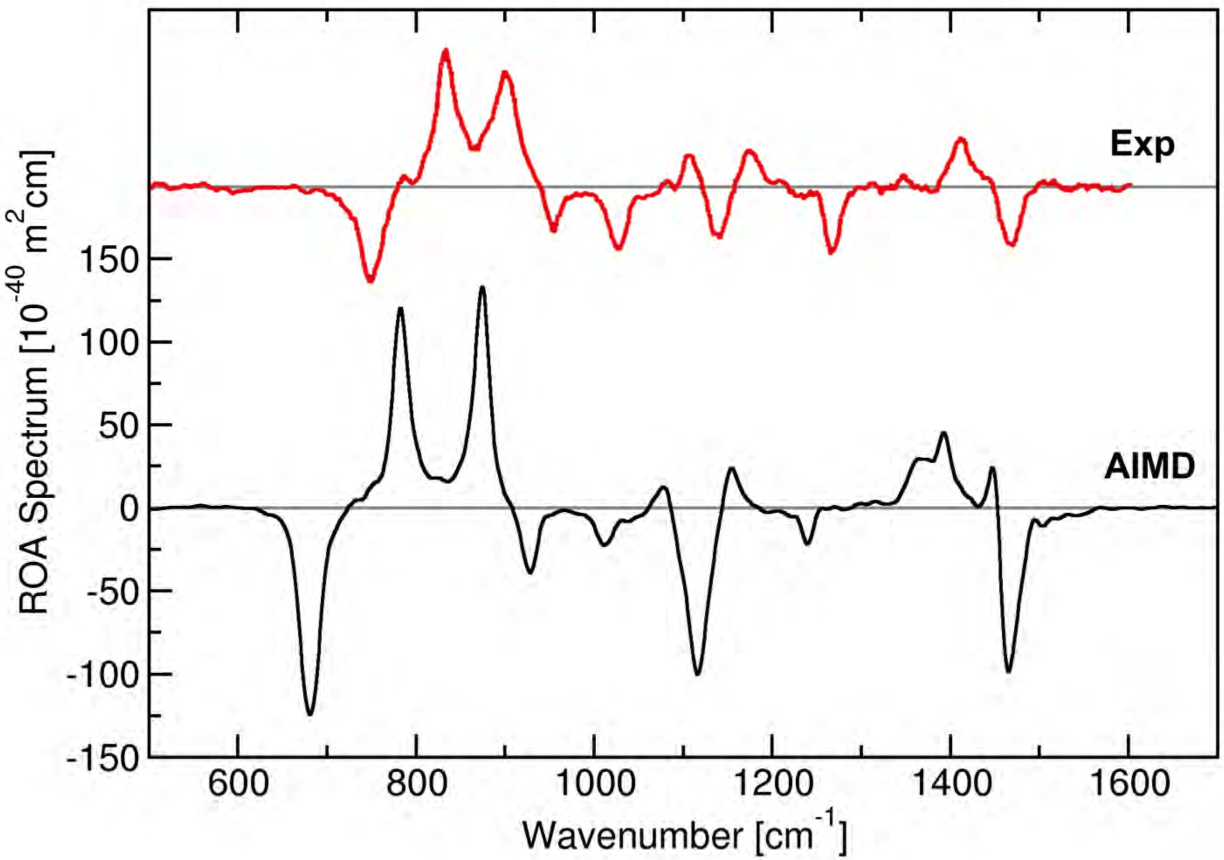
Raman Spectrum





ROA Spectrum





The goal is completely achieved ☺

First predicted bulk phase ROA spectrum.

Published in Summer 2017:

The Journal of Physical Chemistry Letters

Letter

pubs.acs.org/JPCL

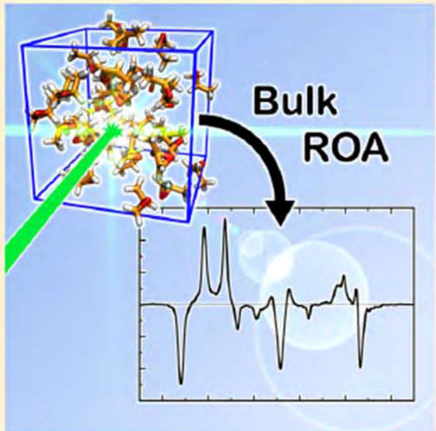
Computing Bulk Phase Raman Optical Activity Spectra from *ab initio* Molecular Dynamics Simulations

Martin Brehm* and Martin Thomas

Institut für Chemie - Theoretische Chemie, Martin-Luther-Universität Halle-Wittenberg, Von-Danckelmann-Platz 4, 06120 Halle (Saale), Germany

Supporting Information

ABSTRACT: We present our novel methodology for computing Raman optical activity (ROA) spectra of liquid systems from *ab initio* molecular dynamics (AIMD) simulations. The method is built upon the recent developments to obtain magnetic dipole moments from AIMD and to integrate molecular properties by using radical Voronoi tessellation. These techniques are used to calculate optical activity tensors for large and complex periodic bulk phase systems. Only AIMD simulations are required as input, and no time-consuming perturbation theory is involved. The approach relies only on the total electron density in each time step and can readily be combined with a wide range of electronic structure methods. To the best of our knowledge, these are the first computed ROA spectra for a periodic bulk phase system. As an example, the experimental ROA spectrum of liquid (*R*)-propylene oxide is reproduced very well.



M. Brehm, M. Thomas: „Computing Bulk Phase Raman Optical Activity Spectra from *ab initio* Molecular Dynamics Simulations“ , *J. Phys. Chem. Lett.* **2017**, *8 (14)*, pp 3409-3414.

Full Set of Vibrational Spectra

Now the full set of vibrational spectra (IR, Raman, VCD, ROA) of a bulk phase system can be computed in one go with CP2k and TRAVIS.

Power Spectra

- Also known as vibrational density of states or phonon density of states
- Can be simply computed from any standard MD as the FT of the velocity autocorrelation function
- Contains all spectral modes of the system, no matter if they are active in IR / Raman / etc.
- A mode in IR / Raman / VCD / ROA can only appear at positions where a band in the power spectrum is

→ Power spectrum contains all band frequency information; only the intensities are determined by the different methods

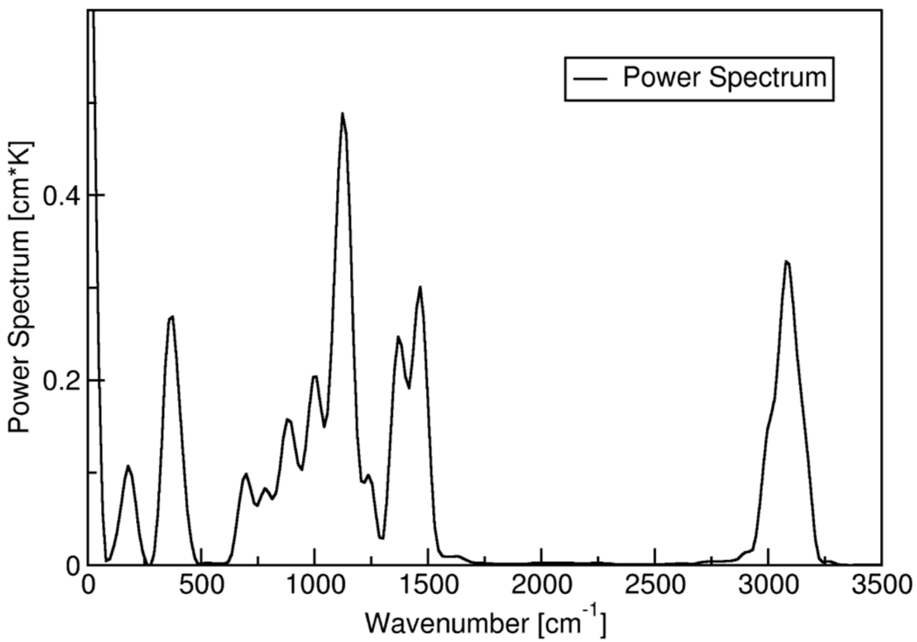
Power Spectra

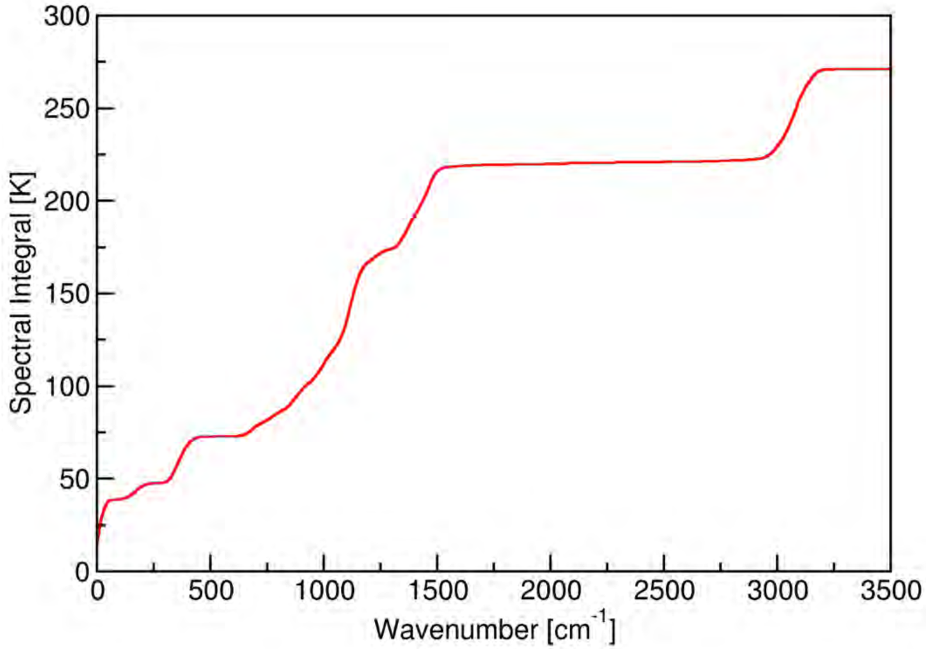
- Nice feature: If the unit of the power spectrum is cm * K (such as in TRAVIS), the integral is in Kelvin

→ Can derermine the „temperature inside some mode“ by simply integrating over the corresponding band

This can help checking the equipartition theorem (i.e. if the trajectory is well equilibrated or not)

Power Spectra





Separation of Frequencies and Intensities

Possibility to compute the trajectory with one method, and then compute the electron densities with some other method

→ Much flexibility

You could, e.g., compute a MP2 dynamics trajectory, and then compute electron densities with DFT

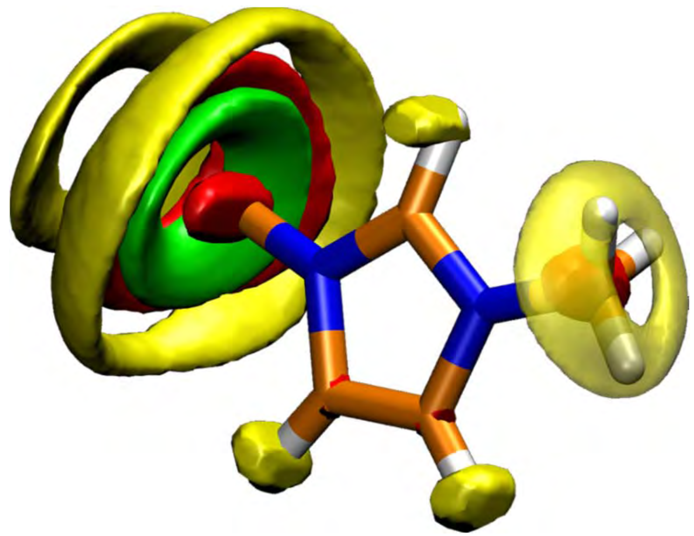
→ Correct band positions (MP2 dynamics) together with DFT band intensities

Vice versa is also possible

Available in TRAVIS

The methods presented here are all implemented
in our freeware program package TRAVIS.

http://www.travis-analyzer.de/



You can start computing bulk phase ROA spectra today ☺

Computational Resources

Used core hours (ch) on Intel Xeon “Haswell”@2.4 GHz:

2 350 ch for equilibration
25 500 ch for AIMD (incl. 3 external field directions)
2 000 ch to solve the PDE

→ Approx 30 000 ch in total.

Takes ≈ 3 weeks on a „small“
server with 64 cores.

Computational Resources

Used core hours (ch) on Intel Xeon “Haswell”@2.4 GHz:

2 350 ch for equilibration
25 500 ch for AIMD (incl. 3 external field directions)
2 000 ch to solve the PDE

→ Approx 30 000 ch in total.

Takes ≈ 3 weeks on a „small“ server with 64 cores.

Available for < 10 000 € today.

Computational Resources

What about the disk storage?

Electron density grid is 160 × 160 × 160 in our case.

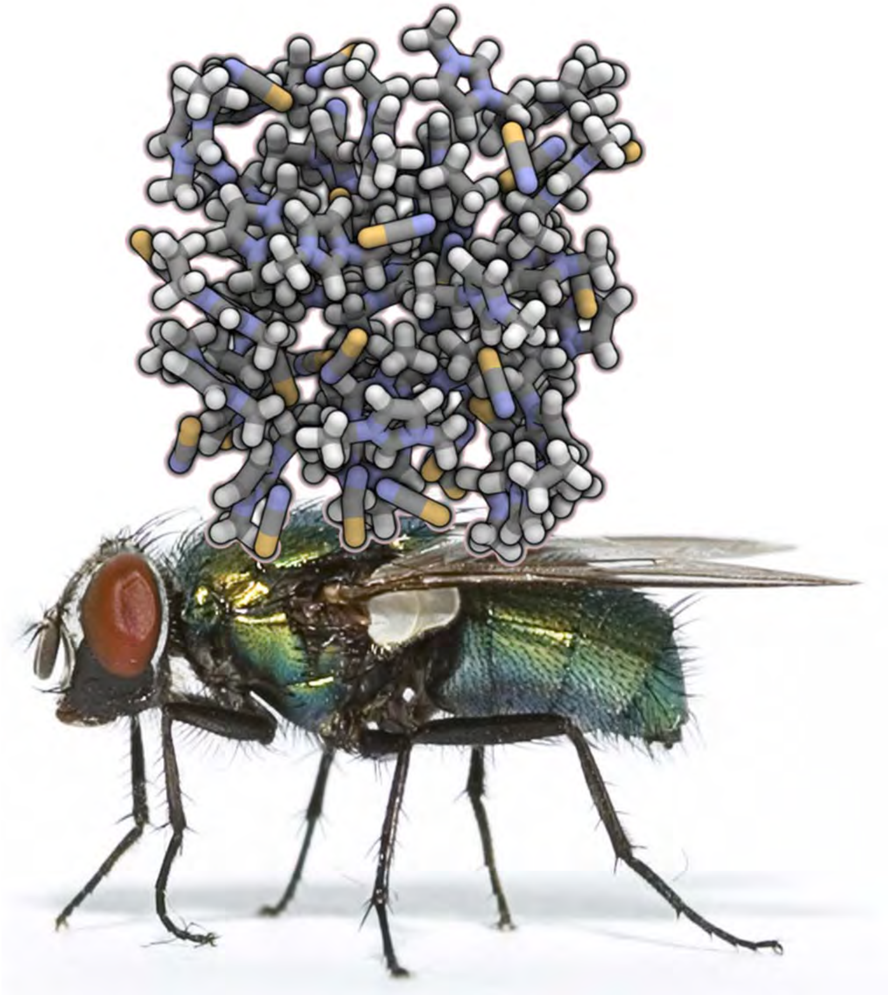
→ 4 million data points per time step.

