

Do-it-yourself programming

Konrad Hinsen

`hinsen@cnr-orleans.fr`

Centre de Biophysique Moléculaire, CNRS Orléans

and

Synchrotron Soleil, Saint Aubin

Biomolecular simulations

Standard tasks:

- Run an MD simulation
- Calculate atomic fluctuations
- Visualize trajectory

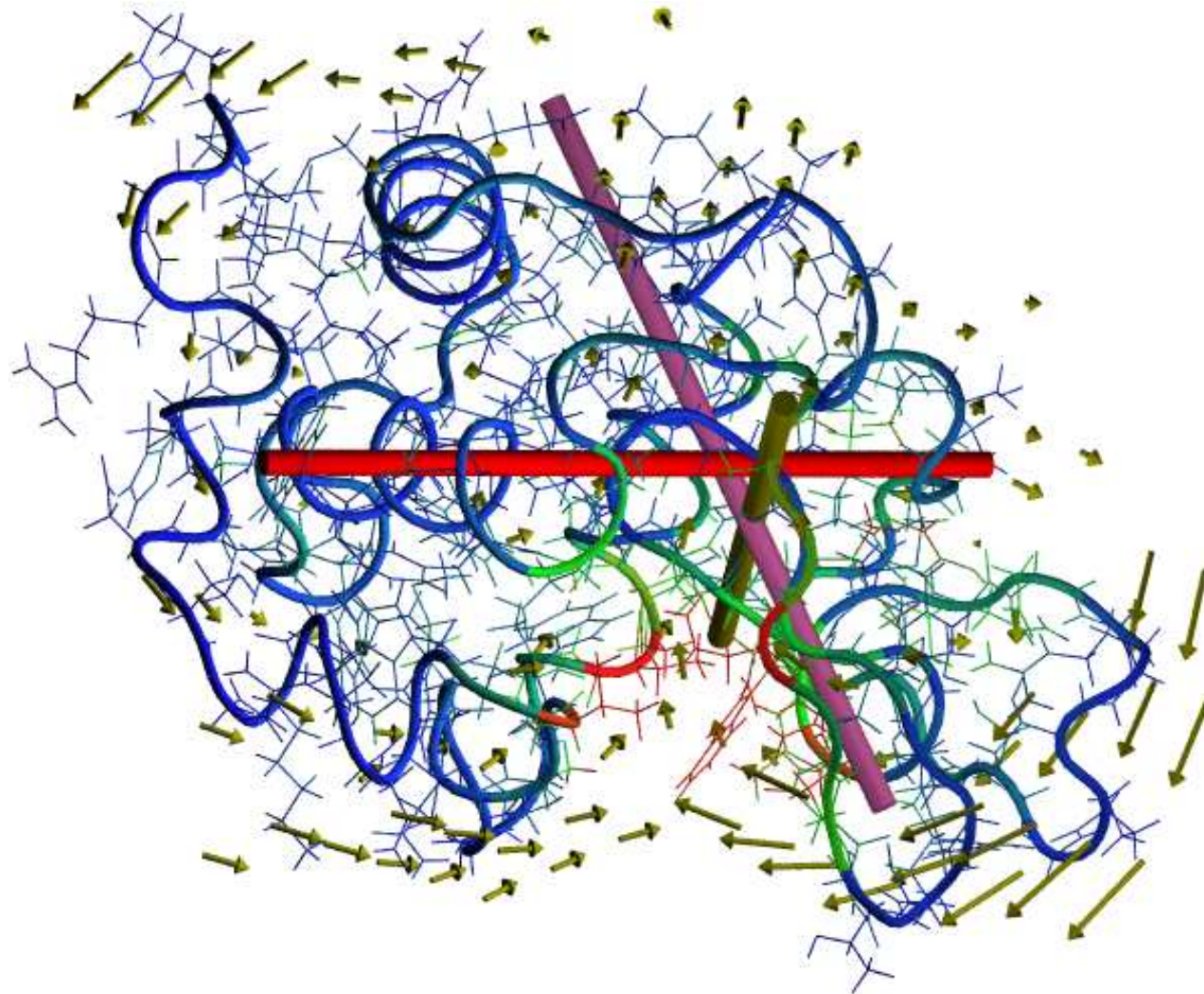
⇒ standard programs

Nonstandard tasks:

