XPLOR-NIH: Recent Developments

Charles Schwieters and John Kuszewski

December 17, 2002

outline

1. description, history
2. Scripting Languages: XPLOR, Python, TCL
3. IVM: dynamics and minimization in internal coordinates
4. Dipolar Coupling potentials
5. Parallel determination of multiple structures
6. VMD molecular graphics interface

What is XPLOR-NIH?

Biomolecular structure determination/manipulation

- Determine structure using minimization protocols based on molecular dynamics/ simulated annealing.
- potential energy terms based on input from NMR (and X-ray) experiments: NOE, dipolar coupling, chemical shift data, etc.
- other potential terms enforce reasonable covalent geometry (bonds and angles).
- knowledge-based potential terms incorporate info from structure database.

includes: program, topology, covalent parameters , potential energy parameters, data for knowledge-based potentials.
[in the future: protocols.]

What XPLOR-NIH is not

Not general purpose molecular dynamics engine. Major deficiency: no Ewald summation for long-range electrostatic potentials. Use CHARMM, Amber, or NAMD.

X-Ray tools are dated. CNS X-ray facilities are more up-to-date.

Not an NMR spectrum analysis tool.
[future: tighter integration with tools such as NMRWish.]

Automatic NOE Assignment:

- XPLOR-NIH includes some support for Aria.
- [future: more complete job of assignment.]

XPLOR-NIH history, distribution

Should I be using CNS or CNX?

(about 1985) XPLOR branched off of CHARMM molecular dynamics code.

(about 1998) CNS is XPLOR renamed with continued X-ray development. Code and scripting language very similar to those of XPLOR. New NMR-related work done on XPLOR. Code easily ported back and forth between XPLOR and CNS.

(about 2000) official development of CNS is discontinued. MSI combines CNS and XPLOR into CNX.

(2002) NIH obtains noncommercial redistribution rights to XPLOR. XPLOR-NIH contains all new development done at NIH, plus contributions from elsewhere.

XPLOR-NIH available from http://nmr.cit.nih.gov/xplor-nih new releases announced on mailing list xplor-nih-announce@nmr.cit.nih.gov.

Scripting Languages- three choices

scripting language:

flexible interpreted language

used to input filenames, parameters, protocols

so recompilation is not necessary

XPLOR language:

strong point:

selection language quite powerful.

weaknesses:

String, Math support problematic.

no support for functions/subroutines.

Parser is hand-coded in Fortran: difficult to update.

NOTE: all old XPLOR scripts should run unchanged in XPLOR-NIH

general purpose scripting languages: Python and TCL

- excellent string support.
- languages have functions: can be used to better encapsulate protocols (e.g. call a function to perform simulated annealing.)
- well known: easier to write scripts.
- Facilitates interaction with other tools. e.g. NMRWish has a TCL interface. [future: we plan on interfacing our programs to merge spectrum analysis, assignment, and structure determination.]

separate processing of input files (assignment tables) is unnecessary: can all be done using XPLOR-NIH.

New development in C++: scripting interfaces (semi-)automatically generated using a tool called SWIG.

Script Examples: string processing

composing an output filename which includes the predefined integer, `count`.

`1gb1_anneal_1.pdb`

XPLOR:
```
evaluate ($file = "1gb1_anneal_" + encode($count) + ".pdb")
```

Python:
```
file = "1gb1_anneal_%d.pdb" % count
```

TCL:
```
set file [format "1gb1_anneal_%d.pdb" $count]
```

print out names/ residue numbers of C_α

XPLOR

```
vector identify (store9) (name ca)
foreach i in id (store9) loop main
  vector show elem (name) (id $i)
  eval ($curName = $result)
  vector show elem (resid) (id $i)
  eval ($curNum = $result)
  display "name" $curName "res num" $curNum
end loop main
```

TCL

```
AtomSel aSel "name ca"
foreach i [aSel indicies] {
  set curName [xplorSim getAtomName $i]
  set curNum [xplorSim getResidueNum $i]
  puts [format "name %s res num %d"
                        $curName $curNum]
}
```