CUBE file is 52 MiB per frame.

We require 4 × 65 000 frames.

→ 13 Terabyte of data for one ROA spectrum 

On-the-fly Processing



→ No need to store full electron density trajectory.

Computational Resources

Computation can be done „on the fly“:

CP2k writes CUBE files; TRAVIS reads and deletes them.

No more disk space requirements at all.

But: Very bad during development phase of code…

Computational Resources

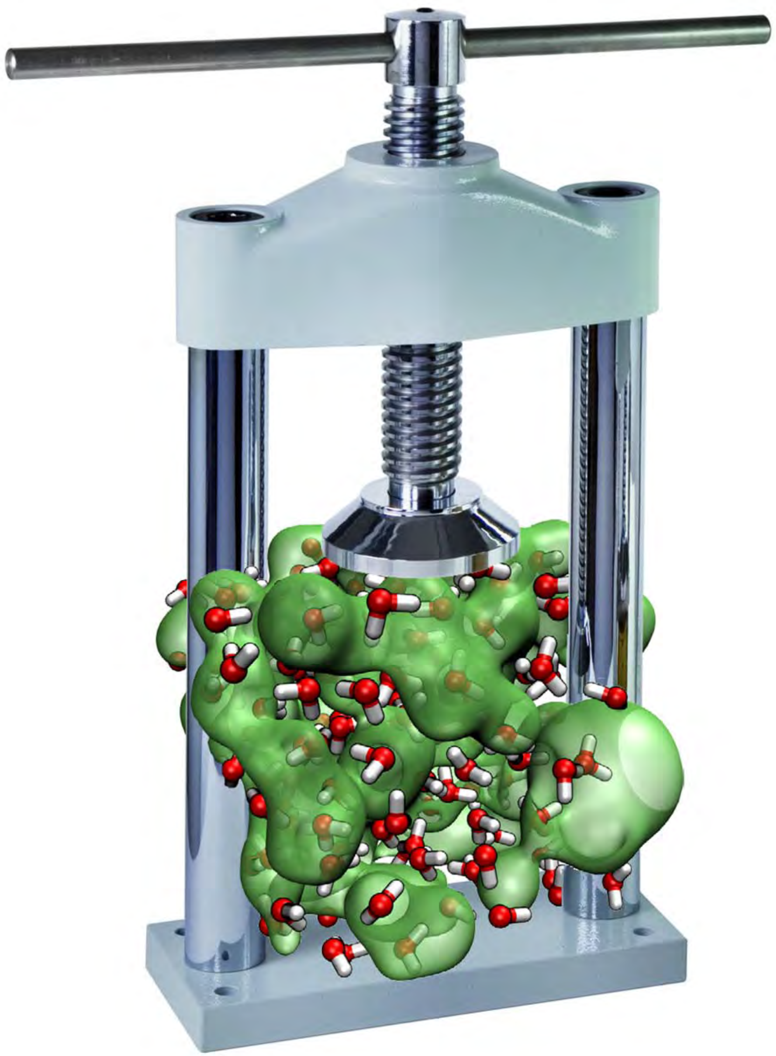
What about *bzip2*-ing the CUBE files?

Compression ratio is around 4.5 : 1

→ Still 3 Terabyte; **very** slow to compress…

Compressing the CUBE files would take longer than the AIMD itself

3. Compression of Volumetric Data

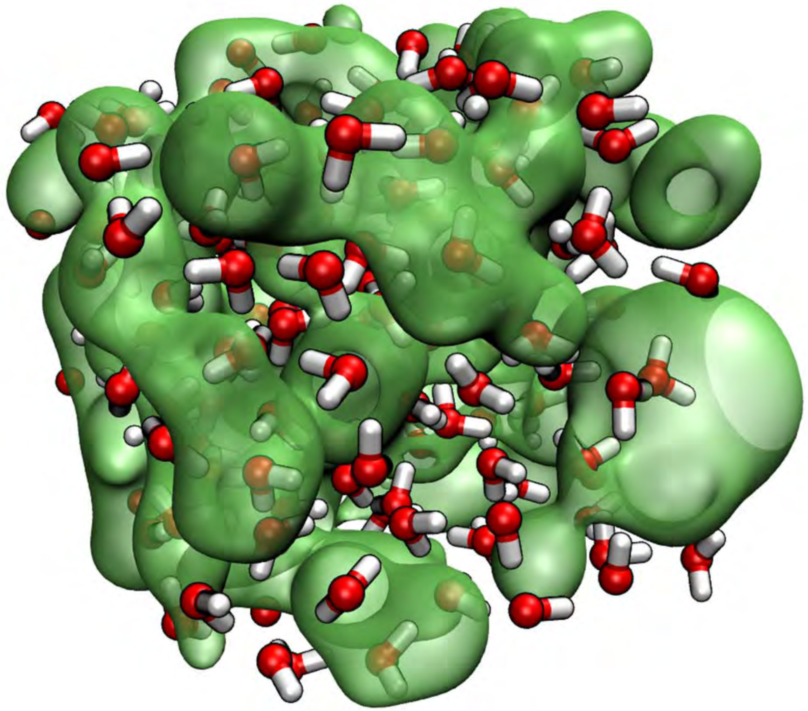


What are Volumetric Data Trajectories?

Answer: A consecutive sequence of 3D cartesian grid frames with real numbers

Examples:

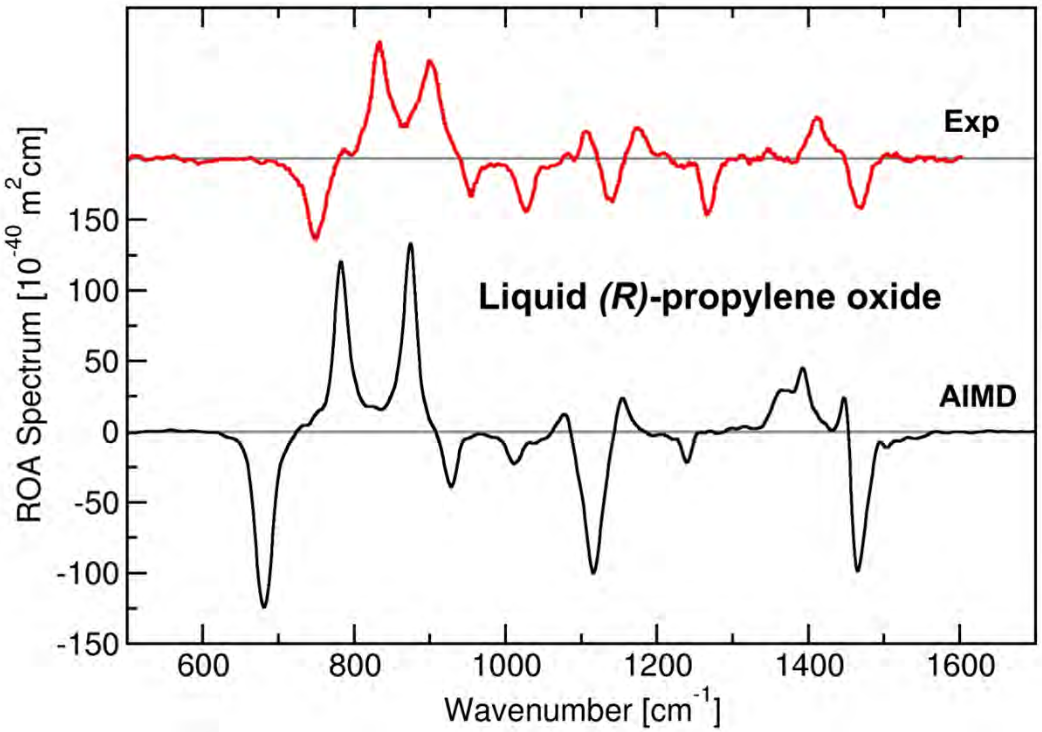
- Electron Density
- Molecular Orbitals
- Electrostatic Potential



Who needs Volumetric Data Trajectories?

Typical Applications:

- Partial Charges from Electron Density
- Electric Dipole / Quadrupole Moments
- Vibrational Spectroscopy (IR, Raman, VCD, ROA)



How are Volumetric Data Trajectories stored?

„Standard“ (since ≈ 40 years)**:** **<u>Gaussian Cube Files</u>**

- Simple text file format
- Used by many programs (Gaussian, Orca, CP2k, CPMD, TurboMole, etc.)
- Trajectory is simple concatenation of single frames
- Typical resolution:
  100 × 100 × 100
  to
  300 × 300 × 300

```
-Quickstep-
 ELECTRON DENSITY
  320    0.000000    0.000000    0.000000
  160    0.184053    0.000000    0.000000
  160    0.000000    0.184053    0.000000
  160    0.000000    0.000000    0.184053
    6    0.000000   20.102072   20.046646    5.104961
    6    0.000000   21.818315   19.569134    3.053013
    6    0.000000   18.173905   22.125253    4.939110
    1    0.000000   20.811234   19.834254    7.084541
...
    1    0.000000    8.440611    6.073738    9.474705
    1    0.000000    7.949889    7.549963   12.643419
    8    0.000000    8.789560   11.402496    7.658100
  0.79844E-02  0.75825E-02  0.69447E-02  0.61531E-02  0.52995E-02  0.44648E-02
  0.37078E-02  0.30622E-02  0.25410E-02  0.21434E-02  0.18602E-02  0.16791E-02
  0.15863E-02  0.15679E-02  0.16100E-02  0.16988E-02  0.18205E-02  0.19611E-02
  0.21068E-02  0.22444E-02  0.23621E-02  0.24505E-02  0.25035E-02  0.25189E-02
  0.24984E-02  0.24468E-02  0.23716E-02  0.22808E-02  0.21822E-02  0.20823E-02
  0.19854E-02  0.18937E-02  0.18070E-02  0.17239E-02  0.16422E-02  0.15594E-02
  0.14735E-02  0.13837E-02  0.12905E-02  0.11961E-02  0.11036E-02  0.10174E-02
  0.94286E-03  0.88572E-03  0.85243E-03  0.84999E-03  0.88608E-03  0.96908E-03
  0.11082E-02  0.13135E-02  0.15953E-02  0.19631E-02  0.24228E-02  0.29723E-02
  0.35962E-02  0.42610E-02  0.49139E-02  0.54874E-02  0.59111E-02  0.61287E-02
  0.61145E-02  0.58805E-02  0.54740E-02  0.49642E-02  0.44264E-02  0.39296E-02
```

What is the Problem?

Consider the Bulk ROA spectrum:

- Grid of 160 x 160 x 160 points *(that‘s still small!)*
- 65 000 time steps
- 4 trajectories for polarizability
  (without field and X, Y, Z field direction)

One Cube frame is 52 MiB

$52\,\text{MiB} \times 65\,000 \times 4 = 13\,\text{TiB}$ data for **one** spectrum

What is the Problem?

What about compression tools (bzip2)?

Can achieve ratio of ≈ 4.5 : 1 → still 3 TiB…

Compression time is longer than AIMD simulation 

→ We needed a better solution.

Two Kinds of Compression Algorithms

Generic Algorithms

- Take *any* input data
- Always *lossless*
- Don‘t have any *a priori* knowledge on data structuring
- Examples: zip, gz, bz2

Specialized Algorithms

- Very specific input data
- Exploit known structure of data (*heuristic*)
- Examples:
  mp3, jpg *(lossy)*
  ogg, png *(lossless)*

→ We need a specialized algorithm for volumetric data.

Volumetric Data Structure

What we need:

- Efficient **lossless** compression algorithm *(to the accuracy of the input data)*

What we know:

- Volumetric data is spatially continuous (smooth, *no sharp edges*)

  → can be exploited

- If from a MD simulation, data is also temporally continuous (*no abrupt jumps*)

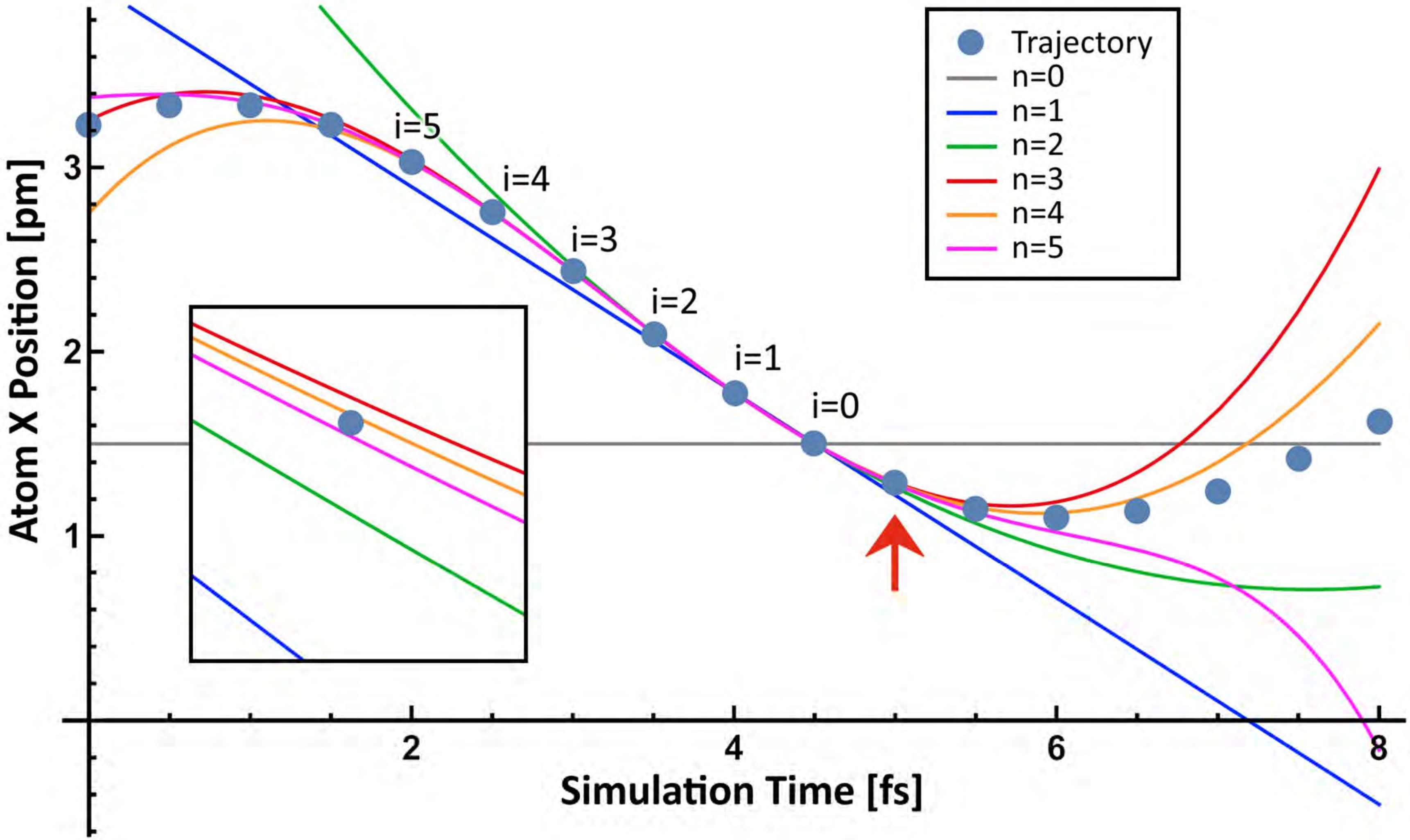
  → can be exploited

General Idea

- Before processing each bin, estimate *(extrapolate)* its value based on earlier values *(both space and time)*
- Then look at the true value and store the deviation from the estimate
- Finite precision: Deviation can be stored as integer numbers
- This converts the input grid of real values into a stream of integers
- **If the extrapolation is good, the integers are small in value *(reduction of information entropy!)***
- Finally: Compress integer stream by entropy encoding