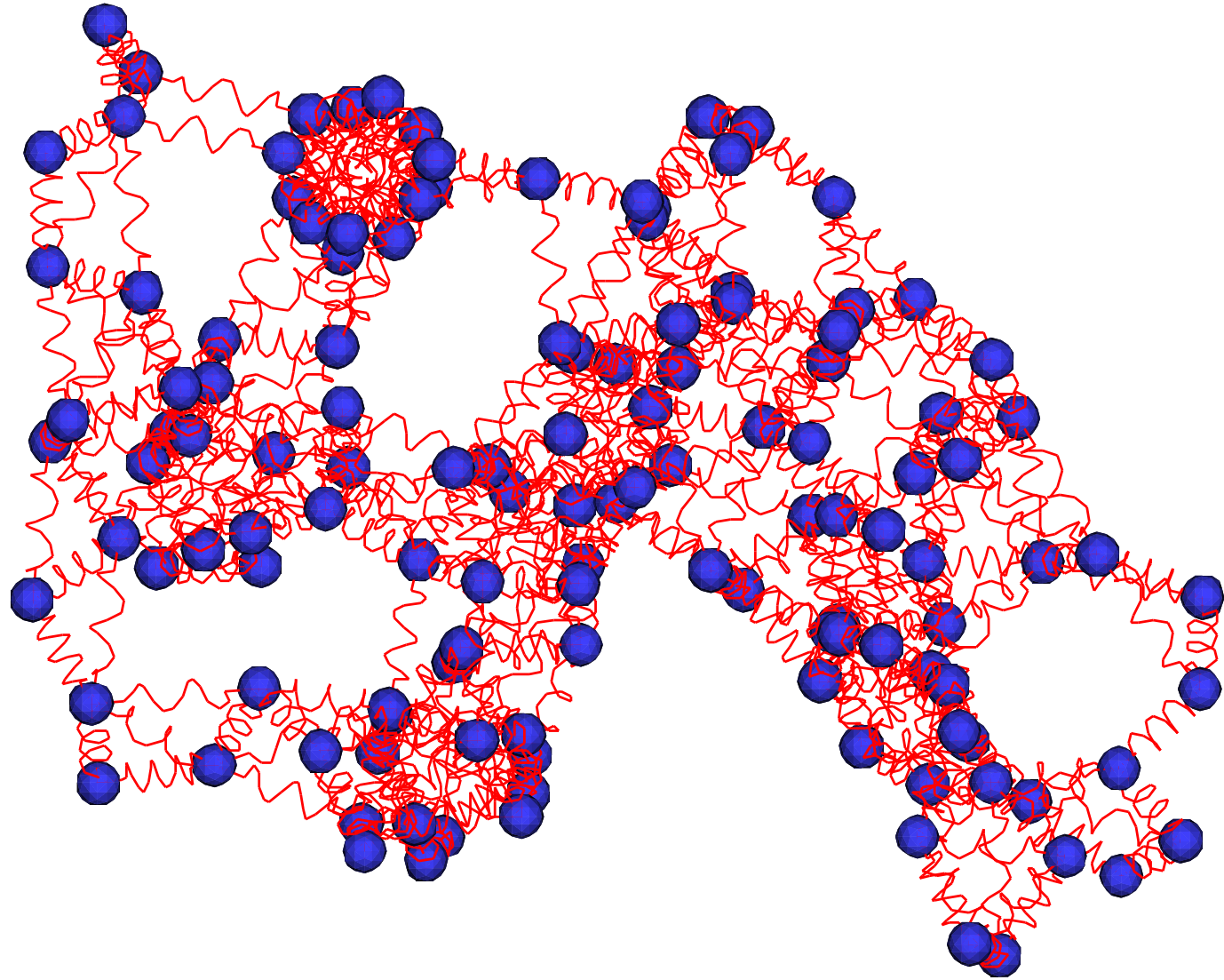
- Automate a multistep operation
- Extract data from a file
- File format conversion
- Implement new algorithms