Python

```
aSel = AtomSel("name ca")
for atom in aSel:
  print "name %d res num %d" % ( atom.atomName(),
                                 atom.residueNum() )
```

features of XPLOR-NIH scripting

- potential terms may be written in GPSL.
- inter-language calls available, with data passing

XPLOR

```
python
from rdcPot import RDCPot1
python_end
tcl
  info commands
returnToXplor
```

TCL

```
XplorCommand "param @$paramFileName end"
package require pyinterp
PyInterp pyth
pyth command "import ivm"
pyth command "integrator=IVM()"
```

Python

```
xplor.command('''struct @1gb1.psf end
                 coor @1gb1.pdb''')
tcl = TCLInterp()
tcl.command('xplorSim setRandomSeed 778')
```

The IVM (internal variable module)

in biomolecular NMR structure determination, many internal coordinates are known or presumed to take usual values:

- bond lengths, angles.
- aromatic amino acid sidechains
- nucleic acid base regions
- non-interfacial regions of protein and nucleic acid complexes (component structures may be known- only interface needs to be determined)

Can we take advantage of this knowledge (find the minima more efficiently)?

- can take larger MD timesteps (without high freq bond stretching)
- configuration space to search is smaller:
  $N_{\text{torsion angles}} \sim {}^{1}/_{3} N_{\text{Cartesian coordinates}}$
- don't have to worry about messing up known coordinates.

MD in internal coordinates is nontrivial

Consider Newton's equation:

$$F = Ma$$

for MD, we need a, the acceleration in internal coordinates, given forces F.
Problems:

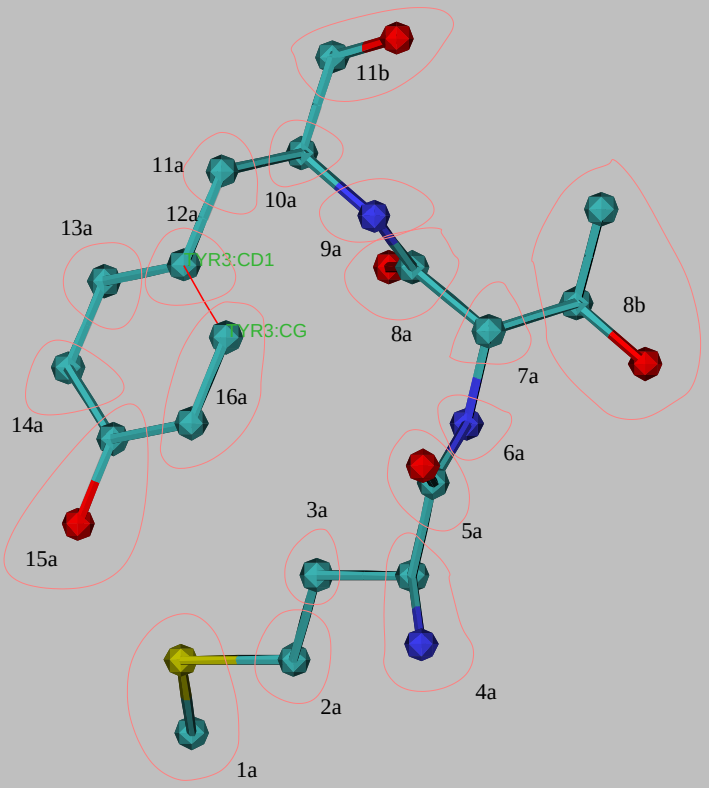
- express forces in internal coordinates
- solve the equation for a.

In Cartesian coordinates a is (vector of) atomic accelerations. M is diagonal.

In internal coordinates M is full and varies as a function of time: solving for a scales as $N^3_{\text{internal coordinates}}$.

Solution: comes to us from the robotics community. Involves clever solution of Newton's equation: The molecule is decomposed into a tree structure, a is solved for by iterating from trunk to branches, and backwards.