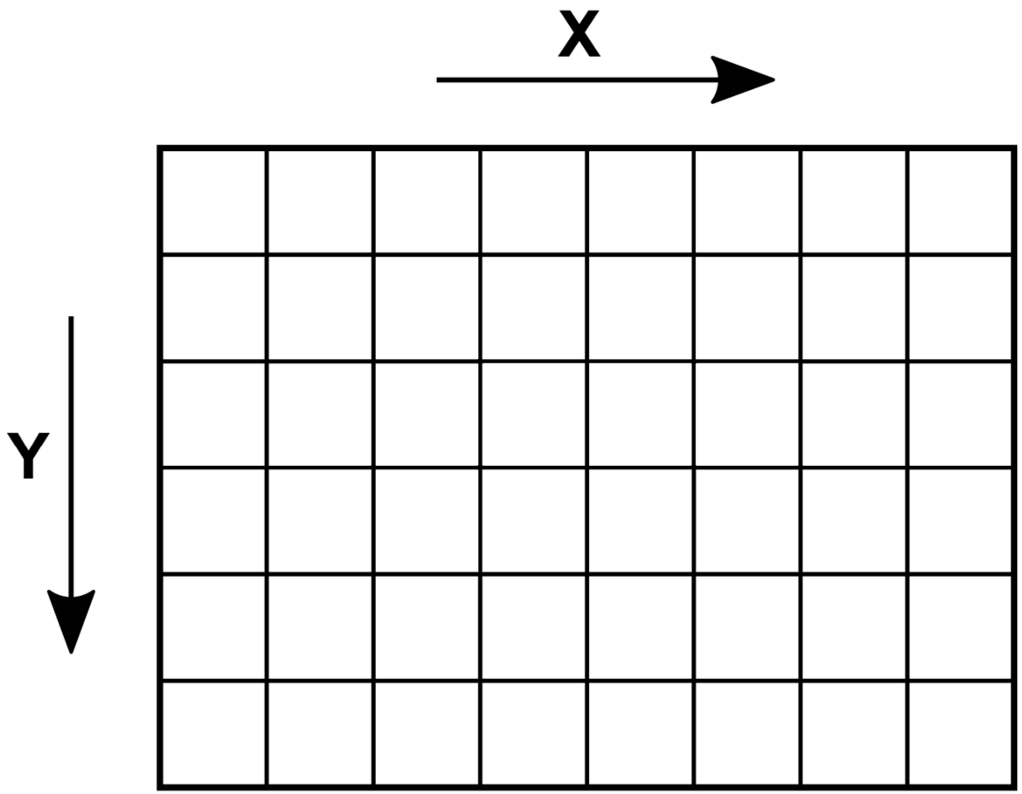
Polynomial Extrapolation

Is easy in 1D:



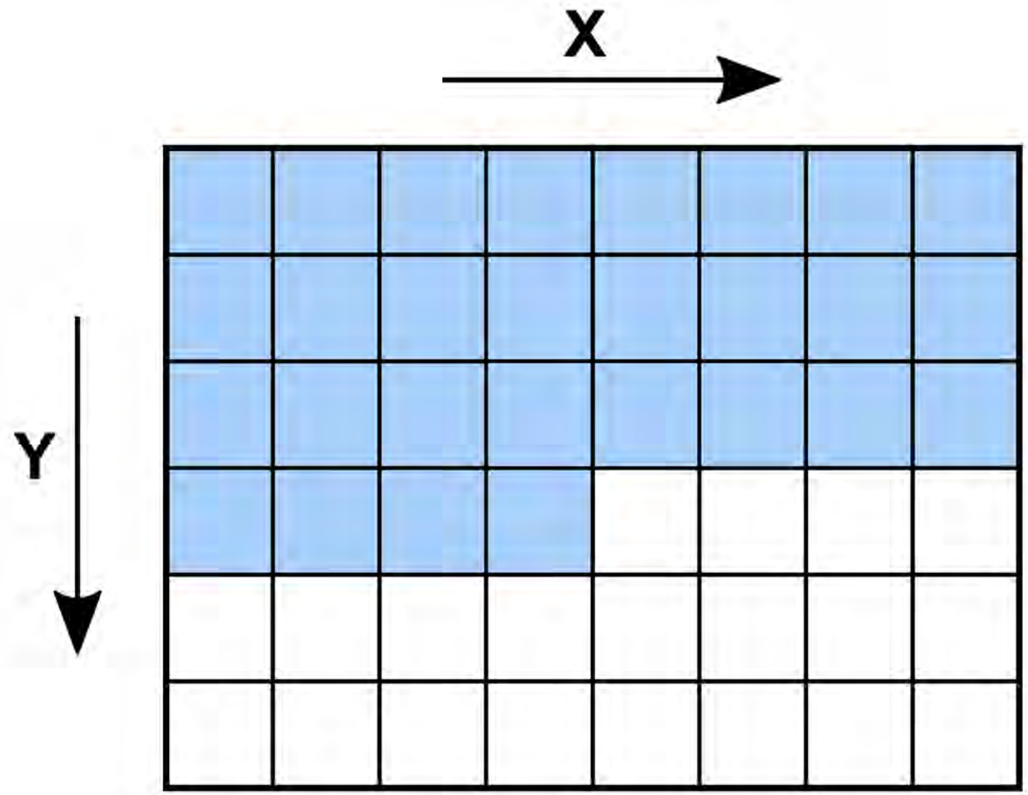
Polynomial Extrapolation

But how to proceed in more dimensions?



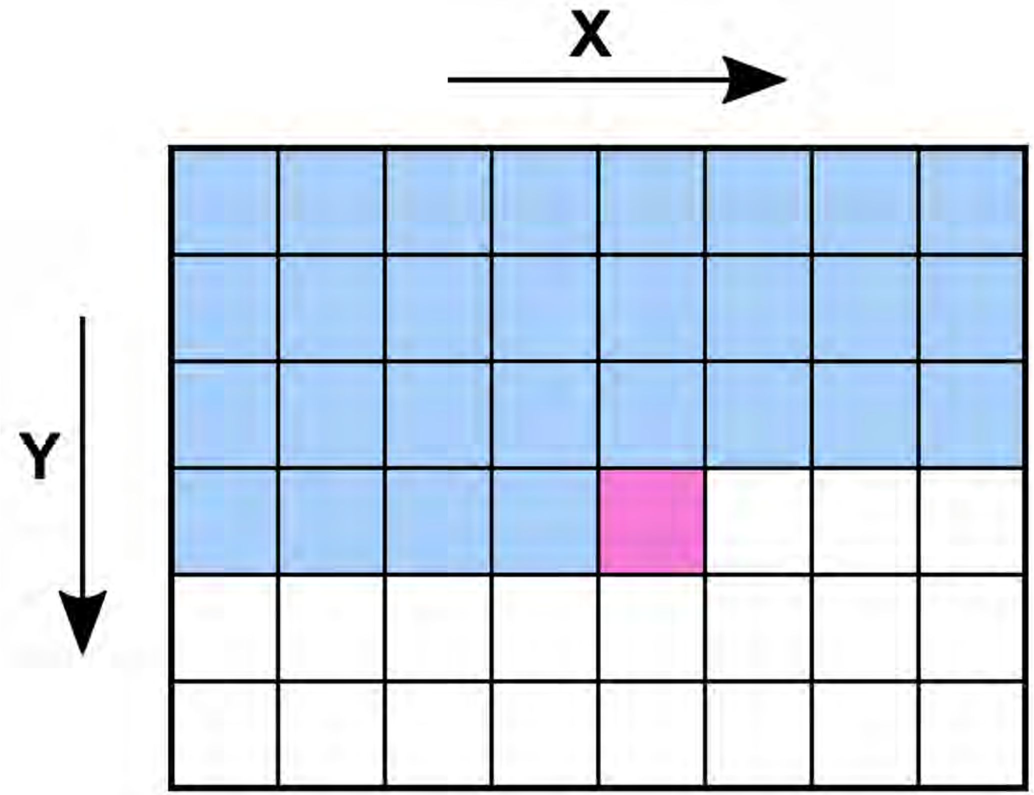
Polynomial Extrapolation

But how to proceed in more dimensions?



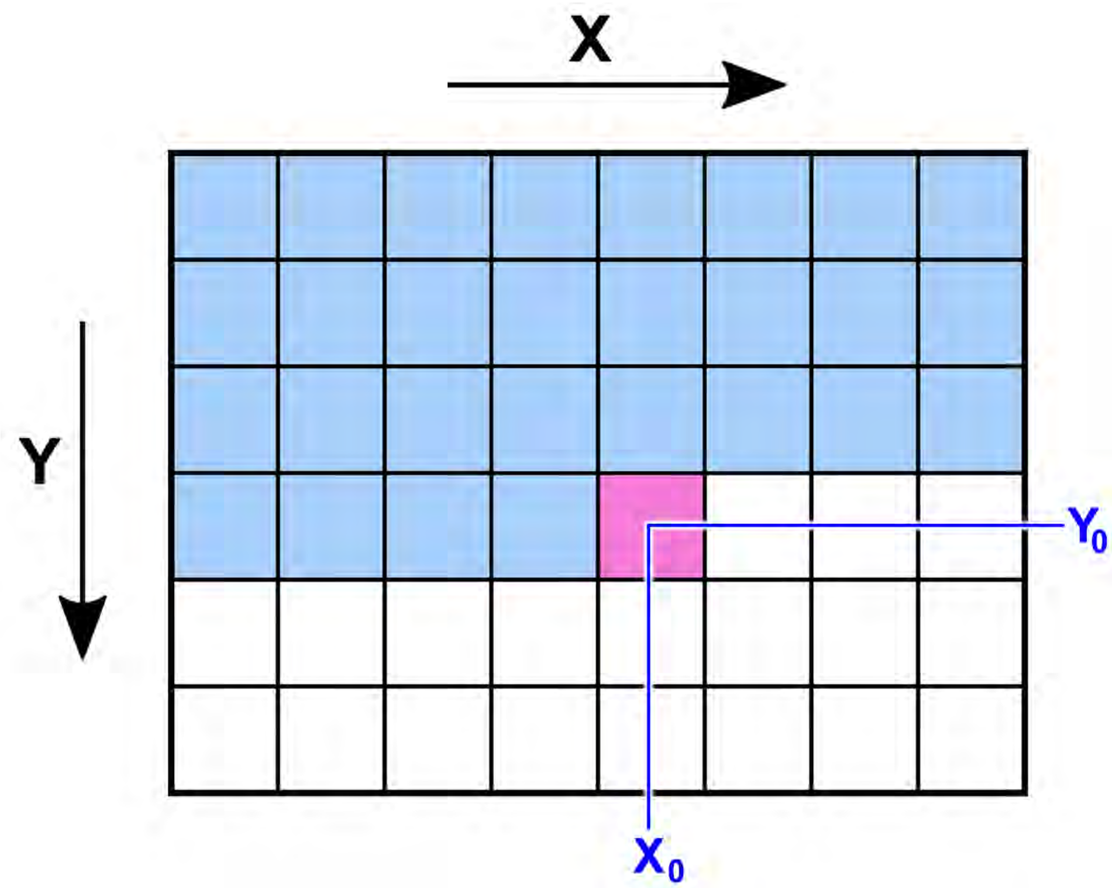
Polynomial Extrapolation

But how to proceed in more dimensions?



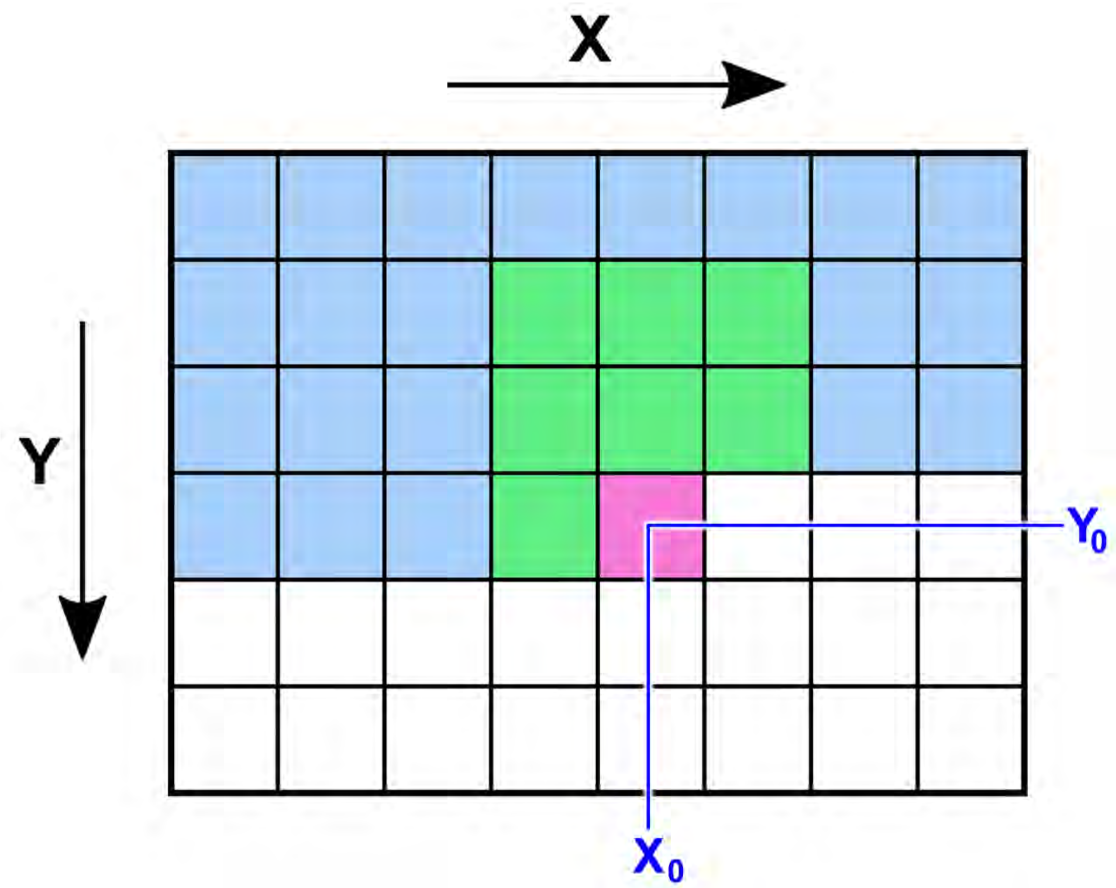
Polynomial Extrapolation

But how to proceed in more dimensions?



