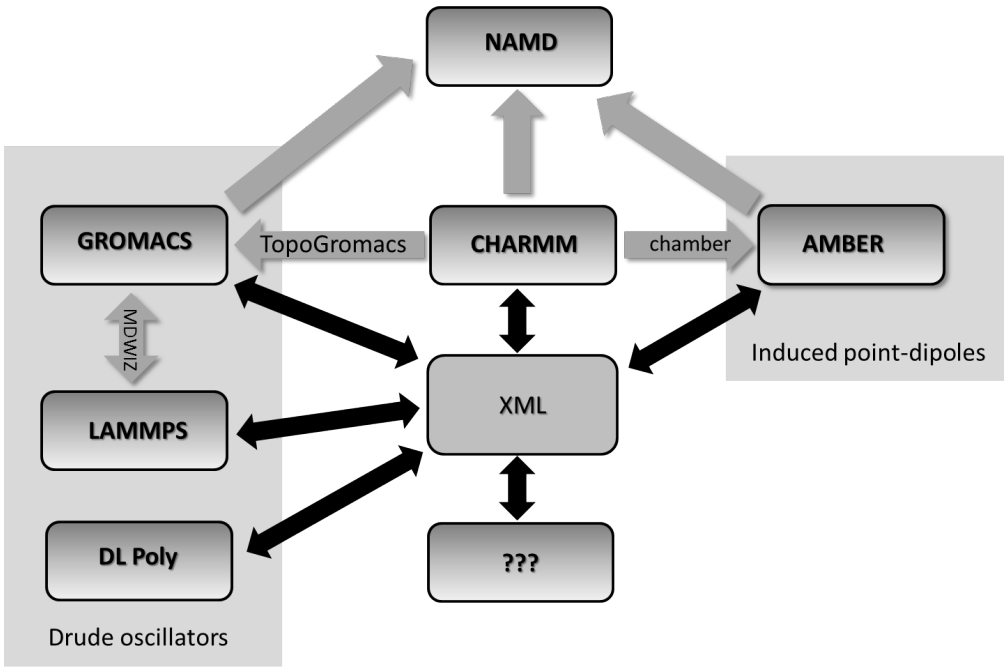
ForConX

A Force Field Conversion Tool Based on XML



Version 0.1

Contents

1 Introduction

ForConX is a Python code for conversion between force field files of different Molecular Dynamics (MD) simulation packages. Currently, conversion for the following MD programs is available:

- Amber
- Charmm
- Dl-Poly
- Gromacs
- Lammps

but can easily be extended to new MD programs by writing an interface to the central XML document. ForConX is free software, distributed under the terms of the GNU General Public License version 3, as published by the Free Software Foundation and included in the source code documentation. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. In no event the authors will be liable to you for damages, including any general, special, incidental or consequential damages (including but not limited arising out of the use or inability to use the program, to loss of data or data being rendered inaccurate, or losses sustained by you or third parties, or a failure of the program to operate with any other programs), even if the author has been advised of the possibility of such damages.

The features of ForConX are described in the following publication

> *"ForConX - A Forcefield Conversion Tool Based on XML"*
> V. Lesch, D. Diddens, C.E.S. Bernardes, B. Golub, A. Dequidt, V. Zeindlhofer,
> M. Sega, C. Schröder, *J. Comp. Chem.* **2017**, (submitted).

Please cite if you are using this tool.

1.1 The central idea

A run of ForConX consists of three phases as visible in Fig. 1. In the first phase the force field files of a MD program are translated to a XML document (for details on the XML format see Section 4.4). During the second phase, the XML document is checked for viability. The last phase concerns the conversion from the XML document to the force field files of the target MD program.