Tree Structure of a Molecule



atoms are placed in rigid bodies, fixed with respect to each other.

internal coordinates are described by θ_k.

rings and other closed loops are broken- replaced with a bond.

Topology Setup

torsion angle dynamics with fixed region:

XPLOR:

```
dynamics internal
   [group rigid sidechains]
   auto torsion
   fix (resid 100:120)
end
```

Python:

```
from ivm import IVM
from selectTools import groupRigidSideChains
integrator = IVM()
integrator.group( groupRidigSideChains() )
integrator.autoTorsion()
integrator.fix( AtomSel("resid 100:120") )
```

IVM Implementation details:

other coordinates also possible: e.g. mixing Cartesian, rigid body and torsion angle motions.

convenient features:

- variable-size timestep algorithm
- will also perform minimization
- facility to treat bonds which cause loops in tree.

Examples of use are given in the XPLOR-NIH eginputs subdirectory.

dynamics with variable timestep

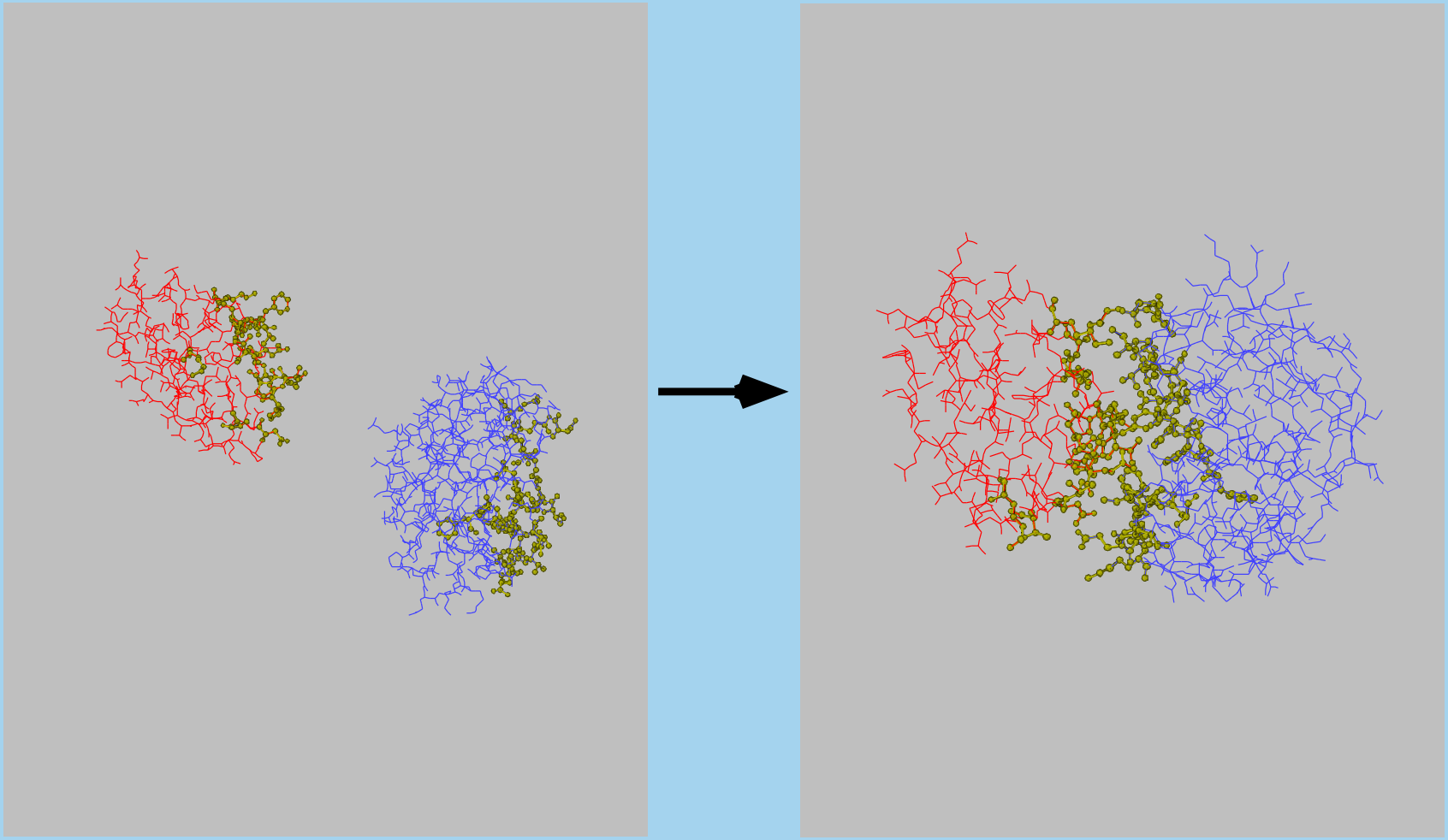
XPLOR:

```
dynamics internal
  itype=pc6
  endt=1
  etol=.1
  nprint=10
  tbath=2000
end
```