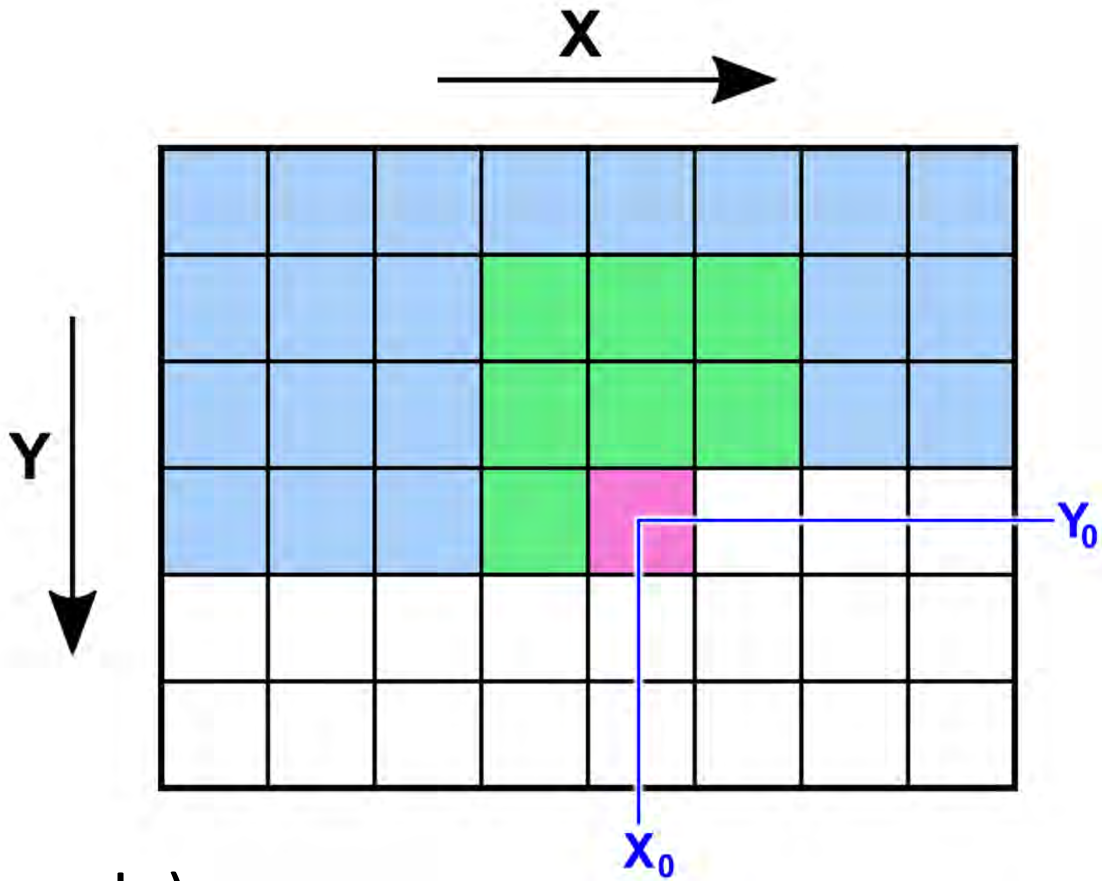
Polynomial Extrapolation

But how to proceed in more dimensions?



Polynomial Extrapolation

But how to proceed in more dimensions?



Ansatz (example):

$$F(x, y) = c_0 + c_x \cdot x + c_y \cdot y + c_{xy} \cdot xy + c_{x^2} \cdot x^2$$

$$F\left(x,y\right) = c_0 + c_x \cdot x + c_y \cdot y + c_{xy} \cdot xy + c_{x^2} \cdot x^2$$

→ Linear System of Equations (7 equations, 5 unknowns):

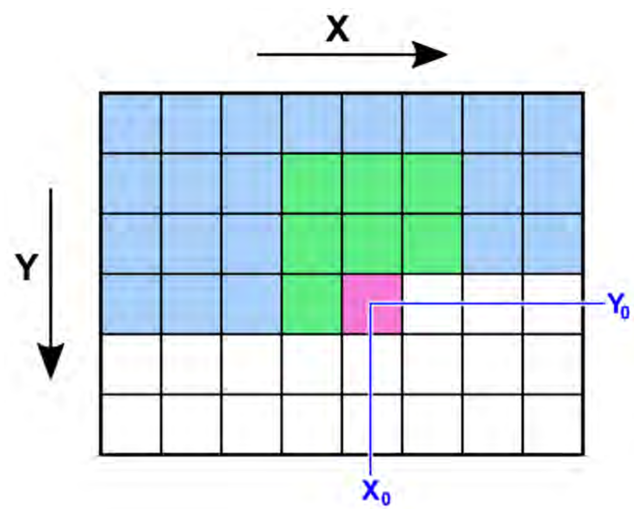
$$\begin{aligned}
F(x_0-1,y_0-2) &= c_0 + c_x(x_0-1) + c_y(y_0-2) + c_{xy}(x_0-1)(y_0-2) + c_{x^2}(x_0-1)^2\\
F(x_0, \quad y_0-2) &= c_0 + c_x\ x_0 \quad + c_y(y_0-2) + c_{xy}\ x_0 \quad (y_0-2) + c_{x^2}\ x_0^2\\
F(x_0+1,y_0-2) &= c_0 + c_x(x_0+1) + c_y(y_0-2) + c_{xy}(x_0+1)(y_0-2) + c_{x^2}(x_0+1)^2\\
F(x_0-1,y_0-1) &= c_0 + c_x(x_0-1) + c_y(y_0-1) + c_{xy}(x_0-1)(y_0-1) + c_{x^2}(x_0-1)^2\\
F(x_0, \quad y_0-1) &= c_0 + c_x\ x_0 \quad + c_y(y_0-1) + c_{xy}\ x_0 \quad (y_0-1) + c_{x^2}\ x_0^2\\
F(x_0+1,y_0-1) &= c_0 + c_x(x_0+1) + c_y(y_0-1) + c_{xy}(x_0+1)(y_0-1) + c_{x^2}(x_0+1)^2\\
F(x_0-1,y_0) \quad &= c_0 + c_x(x_0-1) + c_y\ y_0 \quad + c_{xy}(x_0-1)\ y_0 \quad + c_{x^2}(x_0-1)^2
\end{aligned}$$

System is **over-determined**.

→ no exact solution, only least squares.

How to solve it? Bring it in matrix form!

$$A \cdot \mathbf{x} = \mathbf{b}$$



$$F(x,y) = c_0 + c_x \cdot x + c_y \cdot y + c_{xy} \cdot xy + c_{x^2} \cdot x^2$$

→ Linear System of Equations (7 equations, 5 unknowns):

$$\begin{aligned}
F(x_0-1,y_0-2) &= c_0 + c_x(x_0-1) + c_y(y_0-2) + c_{xy}(x_0-1)(y_0-2) + c_{x^2}(x_0-1)^2 \\
F(x_0, \quad y_0-2) &= c_0 + c_x\, x_0 + c_y(y_0-2) + c_{xy}\, x_0 (y_0-2) + c_{x^2}\, x_0^2 \\
F(x_0+1,y_0-2) &= c_0 + c_x(x_0+1) + c_y(y_0-2) + c_{xy}(x_0+1)(y_0-2) + c_{x^2}(x_0+1)^2 \\
F(x_0-1,y_0-1) &= c_0 + c_x(x_0-1) + c_y(y_0-1) + c_{xy}(x_0-1)(y_0-1) + c_{x^2}(x_0-1)^2 \\
F(x_0, \quad y_0-1) &= c_0 + c_x\, x_0 + c_y(y_0-1) + c_{xy}\, x_0 (y_0-1) + c_{x^2}\, x_0^2 \\
F(x_0+1,y_0-1) &= c_0 + c_x(x_0+1) + c_y(y_0-1) + c_{xy}(x_0+1)(y_0-1) + c_{x^2}(x_0+1)^2 \\
F(x_0-1,y_0) &= c_0 + c_x(x_0-1) + c_y\, y_0 + c_{xy}(x_0-1)\, y_0 + c_{x^2}(x_0-1)^2
\end{aligned}$$

Matrix Form:

$$\begin{pmatrix}
1 & (x_0-1) & (y_0-2) & (x_0-1)(y_0-2) & (x_0-1)^2 \\
1 & x_0 & (y_0-2) & x_0(y_0-2) & x_0^2 \\
1 & (x_0+1) & (y_0-2) & (x_0+1)(y_0-2) & (x_0+1)^2 \\
1 & (x_0-1) & (y_0-1) & (x_0-1)(y_0-1) & (x_0-1)^2 \\
1 & x_0 & (y_0-1) & x_0(y_0-1) & x_0^2 \\
1 & (x_0+1) & (y_0-1) & (x_0+1)(y_0-1) & (x_0+1)^2 \\
1 & (x_0-1) & y_0 & (x_0-1)y_0 & (x_0-1)^2
\end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix}
F(x_0-1, y_0-2) \\
F(x_0, y_0-2) \\
F(x_0+1, y_0-2) \\
F(x_0-1, y_0-1) \\
F(x_0, y_0-1) \\
F(x_0+1, y_0-1) \\
F(x_0-1, y_0)
\end{pmatrix}$$

Matrix Form:

$$\begin{pmatrix} 1 & (x_0-1) & (y_0-2) & (x_0-1)(y_0-2) & (x_0-1)^2 \\ 1 & x_0 & (y_0-2) & x_0(y_0-2) & x_0^2 \\ 1 & (x_0+1) & (y_0-2) & (x_0+1)(y_0-2) & (x_0+1)^2 \\ 1 & (x_0-1) & (y_0-1) & (x_0-1)(y_0-1) & (x_0-1)^2 \\ 1 & x_0 & (y_0-1) & x_0(y_0-1) & x_0^2 \\ 1 & (x_0+1) & (y_0-1) & (x_0+1)(y_0-1) & (x_0+1)^2 \\ 1 & (x_0-1) & y_0 & (x_0-1)y_0 & (x_0-1)^2 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(x_0-1, y_0-2) \\ F(x_0, y_0-2) \\ F(x_0+1, y_0-2) \\ F(x_0-1, y_0-1) \\ F(x_0, y_0-1) \\ F(x_0+1, y_0-1) \\ F(x_0-1, y_0) \end{pmatrix}$$

We define $x_0 = 0, \quad y_0 = 0$

$$\begin{pmatrix} 1 & -1 & -2 & 2 & 1 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 1 & -2 & -2 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(-1, -2) \\ F(0, -2) \\ F(1, -2) \\ F(-1, -1) \\ F(0, -1) \\ F(1, -1) \\ F(-1, 0) \end{pmatrix}$$

Matrix Form:

$$\begin{pmatrix} 1 & (x_0-1) & (y_0-2) & (x_0-1)(y_0-2) & (x_0-1)^2 \\ 1 & x_0 & (y_0-2) & x_0(y_0-2) & x_0^2 \\ 1 & (x_0+1) & (y_0-2) & (x_0+1)(y_0-2) & (x_0+1)^2 \\ 1 & (x_0-1) & (y_0-1) & (x_0-1)(y_0-1) & (x_0-1)^2 \\ 1 & x_0 & (y_0-1) & x_0(y_0-1) & x_0^2 \\ 1 & (x_0+1) & (y_0-1) & (x_0+1)(y_0-1) & (x_0+1)^2 \\ 1 & (x_0-1) & y_0 & (x_0-1)y_0 & (x_0-1)^2 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(x_0-1, y_0-2) \\ F(x_0, y_0-2) \\ F(x_0+1, y_0-2) \\ F(x_0-1, y_0-1) \\ F(x_0, y_0-1) \\ F(x_0+1, y_0-1) \\ F(x_0-1, y_0) \end{pmatrix}$$

We define $x_0 = 0, \quad y_0 = 0$

$$\begin{pmatrix} 1 & -1 & -2 & 2 & 1 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 1 & -2 & -2 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(-1, -2) \\ F(0, -2) \\ F(1, -2) \\ F(-1, -1) \\ F(0, -1) \\ F(1, -1) \\ F(-1, 0) \end{pmatrix}$$

How to solve such a system?

$$A \cdot \mathbf{x} = \mathbf{b}$$

$$\mathbf{x} = A^{-1} \cdot \mathbf{b} \quad ?$$

Matrix Form:

$$\begin{pmatrix} 1 & (x_0-1) & (y_0-2) & (x_0-1)(y_0-2) & (x_0-1)^2 \\ 1 & x_0 & (y_0-2) & x_0(y_0-2) & x_0^2 \\ 1 & (x_0+1) & (y_0-2) & (x_0+1)(y_0-2) & (x_0+1)^2 \\ 1 & (x_0-1) & (y_0-1) & (x_0-1)(y_0-1) & (x_0-1)^2 \\ 1 & x_0 & (y_0-1) & x_0(y_0-1) & x_0^2 \\ 1 & (x_0+1) & (y_0-1) & (x_0+1)(y_0-1) & (x_0+1)^2 \\ 1 & (x_0-1) & y_0 & (x_0-1)y_0 & (x_0-1)^2 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(x_0-1, y_0-2) \\ F(x_0, y_0-2) \\ F(x_0+1, y_0-2) \\ F(x_0-1, y_0-1) \\ F(x_0, y_0-1) \\ F(x_0+1, y_0-1) \\ F(x_0-1, y_0) \end{pmatrix}$$

We define $x_0 = 0, \quad y_0 = 0$

$$\begin{pmatrix} 1 & -1 & -2 & 2 & 1 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 1 & -2 & -2 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix}$$

How to solve such a system?

$$A \cdot \mathbf{x} = \mathbf{b}$$

$$\mathbf{x} = A^{-1} \cdot \mathbf{b} \quad \textbf{?}$$

Can‘t invert non-square matrix… **Oh No!**

Moore–Penrose Pseudo-inverse to the Rescue!

The Moore–Penrose Pseudo-inverse A^+ of A is somehow a „closest match" to a matrix inverse if the real inverse does not exist *(e.g., non-square matrices)*.

Properties:

$$AA^+A = A$$

$$A^+AA^+ = A^+$$

$$(AA^+)^* = AA^+$$

$$(A^+A)^* = A^+A$$

Can be computed by **Singular Value Decomposition** (SVD).

If SVD is given by $A = U\Sigma V$,

then pseudo-inverse is given by $A^+ = V\Sigma^+U^*$.

Moore–Penrose Pseudoinverse to the Rescue!