⇒ **do-it-yourself programming**

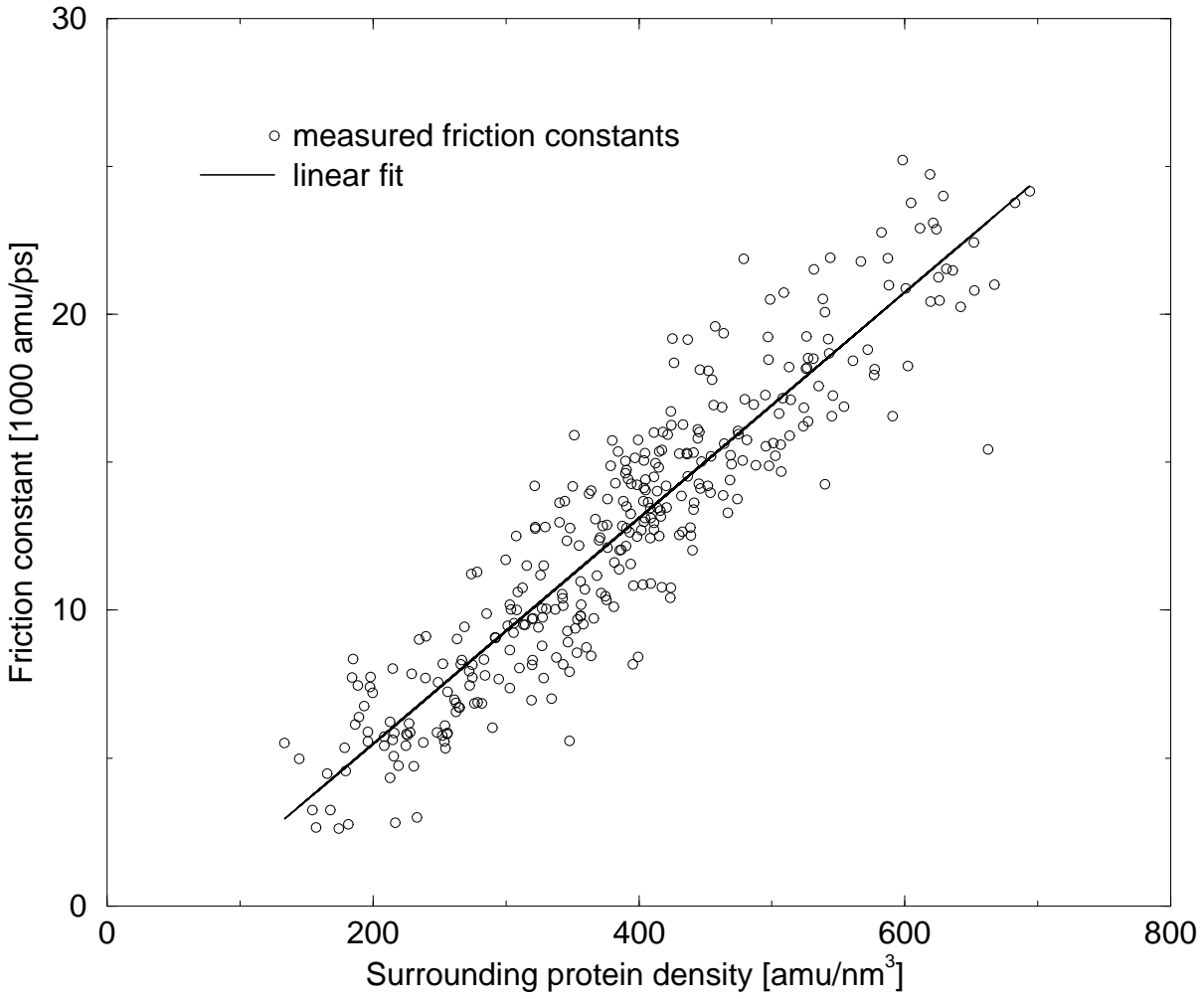
Visualization



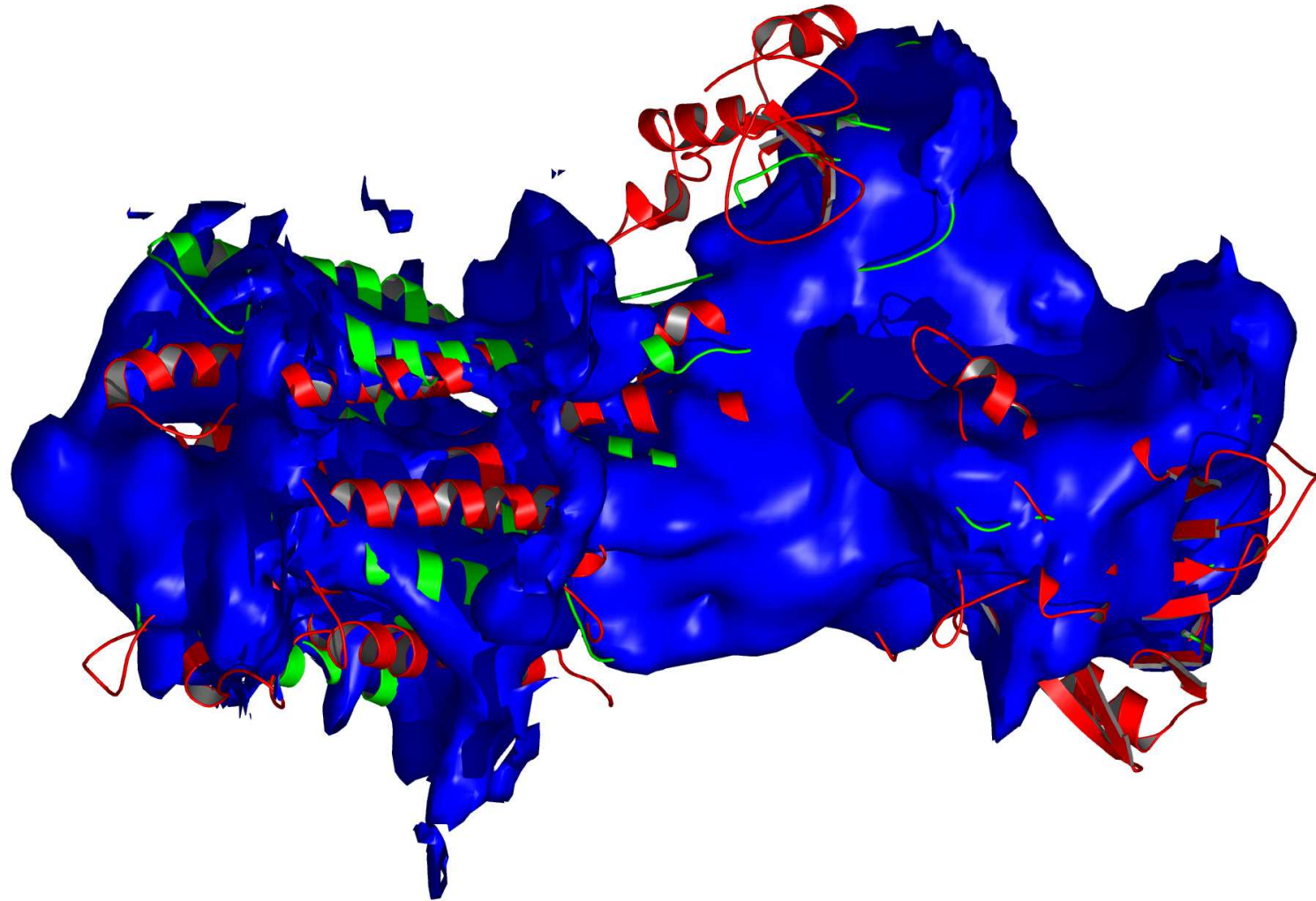
Visualization



Trajectory analysis



Method development



Programming options

Shell scripts:

- quick to write
- work on any (Unix) system
- very limited possibilities
- difficult to read
- very slow execution

Suitable for: automatization of simple operations

Programming options

Low-level languages (C, C++, Fortran, . . .):

- fast execution
- good development tools
- require significant experience
- long development times
- can be difficult to read

Suitable for: time-critical algorithms

Programming options

High-level languages (Perl, **Python**, Ruby, . . .):

- complete programming languages
- easy to learn
- fast development
- easy to read
- interface to low-level languages
- slow execution

Suitable for: everything that is not time-critical

Programming options

Empirical rule: 90% of a program is *not* time critical

Therefore:

- Everyone should know and use a high-level language.
- Developers of numerical methods must also know a low-level language.

Python

Features:

- Clean syntax
- Object-oriented
(→ problem-oriented data structures)
- Good low-level language interface
(C, C++, Fortran)
- Free and portable implementation
- Well established as a scripting language for computational science

Python

Examples:

- Sum up numbers in the third column of a text file

```
sum = 0.  
for line in file('my_data'):  
    sum = sum + float(line.split()[2])  
print sum
```

Python

- Calculate the radius of gyration of a protein

```
from MMTK.Proteins import Protein
import numpy

protein = Protein('protein.pdb')
center = protein.centerOfMass()
r_sq = 0.
for atom in protein.atomList():
    mass = atom.mass()
    distance = atom.position()-center
    r_sq = r_sq + mass*distance*distance
r_sq = r_sq/protein.mass()
print numpy.sqrt(r_sq)
```

Molecular Modelling Toolkit

Molecular simulation library for Python

- Standard operations: Molecular Dynamics, energy minimization, normal modes, molecular surfaces, . . .
- Lower-level basic operations for implementing your own methods
- Interfaces to visualization software
- Can be combined with other Python libraries
- Open Source

<http://dirac.cnrs-orleans.fr/MMTK>

Practical

Molecular simulation with MMTK

- Introduction to MMTK
- Normal modes of a water molecule
- Normal modes of a protein fragment
- Molecular Dynamics of a protein fragment
- Trajectory analysis