Python:

```
bathTemp=2000
integrator.setStepType("pc6")
integrator.setFinalTime(1)
integrator.setETolerance( bathTemp/1000 )
integrator.setPrintInterval(10)
integrator.setBathTemp( bathTemp)
```

Hierarchical Refinement of the Enzyme II/ HPr complex



active degrees of freedom are displayed in yellow.

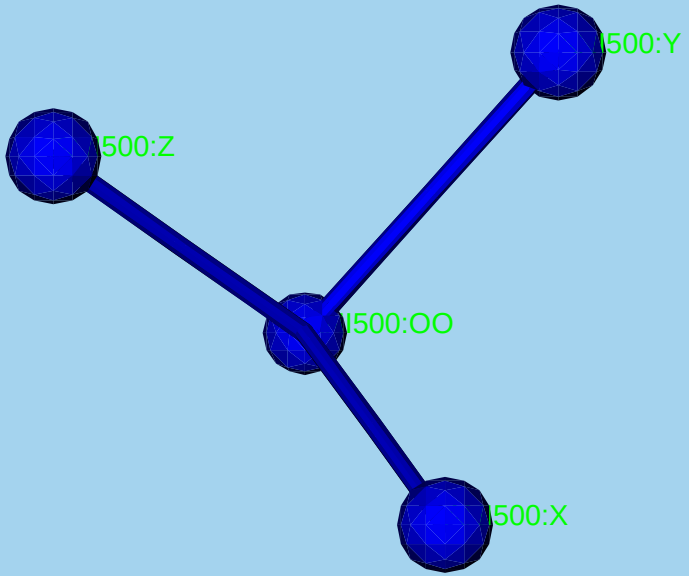
residual dipolar coupling potential terms

Provides orientational information relative to axis fixed in molecule frame.

$$\delta_{\mathsf{calc}} = \delta_{\mathsf{DFS}} + D_a[(3u_z^2 - 1) + \frac{3}{2}R(u_x^2 - u_y^2)] \ ,$$

u_x, u_y, u_z- projection of bond vector onto axes of tensor. D_a, R- related to magnitude of tensor components.

Axis represented by four atoms:

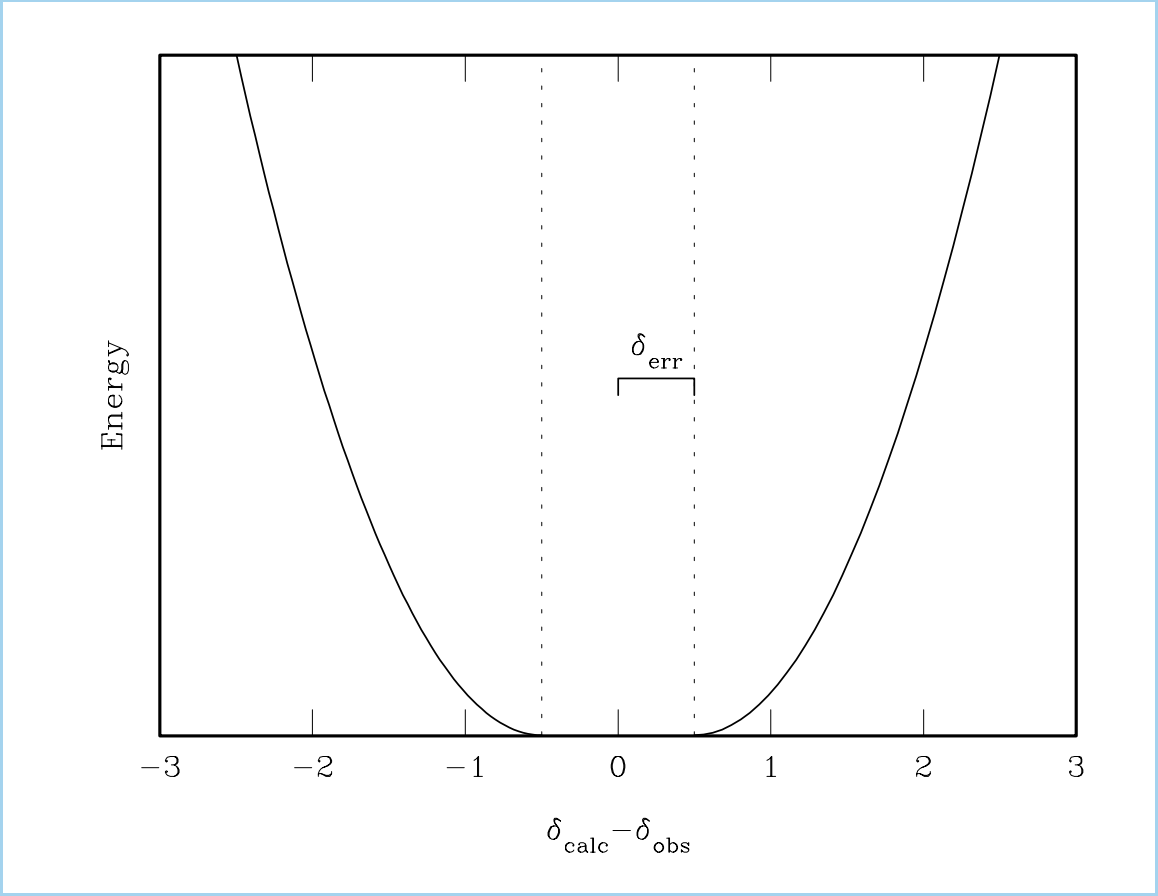


XPLOR-NIH potential terms

XPLOR SANI assignment statement

```
assign (origin selection) (z-atom selection)
       (x-atom selection) (y-atom selection)
       (m-atom selection) (n-atom selection) delta_obs delta_err
```

- specify the values of δ_{DFS}, D_a, R.
- if POTEntial=SQUAre, delta_err is used:



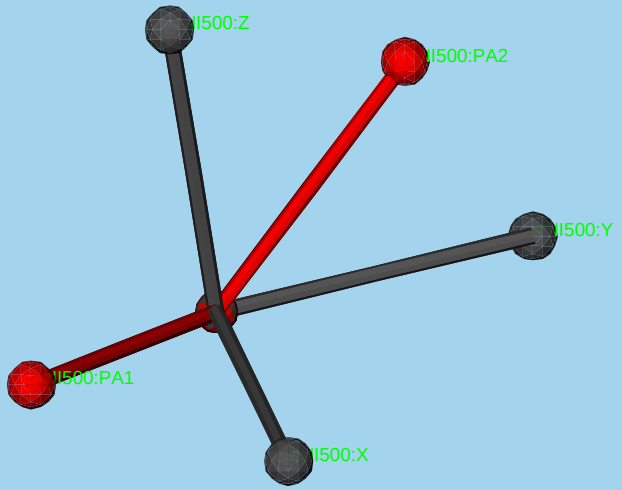
XPLOR-NIH potential terms

DIPO

- allows multiple assignment for bond-vector atoms
- allows ignoring sign of D_a (optional)
- can (optionally) include distance dependence: $D_a \propto 1/r^3$.
- ASSIgnment statement takes extra argument.