Once the pseudo-inverse is known, the least-squares solution to an over–determined system is simply given by

$$
\begin{aligned}
A \cdot \mathbf{x} &= \mathbf{b} \\
\mathbf{x} &= A^{+} \cdot \mathbf{b}
\end{aligned}
$$

Matrix Form:

$$\begin{pmatrix} 1 & -1 & -2 & 2 & 1 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 1 & -2 & -2 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix} = \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix}$$

Use Pseudo-inverse:

$$\begin{pmatrix} -0.29 & 0.14 & -0.57 & 0 & 0.86 & 0.57 & 0.29 \\ 0.14 & -0.24 & -0.38 & -0.17 & 0.24 & 0.88 & -0.48 \\ -0.19 & -0.24 & -0.38 & 0 & 0.24 & 0.38 & 0.19 \\ 0.29 & -0.14 & -0.43 & 0 & 0.14 & 0.43 & -0.29 \\ 0.29 & -0.48 & 0.24 & 0.27 & -0.52 & 0.26 & 0.048 \end{pmatrix} \cdot \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix}$$

How to Perform the Extrapolation

$$\begin{pmatrix} -0.29 & 0.14 & -0.57 & 0 & 0.86 & 0.57 & 0.29 \\ 0.14 & -0.24 & -0.38 & -0.17 & 0.24 & 0.88 & -0.48 \\ -0.19 & -0.24 & -0.38 & 0 & 0.24 & 0.38 & 0.19 \\ 0.29 & -0.14 & -0.43 & 0 & 0.14 & 0.43 & -0.29 \\ 0.29 & -0.48 & 0.24 & 0.27 & -0.52 & 0.26 & 0.048 \end{pmatrix} \cdot \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix}$$

Put the coefficients back into our ansatz function:

$$F(x,y) = c_0 + c_x \cdot x + c_y \cdot y + c_{xy} \cdot xy + c_{x^2} \cdot x^2$$

$$\begin{aligned} F(0,0) &= c_0 + c_x \cdot 0 + c_y \cdot 0 + c_{xy} \cdot 0 + c_{x^2} \cdot 0^2 \\ &= c_0 \end{aligned}$$

How to Perform the Extrapolation

$$\begin{pmatrix} -0.29 & 0.14 & -0.57 & 0 & 0.86 & 0.57 & 0.29 \\ 0.14 & -0.24 & -0.38 & -0.17 & 0.24 & 0.88 & -0.48 \\ -0.19 & -0.24 & -0.38 & 0 & 0.24 & 0.38 & 0.19 \\ 0.29 & -0.14 & -0.43 & 0 & 0.14 & 0.43 & -0.29 \\ 0.29 & -0.48 & 0.24 & 0.27 & -0.52 & 0.26 & 0.048 \end{pmatrix} \cdot \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_x \\ c_y \\ c_{xy} \\ c_{x^2} \end{pmatrix}$$

Put the coefficients back into our ansatz function:

$$F(x,y) = c_0 + c_x \cdot x + c_y \cdot y + c_{xy} \cdot xy + c_{x^2} \cdot x^2$$

$$\begin{aligned} F(0,0) &= c_0 + c_x \cdot 0 + c_y \cdot 0 + c_{xy} \cdot 0 + c_{x^2} \cdot 0^2 \\ &= c_0 \end{aligned}$$

$$F(0,0) = c_0 = \begin{pmatrix} -0.29 & 0.14 & -0.57 & 0 & 0.86 & 0.57 & 0.29 \end{pmatrix} \cdot \begin{pmatrix} F(-1,-2) \\ F(0,-2) \\ F(1,-2) \\ F(-1,-1) \\ F(0,-1) \\ F(1,-1) \\ F(-1,0) \end{pmatrix}$$

→ **The only task that remains is a simple dot product!**

How to Perform the Extrapolation

The 2D example (7 equations, 5 unknowns) was only educational.

A typical 3D „real world" example:

- 350 data points (within a 7 x 7 x 7 cube) used for extrapolation
- 100 terms (all combinations up to $x^6y^6z^6$)

→ Coefficient matrix is 350 x 100

Pseudo-inverse of this matrix takes ≈ 1 second,
but required only once in the beginning

Extrapolation for one point is a dot product between two
350-component vectors, still reasonably fast.

Polynomial Extrapolation

Now we can extrapolate in any number of dimensions with any number of data points.

First Idea: Extrapolate directly in 4D space (*x, y, z, t*)

Known as „Tensor Compression“. Did not work well…

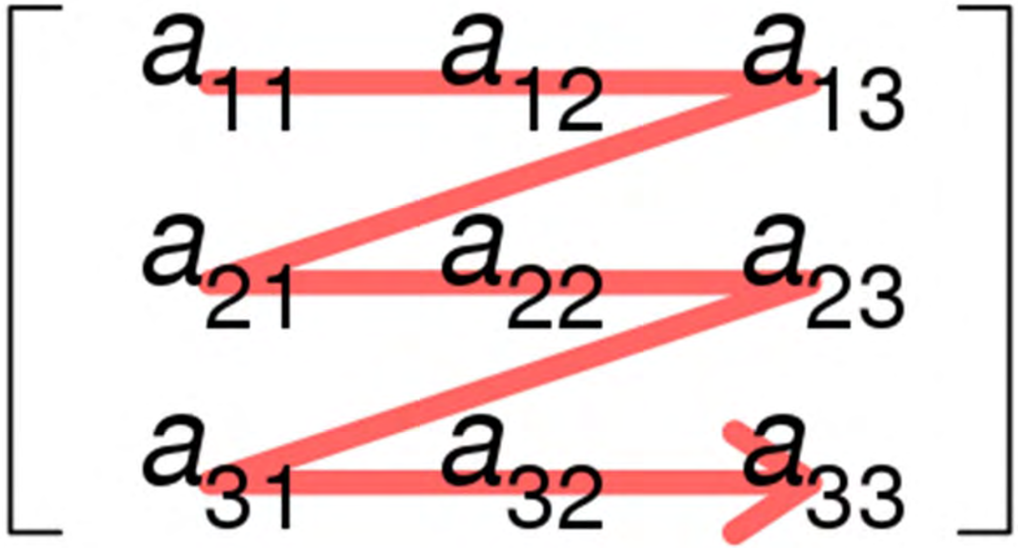
Second Idea: Predictor–Corrector setup

1. Predict all bin values from old frames by temporal extrapolation.

2. Correct estimate of current bin by spatially extrapolating the error in predicted values.

Space–Filling Hilbert Curve

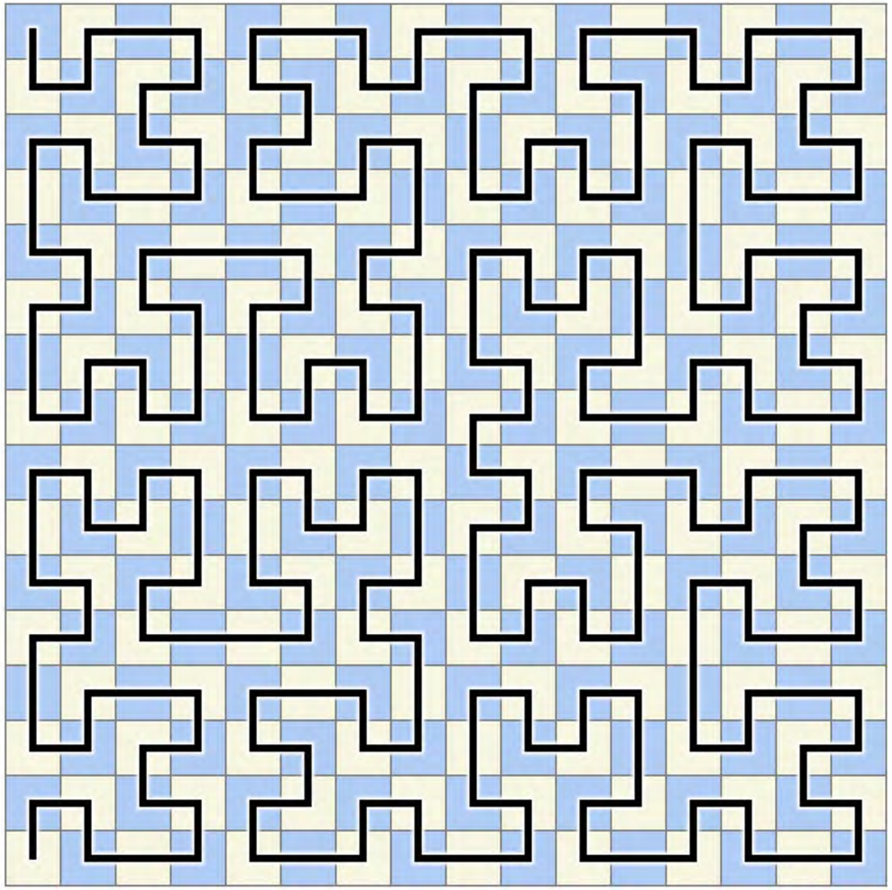
By simply storing the data line by line, we lose ***locality***.

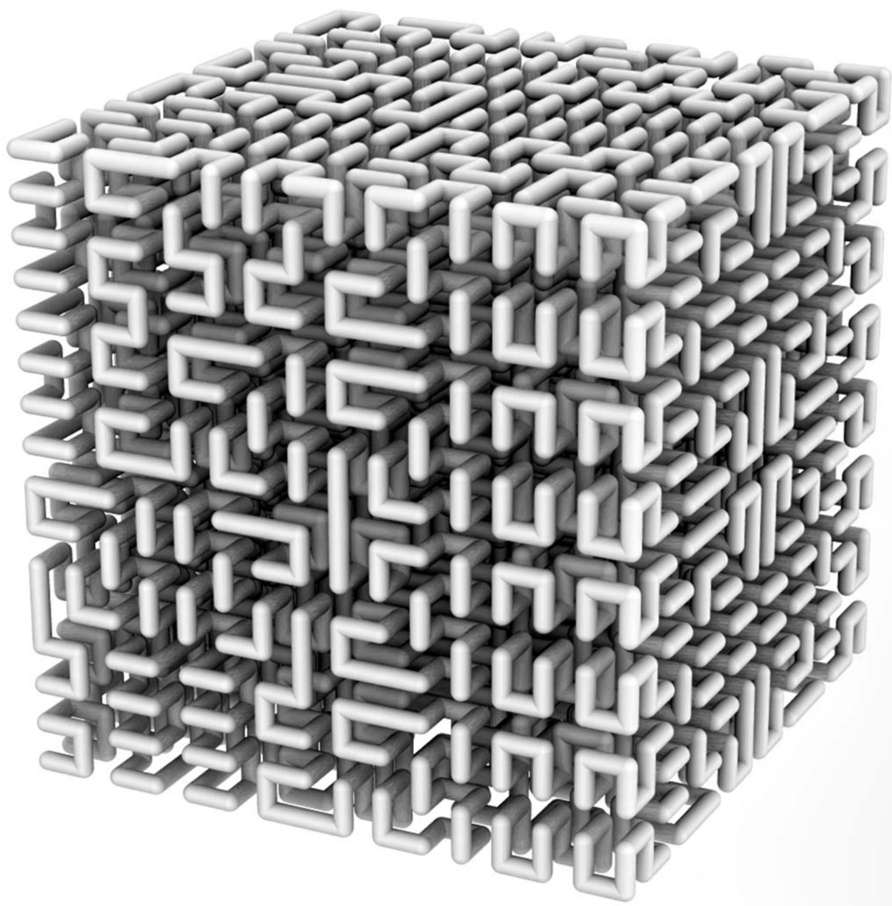


For a good compression ratio, we want to keep locality.

→ Need to re-order the data → **Space-filling curve!**

Space–Filling Hilbert Curve





This saves ~ 10 % of space!

Integer Stream Compression

Until now, we did not gain anything:

Have one integer number for each grid point
→ number of data points not changed…

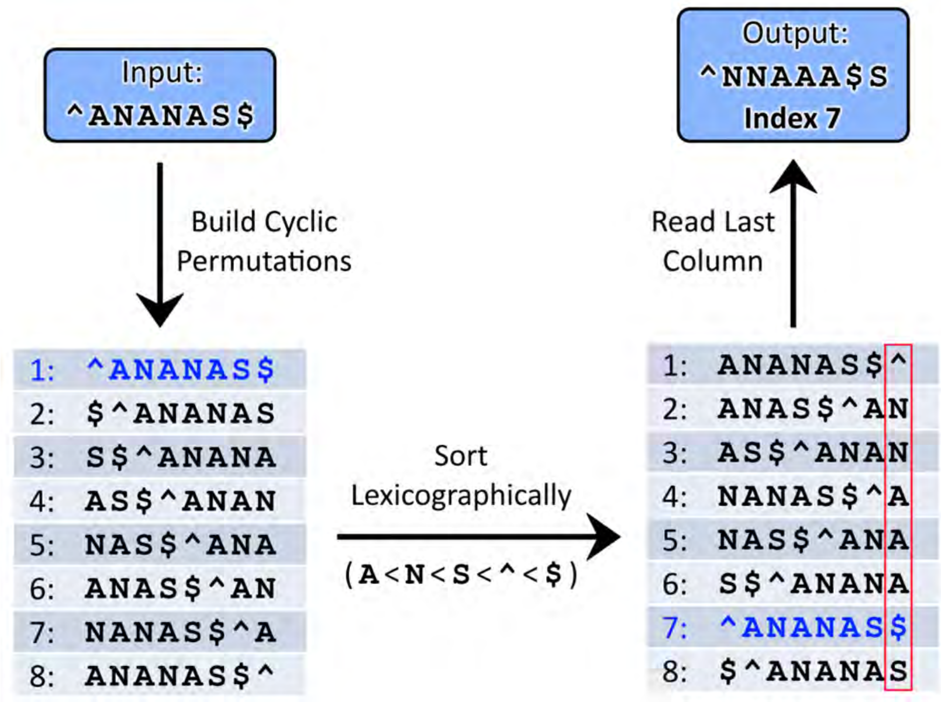
Need to exploit reduced information entropy now.

The compression algorithm is **closely related to bzip2**:

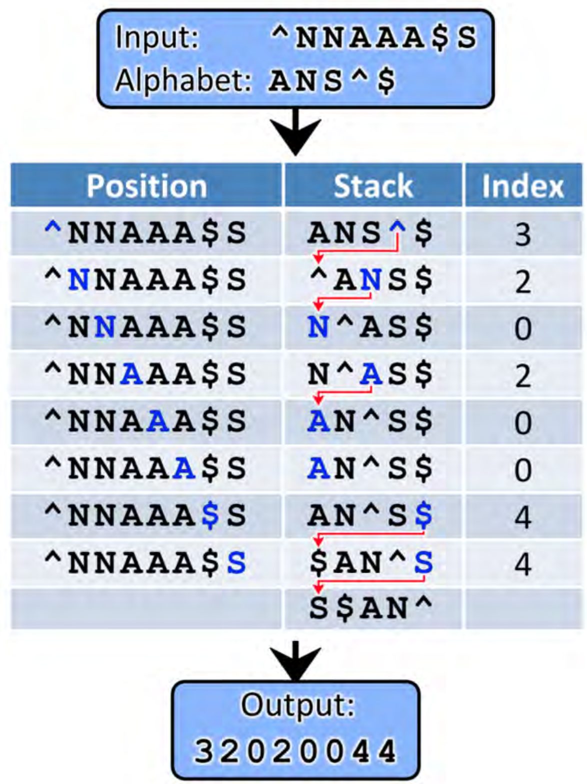
- Burrows–Wheeler Transform (block sorting)
- Move-To-Front-Transform
- Run Length Encoding
- Multi-Table Optimized Huffman Encoding using Canonical Huffman Codes

Integer Stream Compression

Burrows–Wheeler Transformation:



Move-to-Front Transformation:



```
bool CompressCube(
    const char *inp,
    const char *outp,
    const char *ref,
    int start,
    int steps,
    int stride,
    int corder,
    bool optcorder,
    int aorder,
    bool optaorder,
    int eps,
    int csigni,
    int asigni,
    int csplit,
    int asplit,
    int cblock,
    int ablock,
    int ctables,
    bool optctables,
    bool cpreopt,
    int cmaxiter,
    int cmaxchunk,
    int atables,
    bool optatables,
    bool hilbert,
    double nbhfac,
    bool ccoderun,
    bool cbw,
    bool cmtf,
    bool acoderun,
    bool abw,
    bool amtf,
    bool apreopt,
    int amaxiter,
    bool asortatom,
    int amaxchunk,
    bool usecextra,
    bool useaextra,
    int aextratrange,
    int aextratorder,
    double aextratimeexpo,
    int cextrasrangex,
    int cextrasrangey,
    int cextrasrangez,
    int cextratrange,
    int cextrasorder,
    int cextratorder,
    int cextraoffsetx,
    int cextraoffsety,
    int cextraoffsetz,
    bool cextracrosss,
    bool cextracrosst,
    bool cextrawrap,
    bool cextracrossranges,
    bool cextracrossranget,
    double cextradistexpo,
    double cextratimeexpo,
    bool cextrapredcorr,
    double cextracorrfactor,
    int optimize,
    int optsteps,
    bool onlyopt,
    bool comment,
    bool compare,
    bool dummyread,
    bool verbose
);
```

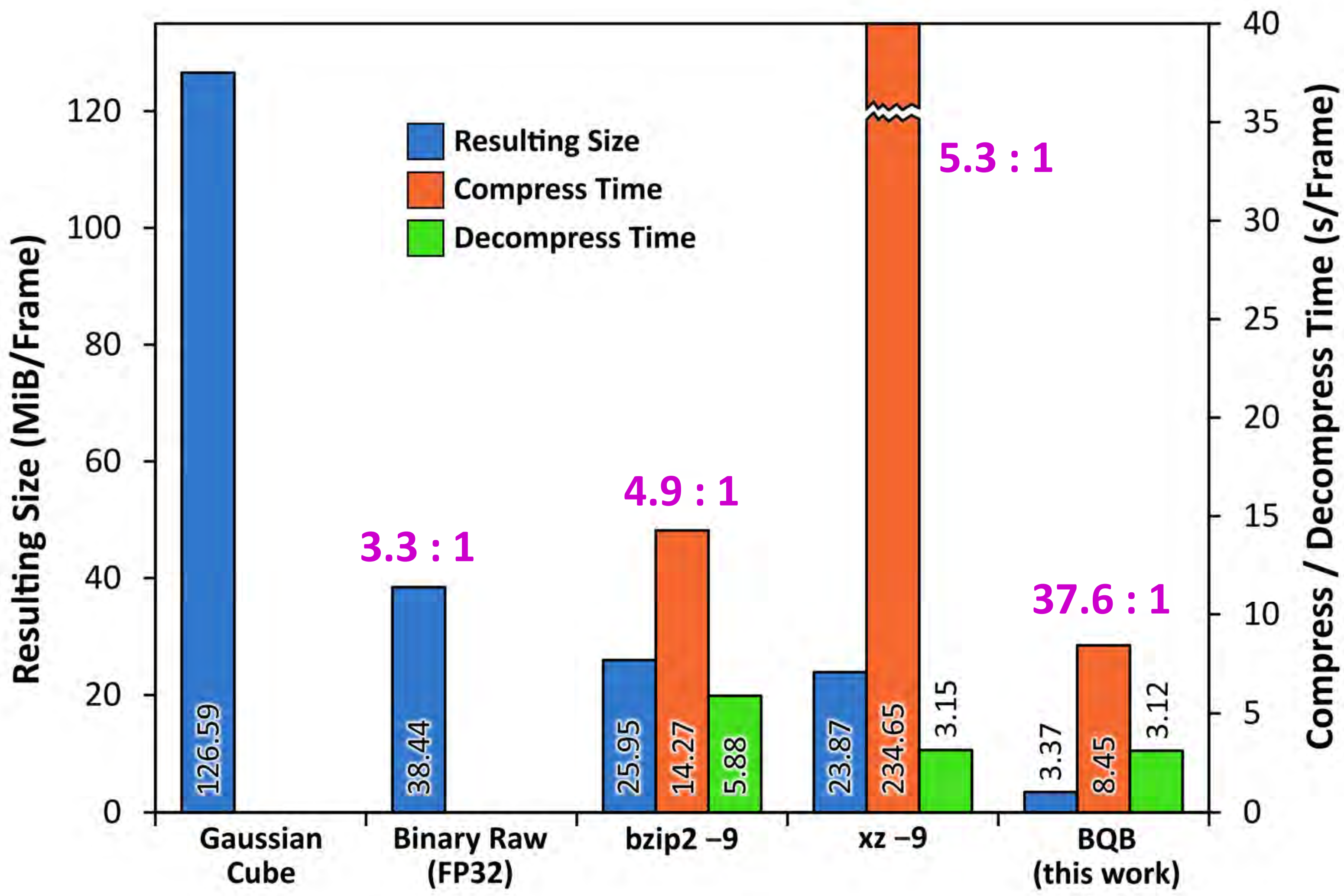
Many technical parameters determine compression ratio and time *(57 parameters!)*

Our implementation tries to optimize these parameters for each input trajectory in the beginning

4 optimization modes available
(from „quick“ to „exhaustive“)

Result: Volumetric Data

Electron Density, 36 [EMIm][OAc], 936 Atoms,
Avg. over 1000 frames, Grid 216 x 216 x 216, Δt = 0.5 fs.



Single Volumetric Frames

If there is ony 1 cube frame to compress, no temporal extrapolation is possible.

But spatial extrapolation can go to a higher order then ☺

→ We still achieve a ratio of ≈ 19 : 1

→ Also very efficient for single cube frames.

Position Trajectories

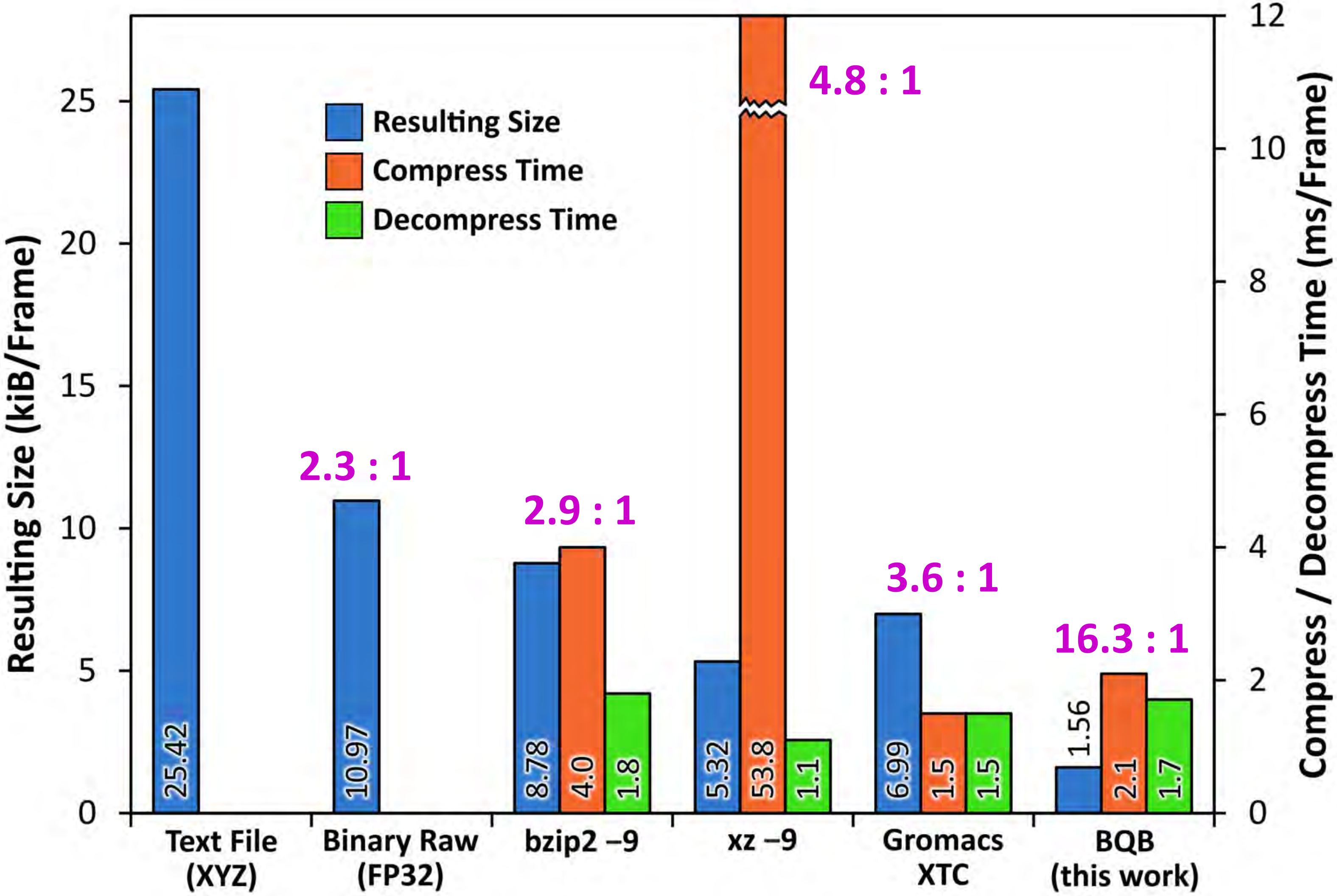
Would this also work for „normal“ position trajectories?

Use temporal extrapolation from last atom positions.

Result: Yes, it works ☺

Result: Position Trajectory

36 [EMIm][OAc], 936 Atoms, Precision 10^{-5} Angstrom,
Avg. over 1000 Frames, $\Delta t = 0.5$ fs.



For both position trajectories („.xyz“) and volumetric trajectories („.cube“), our format has by far the best compression ratio, but is still fast to read / write.

That‘s nice ☺

Example: The 13 TiB of volumetric data from the ROA spectrum are now merely 350 GiB.

The BQB File Format

Compressed data is saved in the newly developed BQB file format:

- Versatile multi-purpose format
- Open source and well documented (in future ☺)
- Can store all required data *(cell vectors, atom labels, charges, comment lines, velocities, etc.)*
- Contains headers & checksums → Corruption resistent
- Contains index → Fast seeking and random access

We hope that the BQB format is adopted in many programs and will be widely used.

The BQB File Format

Why didn‘t we use an existing format such as HDF5?

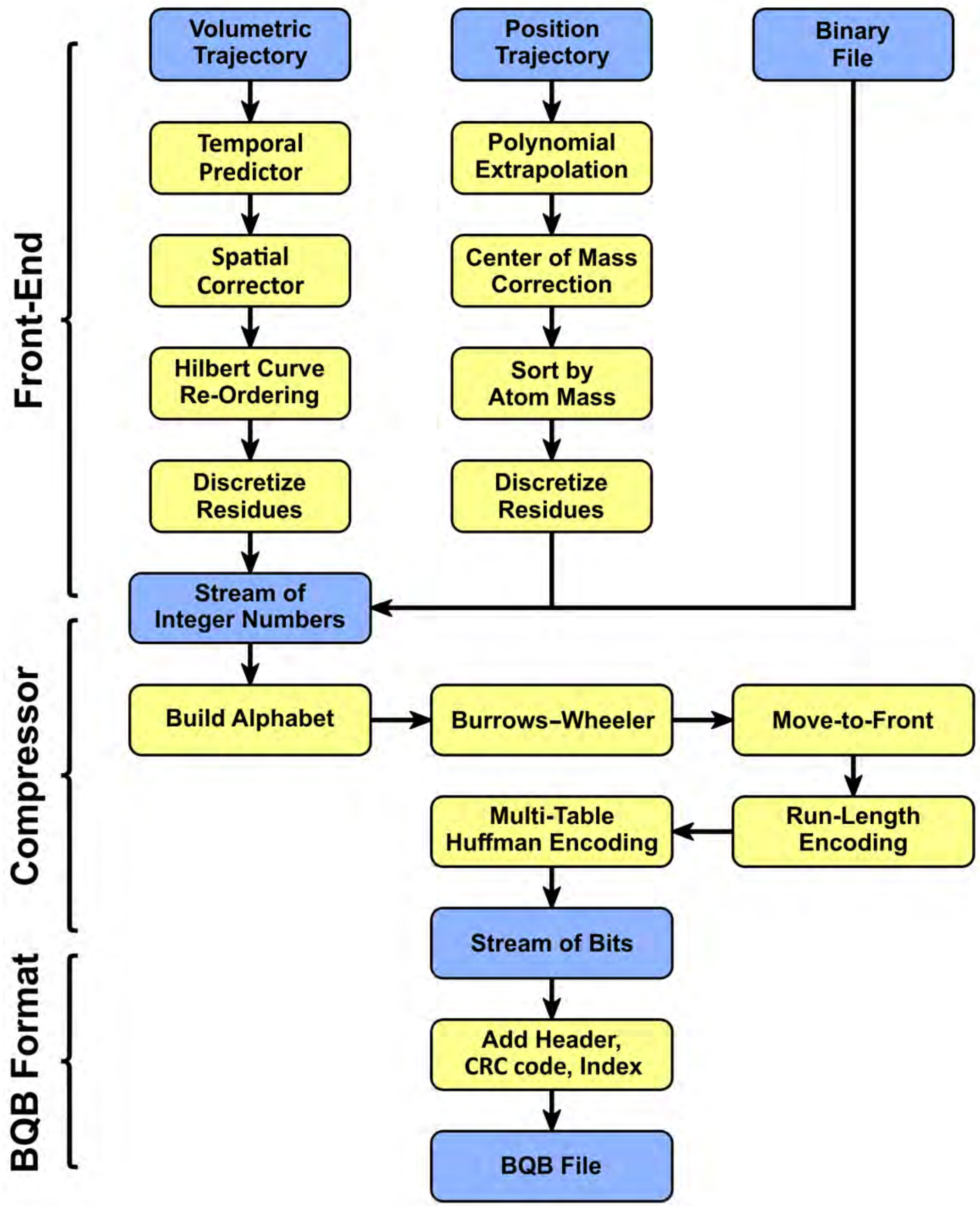
HDF5 is a general-purpose format with a **huge flexibility** for all different applications.

BQB is specifically designed for simulation trajectories, and aims at **maximum compression ratio**.

BQB stores bit streams and does not care for byte boundaries → not a single bit is wasted.

→ Both formats are not at all competitors.