rdcPot (in Python)

- allows Da, R to vary: values encoded using extra atoms.
- reads both SANI and DIPO assignment tables.
- additional facilities to deal with multiple, related experiments (can restrain ratios of D_a to same value).



parallel computation of multiple structures

computation of multiple structures with different initial velocities and/or coordinates: gives idea of precision of NMR structure.

```
xplor -parallel -machines <machine file>
```

convenient XPLOR-NIH parallelization

- spawns multiple versions of xplor on multiple machines via ssh or rsh.
- structure and log files collected in the current local directory.

requirements:

- ability to login to remote nodes via ssh or rsh, without password
- shared filesystem which looks the same to each node

following environment variables set: `XPLOR_NUM_PROCESSES`, `XPLOR_PROCESS`

example XPLOR script

at beginning of script add:

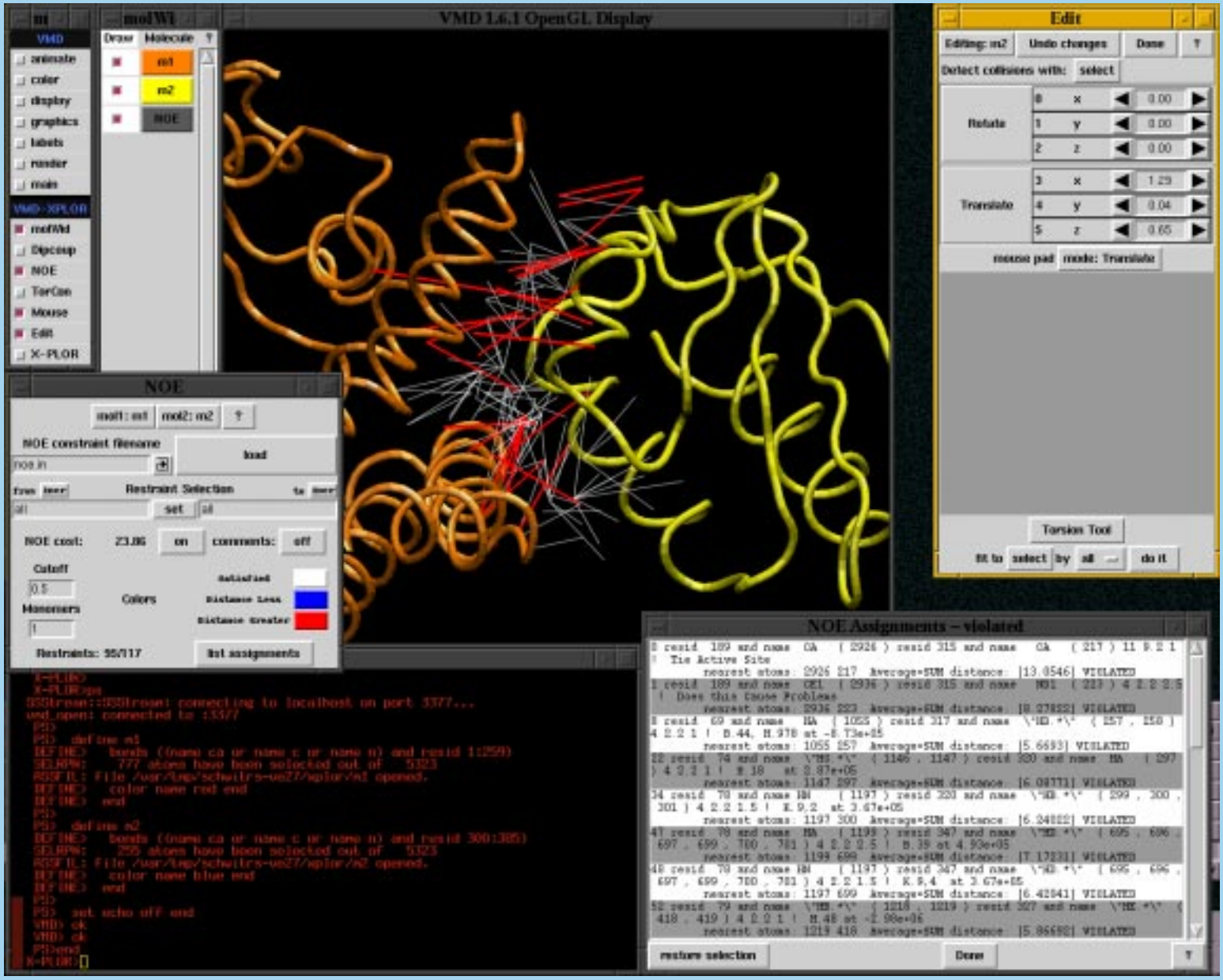
```
eval ($numStructs = 100)     !total number of structures to calculate
eval ($randomSeed = 785)     !random seed
!
! get parallel info
!
cpyth "from os import environ as env"
cpyth "xplor.command('eval ($proc_num=%s)'  % env['XPLOR_PROCESS']      )"
cpyth "xplor.command('eval ($num_procs=%s)' % env['XPLOR_NUM_PROCESSES'])"
eval ($firstStruct = ($proc_num * $numStructs) / $num_procs)
eval ($lastStruct  = (($proc_num+1)  * $numStructs) / $num_procs)
```