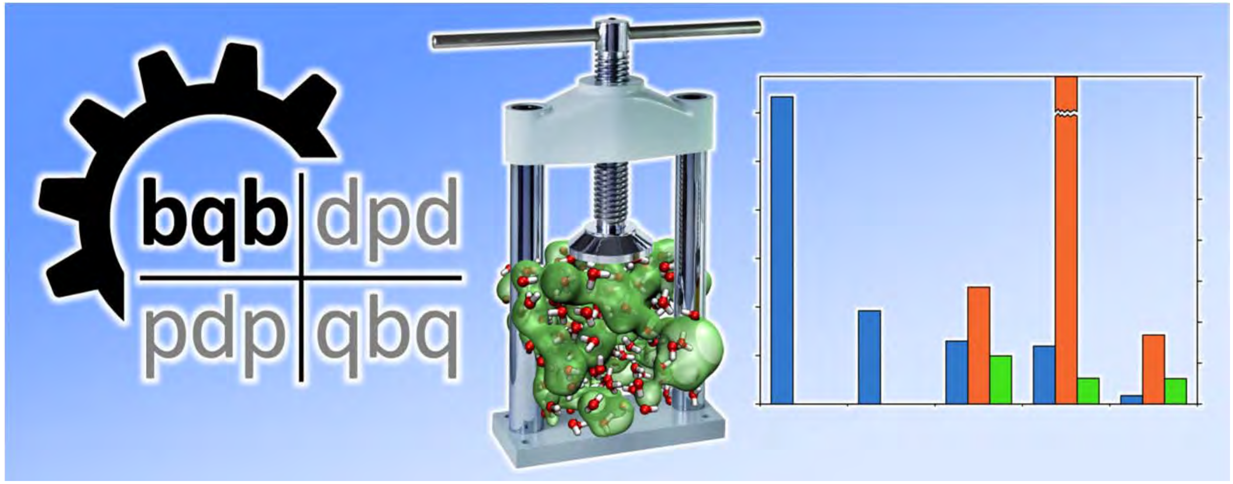
Flowchart



Front-End
Compressor
BQB Format
Volumetric Trajectory
Temporal Predictor
Spatial Corrector
Hilbert Curve Re-Ordering
Discretize Residues
Position Trajectory
Polynomial Extrapolation
Center of Mass Correction
Sort by Atom Mass
Discretize Residues
Binary File
Stream of Integer Numbers
Build Alphabet
Burrows–Wheeler
Move-to-Front
Run-Length Encoding
Multi-Table Huffman Encoding
Stream of Bits
Add Header, CRC code, Index
BQB File

Article

Article recently submitted:



M. Brehm, M. Thomas: "An Efficient Lossless Compression Algorithm for Trajectories of Atom Positions and Volumetric Data", *J. Chem. Inf. Model.* **2018**, *submitted*.

Will hopefully be accepted in the next weeks ☺

Code will be published

As soon as the article is online, you can find the implementation and documentation on

`www.brehm-research.de/bqb`

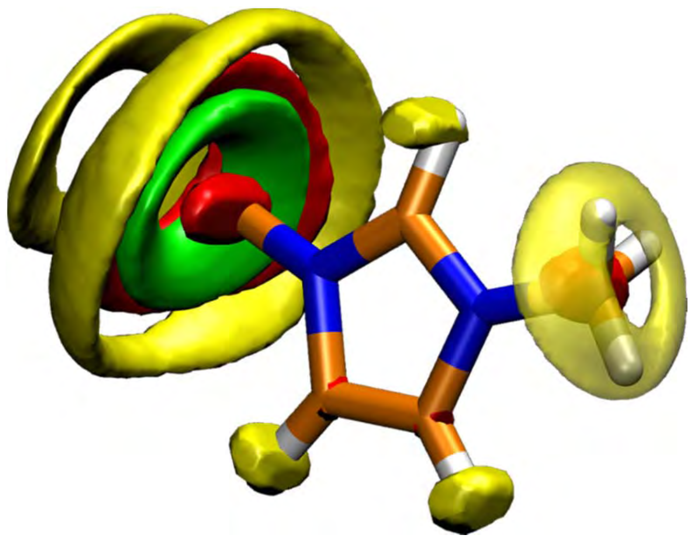
What will be available?

- **bqbtool**: Command line tool for working with bqb files
- **libbqb**: C++ library (≈ 36 000 lines) to include in other projects (maybe CP2k? ☺)
- **Technical documentation** of the bqb file format

All licensed under **GNU LGPL v3**.

Implemented in TRAVIS

Whole implementation is also included in TRAVIS
(you can use TRAVIS instead of bqbtool).



Will be available to the public after the paper is out.

4. Practical Workflow for Spectra

General Workflow

0. **Preparation**
   → *Starting configuration* **10 kiB**

1. **Simulate trajectory**
   → *XYZ file* **1 GiB**

2. **Obtain electron density trajectories w/ ext. field**
   → *CUBE files* **3 TiB**

3. **Compress volumetric trajectories (optional)**
   → *BQB files* **100 GiB**

4. **Solve current PDE, perform Voronoi Integration**
   → *EMP files* **1 GiB**

5. **Compute spectra from EMP property files**
   → *Spectra (text files)* **10 kiB**

General Workflow

0. **Preparation**
   → *Starting configuration* 10 kiB
1. **Simulate trajectory**
   → *XYZ file* 1 GiB
2. **Obtain electron density trajectories w/ ext. field**
   → *CUBE files* 3 TiB

   } **CP2k** (steps 1–2)
3. **Compress volumetric trajectories (optional)**
   → *BQB files* 100 GiB

   bqbtool
4. **Solve current PDE, perform Voronoi Integration**
   → *EMP files* 1 GiB
5. **Compute spectra from EMP property files**
   → *Spectra (text files)* 10 kiB

   } **TRAVIS** (steps 4–5)

0.) Preparation

- Decide on the system composition ☺
- Prepare bulk phase cell, e.g. with PackMol
- Run force-field MD for equilibration (e.g. with OPLS-AA)
- Extract the last snapshot as starting configuration for AIMD

1.) Trajectory

a) Massive Equilibration (1/4)

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

1.) Trajectory

a) Massive Equilibration (1/4)

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL
    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID
    &QS
      EPS_DEFAULT 1.0E-12
    &END QS
    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

- „Officially“, you would need to converge your PW cutoff…
- A PW cutoff of 350 Ry is typically Ok for organic liquids
- NGRIDS 4 is a good setting for MOLOPT basis sets
- REL_CUTOFF 40 is default, 50 is more accurate

1.) Trajectory

a) Massive Equilibration (1/4)

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

- EPS_DEFAULT of 10^{-12} is often a good compromise

1.) Trajectory

a) Massive Equilibration (1/4)

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

EPS_SCF should always be square root of EPS_DEFAULT

→ EPS_DEFAULT 1.0E-12 means
EPS_SCF 1.0E-6

This is a good compromise for SCF convergence.

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

1.) Trajectory

a) Massive Equilibration (1/4)

- OT („Orbital Transformation“) is very fast and efficient for molecular liquids
- Choice of preconditioner is crucial for efficiency
- FULL_KINETIC is a fast choice for well-behaved MD runs
- If there are problems with SCF convergence, use FULL_SINGLE_INVERSE, or even FULL_ALL together with ENERGY_GAP 0.001

1.) Trajectory

a) Massive Equilibration (1/4)

- OUTER_SCF is important to cope with convergence problems, especially in the first steps
- Make sure to use the same EPS_SCF value at both positions!

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

1.) Trajectory

a) Massive Equilibration (1/4)

```
&FORCE_EVAL
  METHOD Quickstep
  &DFT
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL

    &MGRID
      CUTOFF 350
      NGRIDS 4
      REL_CUTOFF 40
    &END MGRID

    &QS
      EPS_DEFAULT 1.0E-12
    &END QS

    &SCF
      SCF_GUESS ATOMIC
      MAX_SCF 15
      &OT
        PRECONDITIONER FULL_KINETIC
        MINIMIZER DIIS
      &END
      &OUTER_SCF
        MAX_SCF 20
        EPS_SCF 1.0E-6
      &END
      EPS_SCF 1.0E-6
      &PRINT
        &RESTART
          &EACH
            MD 0
          &END EACH
        &END
      &END
    &END SCF
```

- Stop printing WFN file in every MD step (can be large → slows down the simulation)

1.) Trajectory

a) Massive Equilibration (2/4)

```
  &XC
    &XC_FUNCTIONAL BLYP
    &END XC_FUNCTIONAL

    &XC_GRID
      XC_DERIV NN10_SMOOTH
      XC_SMOOTH_RHO NN10
    &END XC_GRID

    &vdW_POTENTIAL
      DISPERSION_FUNCTIONAL PAIR_POTENTIAL
      &PAIR_POTENTIAL
        TYPE DFTD3
        PARAMETER_FILE_NAME dftd3.dat
        REFERENCE_FUNCTIONAL BLYP
      &END PAIR_POTENTIAL
    &END vdW_POTENTIAL
  &END XC
&END DFT

&SUBSYS
  &CELL
    ABC  19.8302  19.8302  19.8302
  &END CELL

  &COORD
    C    6.54092440  5.69704107  9.82218709
    H    6.10696440  5.01165107 10.56792709
    ...
  &END COORD
```

1.) Trajectory

a) Massive Equilibration (2/4)

- Take functional of your choice ☺
- For organic liquids, often BLYP and PBE are reasonable choices.

```
  &XC
    &XC_FUNCTIONAL BLYP
    &END XC_FUNCTIONAL
    &XC_GRID
      XC_DERIV NN10_SMOOTH
      XC_SMOOTH_RHO NN10
    &END XC_GRID
    &vdW_POTENTIAL
      DISPERSION_FUNCTIONAL PAIR_POTENTIAL
      &PAIR_POTENTIAL
        TYPE DFTD3
        PARAMETER_FILE_NAME dftd3.dat
        REFERENCE_FUNCTIONAL BLYP
      &END PAIR_POTENTIAL
    &END vdW_POTENTIAL
  &END XC
&END DFT

&SUBSYS
  &CELL
    ABC  19.8302  19.8302  19.8302
  &END CELL

  &COORD
    C    6.54092440  5.69704107  9.82218709
    H    6.10696440  5.01165107 10.56792709
    ...
  &END COORD
```

```
  &XC
    &XC_FUNCTIONAL BLYP
    &END XC_FUNCTIONAL
    &XC_GRID
      XC_DERIV NN10_SMOOTH
      XC_SMOOTH_RHO NN10
    &END XC_GRID
    &vdW_POTENTIAL
      DISPERSION_FUNCTIONAL PAIR_POTENTIAL
      &PAIR_POTENTIAL
        TYPE DFTD3
        PARAMETER_FILE_NAME dftd3.dat
        REFERENCE_FUNCTIONAL BLYP
      &END PAIR_POTENTIAL
    &END vdW_POTENTIAL
  &END XC
&END DFT

&SUBSYS
  &CELL
    ABC  19.8302  19.8302  19.8302
  &END CELL

  &COORD
    C     6.54092440  5.69704107  9.82218709
    H     6.10696440  5.01165107 10.56792709
    ...
  &END COORD
```

1.) Trajectory

a) Massive Equilibration (2/4)

- Smoothing mitigates the break of translational invariance due to the plane waves
- For cutoffs < 600 Ry (as we all use), this is **absolutely mandatory**

1.) Trajectory

a) Massive Equilibration (2/4)

```
  &XC
    &XC_FUNCTIONAL BLYP
    &END XC_FUNCTIONAL

    &XC_GRID
      XC_DERIV NN10_SMOOTH
      XC_SMOOTH_RHO NN10
    &END XC_GRID

    &vdW_POTENTIAL
      DISPERSION_FUNCTIONAL PAIR_POTENTIAL
      &PAIR_POTENTIAL
        TYPE DFTD3
        PARAMETER_FILE_NAME dftd3.dat
        REFERENCE_FUNCTIONAL BLYP
      &END PAIR_POTENTIAL
    &END vdW_POTENTIAL
  &END XC
&END DFT

&SUBSYS
  &CELL
    ABC  19.8302  19.8302  19.8302
  &END CELL

  &COORD
    C    6.54092440  5.69704107  9.82218709
    H    6.10696440  5.01165107 10.56792709
    ...
  &END COORD
```

- For organic liquids, you always want to use a dispersion correction
- Grimme‘s D3 is often a very good choice

1.) Trajectory

a) Massive Equilibration (3/4)

```
    &KIND C
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q4
    &END KIND
    &KIND H
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q1
    &END KIND
    &KIND N
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q5
    &END KIND
  &END SUBSYS
&END FORCE_EVAL


&GLOBAL
  PROJECT SomeSystem
  RUN_TYPE MD
  PRINT_LEVEL LOW
  FFTW_PLAN_TYPE EXHAUSTIVE
  FFTW_WISDOM_FILE_NAME wisdom.dat
&END GLOBAL
```

1.) Trajectory

a) Massive Equilibration (3/4)

```
    &KIND C
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q4
    &END KIND

    &KIND H
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q1
    &END KIND

    &KIND N
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q5
    &END KIND
  &END SUBSYS
&END FORCE_EVAL


&GLOBAL
  PROJECT SomeSystem
  RUN_TYPE MD
  PRINT_LEVEL LOW
  FFTW_PLAN_TYPE EXHAUSTIVE
  FFTW_WISDOM_FILE_NAME wisdom.dat
&END GLOBAL
```

- Basis sets of the type DZVP-MOLOPT-SR-GTH are a **very good compromise** for computing spectra
- For non-homogeneous systems (gas phase, interfaces), you may want to leave out the „-SR-“
- For high accuracy, you can also go to TZVP or even TZVPP.

```
    &KIND C
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q4
    &END KIND
    &KIND H
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q1
    &END KIND
    &KIND N
      BASIS_SET DZVP-MOLOPT-SR-GTH
      POTENTIAL GTH-BLYP-q5
    &END KIND
  &END SUBSYS
&END FORCE_EVAL


&GLOBAL
  PROJECT SomeSystem
  RUN_TYPE MD
  PRINT_LEVEL LOW
  FFTW_PLAN_TYPE EXHAUSTIVE
  FFTW_WISDOM_FILE_NAME wisdom.dat
&END GLOBAL
```

1.) Trajectory

a) Massive Equilibration (3/4)

- This takes 1-2 minutes in the start of the run, but can make each MD step faster by ≈ 15 %
- I recommend this for runs of 12 hours or longer
- The wisdom.dat file is written at the end, and makes FFT planning faster in following runs

1.) Trajectory

a) Massive Equilibration (4/4)

```
&MOTION
  &MD
    ENSEMBLE NVT
    STEPS 2000
    TIMESTEP 0.5
    &THERMOSTAT
      TYPE NOSE
      REGION MASSIVE
      &NOSE
        TIMECON 10.00
      &END NOSE
    &END THERMOSTAT
    TEMPERATURE 350
  &END MD
  &PRINT
    &RESTART
      BACKUP_COPIES 0
      &EACH
        MD 1
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

```
&MOTION
  &MD
    ENSEMBLE NVT
    STEPS 2000
    TIMESTEP 0.5
    &THERMOSTAT
      TYPE NOSE
      REGION MASSIVE
      &NOSE
        TIMECON 10.00
      &END NOSE
    &END THERMOSTAT
    TEMPERATURE 350
  &END MD
  &PRINT
    &RESTART
      BACKUP_COPIES 0
      &EACH
        MD 1
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

1.) Trajectory

a) Massive Equilibration (4/4)

- If you have hydrogen atoms, always use Δt = 0.5 fs
- **Don‘t** feel tempted to 1.0 fs (as in force field MD); every MD step will take more SCF cycles, such that you save almost nothing
- Solving the current PDE (for VCD and ROA) requires 0.5 fs even if no hydrogen atoms are present.

```
&MOTION
  &MD
    ENSEMBLE NVT
    STEPS 2000
    TIMESTEP 0.5
    &THERMOSTAT
      TYPE NOSE
      REGION MASSIVE
      &NOSE
        TIMECON 10.00
      &END NOSE
    &END THERMOSTAT
    TEMPERATURE 350
  &END MD
  &PRINT
    &RESTART
      BACKUP_COPIES 0
      &EACH
        MD 1
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

1.) Trajectory

a) Massive Equilibration (4/4)

- Massive thermostating and rather strong coupling (time constant of 10 fs) in the start

1.) Trajectory

a) Massive Equilibration (4/4)

```
&MOTION
  &MD
    ENSEMBLE NVT
    STEPS 2000
    TIMESTEP 0.5
    &THERMOSTAT
      TYPE NOSE
      REGION MASSIVE
      &NOSE
        TIMECON 10.00
      &END NOSE
    &END THERMOSTAT
    TEMPERATURE 350
  &END MD
  &PRINT
    &RESTART
      BACKUP_COPIES 0
      &EACH
        MD 1
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

- Use temperature of your choice
- As AIMD runs are quite short (picoseconds), higher temperature helps to improve sampling
- As „room temperature“, we typically use 350 K

1.) Trajectory

a) Massive Equilibration (4/4)

```
&MOTION
  &MD
    ENSEMBLE NVT
    STEPS 2000
    TIMESTEP 0.5
    &THERMOSTAT
      TYPE NOSE
      REGION MASSIVE
      &NOSE
        TIMECON 10.00
      &END NOSE
    &END THERMOSTAT
    TEMPERATURE 350
  &END MD
  &PRINT
    &RESTART
      BACKUP_COPIES 0
      &EACH
        MD 1
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

- Stop spamming all kinds of restart backup / history files
- Only one single restart file, which is written in every MD step

1.) Trajectory

a) Massive Equilibration

Run for ≈ 2000 time steps (1.0 ps)

1.) Trajectory

b) Non-massive Equilibration

```
&FORCE_EVAL
  &DFT
    &SCF
      SCF_GUESS RESTART
    &END SCF
  &END DFT
&END FORCE_EVAL

&MOTION
  &MD
    &THERMOSTAT
      REGION MASSIVE
      &NOSE
        TIMECON 100.00
      &END NOSE
    &END THERMOSTAT
  &END MD
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

1.) Trajectory

b) Non-massive Equilibration

- Restart from last WFN instead of initial guess

```
&FORCE_EVAL
  &DFT
    &SCF
      SCF_GUESS RESTART
    &END SCF
  &END DFT
&END FORCE_EVAL

&MOTION
  &MD
    &THERMOSTAT
      REGION MASSIVE
      &NOSE
        TIMECON 100.00
      &END NOSE
    &END THERMOSTAT
  &END MD
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

1.) Trajectory

b) Non-massive Equilibration

```
&FORCE_EVAL
  &DFT
    &SCF
      SCF_GUESS RESTART
    &END SCF
  &END DFT
&END FORCE_EVAL

&MOTION
  &MD
    &THERMOSTAT
      REGION MASSIVE
      &NOSE
        TIMECON 100.00
      &END NOSE
    &END THERMOSTAT
  &END MD
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

- Remove the „MASSIVE“ after the first equilibration phase

```
&FORCE_EVAL
  &DFT
    &SCF
      SCF_GUESS RESTART
    &END SCF
  &END DFT
&END FORCE_EVAL

&MOTION
  &MD
    &THERMOSTAT
      REGION MASSIVE
      &NOSE
        TIMECON 100.00
      &END NOSE
    &END THERMOSTAT
  &END MD
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

1.) Trajectory

b) Non-massive Equilibration

- Weaker thermostat coupling (time constant 100 fs) for second equilibration and production run
- Strong thermostat coupling might distort the dynamics and spectra…

1.) Trajectory

b) Non-massive Equilibration

```
&FORCE_EVAL
  &DFT
    &SCF
      SCF_GUESS RESTART
    &END SCF
  &END DFT
&END FORCE_EVAL

&MOTION
  &MD
    &THERMOSTAT
      REGION MASSIVE
      &NOSE
        TIMECON 100.00
      &END NOSE
    &END THERMOSTAT
  &END MD
&END MOTION
```

```
&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

- You need the EXT_RESTART block now
- Make sure the name of the restart file matches
- After turning off MASSIVE, you want RESTART_THERMOSTATE .FALSE.

1.) Trajectory

b) Non-massive Equilibration

Run for ≈ 20 000 time steps (10.0 ps)

1.) Trajectory

c) Production Run

```
&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

```
&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
  RESTART_THERMOSTAT .FALSE.
&END
```

1.) Trajectory

c) Production Run

- Don‘t forget to remove RESTART_THERMOSTATE .FALSE. before starting production run

1.) Trajectory

c) Production Run

Run for ≈ 60 000 time steps (30.0 ps)

Now we have a 30 ps production trajectory which contains all motions for the spectra.

2.) Electron Density

- We traverse the trajectory again, and store volumetric electron density in every *n*-th frame
- For IR and VCD: Only one field-free calculation required
- For Raman and ROA: Field-free + 3 field directions → 4 runs
- For IR and Raman: Sufficient to consider every 8th frame (→ every 4.0 fs)
- For VCD and ROA: Need every frame (every 0.5 fs)

2.) Electron Density (1/2)

```
&FORCE_EVAL
  &DFT
    &PERIODIC_EFIELD
      INTENSITY  5.0E-3
      POLARISATION  1.0  0.0  0.0
    &END PERIODIC_EFIELD

    &PRINT
      &E_DENSITY_CUBE
        STRIDE 1 1 1
        FILENAME =result.cube
        APPEND
      &END
    &END PRINT

    &LOCALIZE
      METHOD CRAZY
      JACOBI_FALLBACK
      MAX_ITER 500
      &PRINT
        &WANNIER_CENTERS
          IONS+CENTERS
          FILENAME =wannier.xyz
          &EACH
            MD 1
          &END EACH
        &END
      &END PRINT
    &END LOCALIZE

  &END DFT
&END FORCE_EVAL
```

2.) Electron Density (1/2)

```
&FORCE_EVAL
  &DFT
    &PERIODIC_EFIELD
      INTENSITY  5.0E-3
      POLARISATION  1.0  0.0  0.0
    &END PERIODIC_EFIELD

    &PRINT
      &E_DENSITY_CUBE
        STRIDE 1 1 1
        FILENAME =result.cube
        APPEND
      &END
    &END PRINT

    &LOCALIZE
      METHOD CRAZY
      JACOBI_FALLBACK
      MAX_ITER 500
      &PRINT
        &WANNIER_CENTERS
          IONS+CENTERS
          FILENAME =wannier.xyz
          &EACH
            MD 1
          &END EACH
        &END
      &END PRINT
    &END LOCALIZE

  &END DFT
&END FORCE_EVAL
```

- We need an external electric field which works with periodic systems.
- An external field strength of 5.0E-3 a.u. is a good compromise between noise and linearity (corresponds to $2.5 * 10^9$ V/m !)
- POLARIZATION gives the field vector (here: positive X direction)

2.) Electron Density (1/2)

```
&FORCE_EVAL
  &DFT
    &PERIODIC_EFIELD
      INTENSITY  5.0E-3
      POLARISATION  1.0  0.0  0.0
    &END PERIODIC_EFIELD

    &PRINT
      &E_DENSITY_CUBE
        STRIDE 1 1 1
        FILENAME =result.cube
        APPEND
      &END
    &END PRINT

    &LOCALIZE
      METHOD CRAZY
      JACOBI_FALLBACK
      MAX_ITER 500
      &PRINT
        &WANNIER_CENTERS
          IONS+CENTERS
          FILENAME =wannier.xyz
          &EACH
            MD 1
          &END EACH
        &END
      &END PRINT
    &END LOCALIZE

  &END DFT
&END FORCE_EVAL
```

- Write the electron density in each MD step to a CUBE trajectory
- STRIDE 1 1 1 is vital for Voronoi integration

2.) Electron Density (1/2)

```
&FORCE_EVAL
  &DFT
    &PERIODIC_EFIELD
      INTENSITY  5.0E-3
      POLARISATION  1.0  0.0  0.0
    &END PERIODIC_EFIELD

    &PRINT
      &E_DENSITY_CUBE
        STRIDE 1 1 1
        FILENAME =result.cube
        APPEND
      &END
    &END PRINT

    &LOCALIZE
      METHOD CRAZY
      JACOBI_FALLBACK
      MAX_ITER 500
      &PRINT
        &WANNIER_CENTERS
          IONS+CENTERS
          FILENAME =wannier.xyz
          &EACH
            MD 1
          &END EACH
        &END
      &END PRINT
    &END LOCALIZE
  &END DFT
&END FORCE_EVAL
```

- Don‘t compute Wannier centers if you don‘t have to (can waste **a lot** of time if CRAZY does not converge)
- If you really need it, insert this section
- Higher numbers for MAX_ITER typically are of no use (if it did not converge after 500 iterations, it will often never converge)

2.) Electron Density (2/2)

```
&MOTION
  &MD
    ENSEMBLE REFTRAJ
    STEPS 1024
    &REFTRAJ
      EVAL_ENERGY_FORCES
      FIRST_SNAPSHOT 1
      TRAJ_FILE_NAME SomeSystem-pos-1.xyz
    &END REFTRAJ
  &END MD
  &PRINT
    &RESTART
      &EACH
        MD 0
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
&END
```

2.) Electron Density (2/2)

```
&MOTION
  &MD
    ENSEMBLE REFTRAJ
    STEPS 1024
    &REFTRAJ
      EVAL_ENERGY_FORCES
      FIRST_SNAPSHOT 1
      TRAJ_FILE_NAME SomeSystem-pos-1.xyz
    &END REFTRAJ
  &END MD
  &PRINT
    &RESTART
      &EACH
        MD 0
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
&END
```

- Follow the pre-computed reference trajectory instead of doing a „true“ MD
- Make sure to specify the correct reference trajectory file name
- Enter the FIRST_SNAPSHOT and STEPS according to your needs
- EVAL_ENERGY _FORCES is important to re-compute the electron structure

2.) Electron Density (2/2)

```
&MOTION
  &MD
    ENSEMBLE REFTRAJ
    STEPS 1024
    &REFTRAJ
      EVAL_ENERGY_FORCES
      FIRST_SNAPSHOT 1
      TRAJ_FILE_NAME SomeSystem-pos-1.xyz
    &END REFTRAJ
  &END MD
  &PRINT
    &RESTART
      &EACH
        MD 0
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION

&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
&END
```

- This time: No restart files at all, because we just follow the reference trajectory

2.) Electron Density (2/2)

```
&MOTION
  &MD
    ENSEMBLE REFTRAJ
    STEPS 1024
    &REFTRAJ
      EVAL_ENERGY_FORCES
      FIRST_SNAPSHOT 1
      TRAJ_FILE_NAME SomeSystem-pos-1.xyz
    &END REFTRAJ
  &END MD
  &PRINT
    &RESTART
      &EACH
        MD 0
      &END EACH
    &END RESTART
    &RESTART_HISTORY
      &EACH
        MD 0
      &END EACH
    &END RESTART_HISTORY
  &END PRINT
&END MOTION
```

```
&EXT_RESTART
  EXTERNAL_FILE SomeSystem-1.restart
&END
```

- EXT_RESTART section not required for ENSEMBLE REFTRAJ.

3.) Compress Volumetric Trajectories (Optional)

- Using the bqbtool, we can compress the CUBE trajectories
- Typical saving of space is factor 30 – 40 (for $\Delta t = 0.5\,\text{fs}$)

  → Gigabytes instead of Terabytes

  Takes ≈ 5 seconds per frame on 1 CPU core
- If you don‘t want to keep the electron density data, you can leave out this step

Command:

```
bqbtool compress cube result.cube result.bqb
```

or

```
travis compress cube result.cube result.bqb
```

4.) Solve current PDE, perform Voronoi Integration

For every atom in each trajectory frame, we want to compute the **e**lectro**m**agnetic **p**roperties („EMP“). These include:

- Electric dipole vector *(for IR, Raman, VCD, ROA)*
- Electric quadrupole tensor *(for ROA)*
- Electric current vector *(for VCD, ROA)*
- Magnetic dipole vector *(for VCD, ROA)*

This gives one EMP file per trajectory.

Takes around 1 second per frame *(only electric moments)*
or around 15 seconds per frame *(magnetic moments required)*.

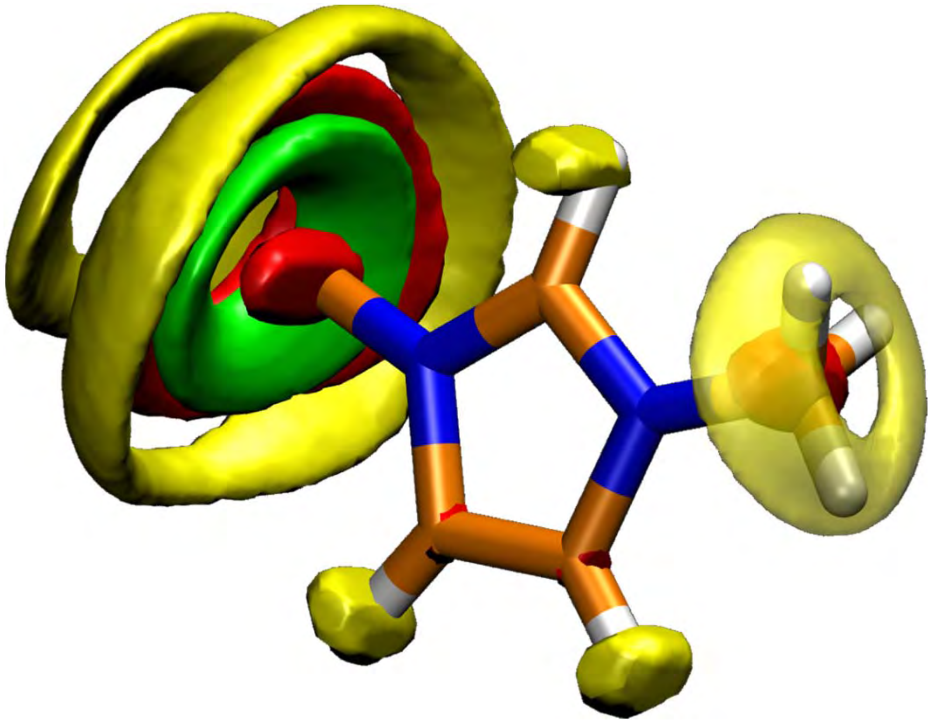
If we have trajectories with external electric field,
we can compute the polarizabilities by finite differences.

5.) Compute spectra from EMP property files

Supply the single field-free EMP file *(for IR, VCD)*
or the set of four EMP files *(for Raman, ROA)* to TRAVIS.

Computation of spectra from EMP files
takes ≈ 1 – 2 minutes in total ☺

All computationally demanding parts
have already been performed before.



Thank you for your attention!