structure loop should look like:

```
eval ($count = $firstStruct)
while ($count < $lastStruct) loop structure
    eval ($seed = $randomSeed+$count)
    set seed $seed end
        .
        .
        .
    evaluate ($file = "1gb1_" + _ + encode($proc_num) + ".pdb")
    write coor output= $file end
    eval ($count = $count + 1)
end loop structure
```

example Biowulf PBS script: http://nmr.cit.nih.gov/xplor-nih/nih

VMD interface



vmd-xplor screenshot

Use VMD-XPLOR to

- visualize molecular structures
- visualize restraint info
- manually edit structures

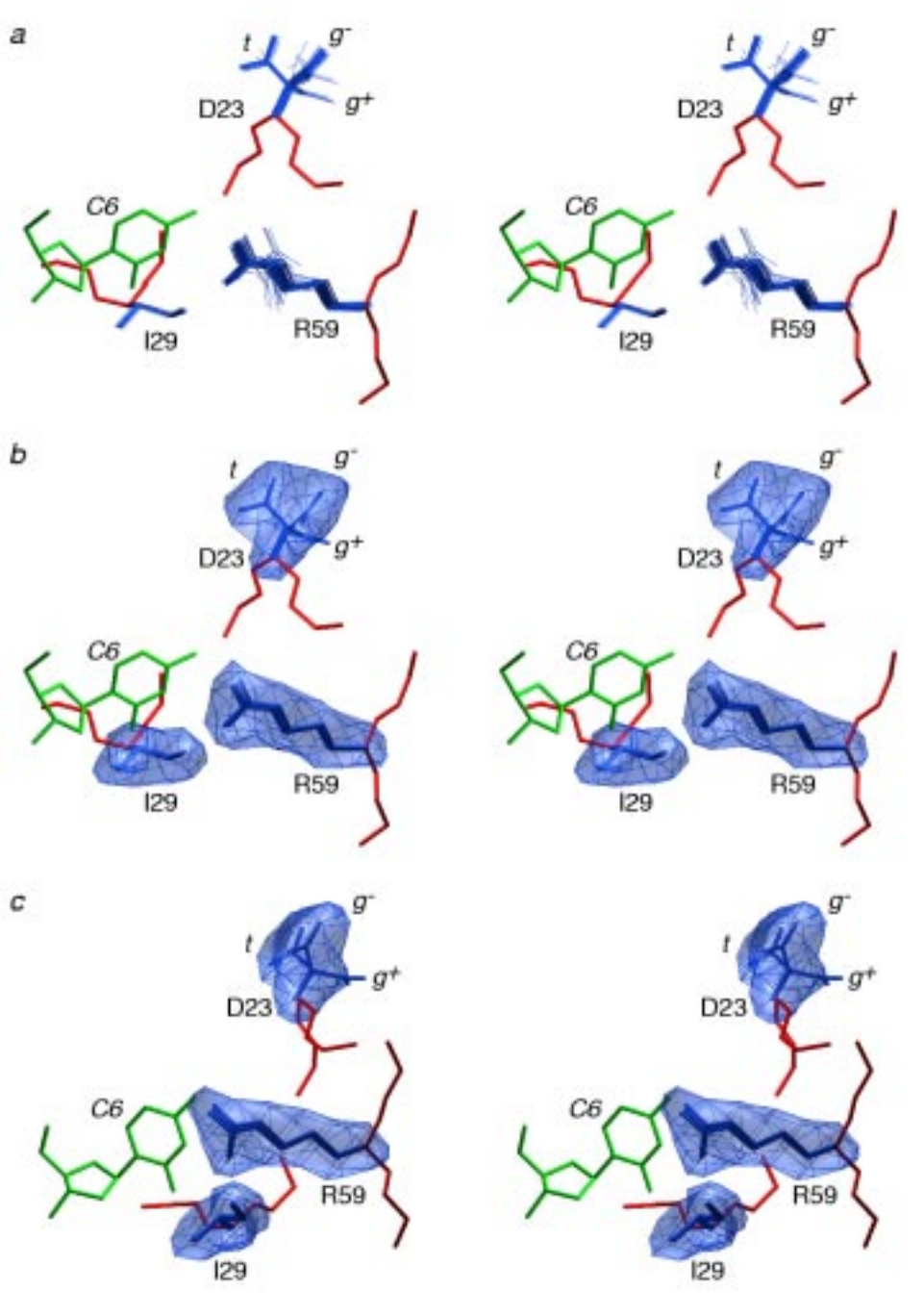
command-line invocation of separate XPLOR-NIH and VMD-XPLOR jobs:

```
% vmd-xplor -port 3359 -noxplor
% xplor -port 3359
```

XPLOR snippet to draw bonds between backbone atoms and labels:

```
ps
 define x bonds (name ca or name c or name n) end
 define label
   label size=2 resid=t resname=f name=f select (name ca)
  end
end
```

Graphical Representation of ensembles



Stereoviews illustrating various representations of the side chains of Asp23, Ile29 and Arg59 of the hn-RNPK KH3-ssDNA complex. (a) Superposition of the sidechains of 100 simulated annealing structures (blue). (b) and (c) Isosurface of the reweighted atomic density map for the three side chains drawn at a value of 20% of maximum; within the map, the coordinates of the sidechains of three representative structures are displayed in blue. The view shown in (b) is identical to that in (a). The coordinates of the protein backbone and the C6 nucleotide of the restrained regularized mean structure are shown in red and green, respectively.

Where to go for help

mailing list: xplor-nih@nmr.cit.nih.gov [post only, previous postings+answers available]

within the xplor distribution: complete example scripts can be found in the eginput and tutorial subdirectories

XPLOR language reference:
A.T. Brünger, "XPLOR Manual Version 3.1" (Yale University Press, New Haven, 1993).
http://nmr.cit.nih.gov/xplor-nih/xplorMan

Python:
M. Lutz and D. Ascher, "Learning Python," (O'Reilly, 1999).
http://python.org

TCL:
J.K. Ousterhout "TCL and the TK Toolkit" (Addison Wesley, 1994).
http://www.tcl.tk