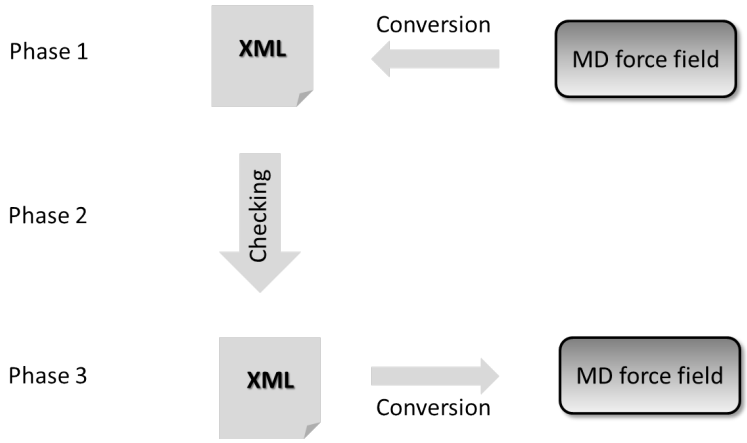


Figure 1: *The ForConX conversion process.*

The conversion of a force field from one MD program to another may necessitate several runs of ForConX. During the first run, the XML document is augmented by the information ForConX extracted from the original MD force field files. If the program terminates because of insufficient information, the user may add, remove or modify particular lines in the XML document and re-run Phase 2 and 3, starting from the XML-file and not again from the original MD force-field file. In any case, Phase 2 is always part of a ForConX run. Here, all XML lines are checked for completeness and it is verified that the information in the parameter and topology sections is compatible for every molecule.

Making a new MD program available in ForConX is realized by writing an interface that transforms the information from the desired MD program force field files to the XML structure

and vice-versa. It is not necessary to write pairwise conversion tools between the MD programs.

2 Installation and running

To install ForConX, Python 2.x must be present in the operating system. If you need to install python, please visit `https://www.python.org`.

2.1 Installation

Please unzip the source code in a directory and change to the ForConX directory. The file `setup.py` should be present in that directory. The program can be compiled using the standard python method:

```
$ python setup.py install --user
```

This will place the executable (in principle) in the standard directories

```
.local/bin
```

or

```
~/Library/Python/X.X/bin
```

in case of MacOs. Now, you should be able to execute ForConX in any directory on your system.

2.2 Running

To run the program, type:

```
$ forconx inputfile.xml
```

where `inputfile.xml` is a XML file containing the conversion information (see below for details).

3 XML file

The XML file serves multiple purposes. Before the conversion, the user specifies the desired input and output format in the input/output section of the XML. During the ForConX run, the XML is used as "real-time" storage. The XML input file can be composed by several elements that are included in a <ForConX> ... </ForConX> section. For any run, the xml file must contain <input>, <output> and <molecule> (one for each molecule) elements.

```
<?xml version="1.0" ?>
<ForConX>
    <input md="MD program 1">
    ...
    </input>

    <output md="MD program 2">
    ...
    </output>

    <molecule name="MOLNAME1" nmol="..."/>
    ...
</ForConX>
```

The remaining sections are facultative and include the details of the bonded and nonbonded force field parametrizations and will be filled when this information is retrieved from the initial input file.

Currently, the MD programs Amber, Charmm, Dl-Poly, Gromacs and Lammps are supported. Alternatively, one may start the construction of a force field file directly from the XML structure.

During the conversion process, ForConX writes a seperate XML document after the completion of each phase: forconx_1.xml, forconx_2.xml and forconx_3.xml. These files can be used to add information and re-run ForConX.

3.1 <input> ... </input> or <output> ... </output>

In the <input> ... </input> section the user specifies the starting MD program and files that should be converted. The target format is declared in the <output> ... </output> section. The <input> ... </input> section has the same keywords as <output> ... </output> except for the citation defined in <reference> ... </reference>:

```
<input md="XML">
   <energy   unit="KCAL"      />
   <distance unit="ANGSTROEM" />
   <coordinates pdb="..." />
   <reference>
    <title> ... </title>
    <author> ... </author>
    <journal> ... </journal>
    <volume> ... </volume>
    <year> ... </year>
    <pages> ... </pages>
   </reference>
   ...
 </input>
```

Energy, distance and polarizability unit are usually defined by the MD program. For example, CHARMM explicitly uses kcal/mol, Å and $Å^3$. Consequently, if md="CHARMM", FORCONX automatically overwrites the unit definitions with the default values mentioned above. Other programs allow for several units, *e.g.* DL-POLY may use kcal/mol, kJ/mol or even K as energy unit. Also when starting from a XML structure, the units have to be specified with the options tabled in Tab. 1. During the conversion process, the unit definitions of <input> ... </input> and <output> ... </output> are used to convert the force field parameters into the units typical for the target MD program.

The force field file structure differs for each MD program. Sometimes several files are needed to setup a force field. MD program specific parts are given in Tab. 2.

Table 1: *Units used in the input/output sections.*

Element	**Key**	Unit
<energy unit="KEY"/>	KCAL	kcal/mol
	KJ	kJ/mol = 0.239 kcal/mol
	K	Kelvin
<distance unit="KEY"/>	ANGSTROEM	Å
	NM	nm = 10 Å
<polarizability unit="KEY"/>	ANGSTROEM	$Å^3$
	NM	nm^3 = 1000 $Å^3$
	BOHR	a_0^3 = (0.529 Å)^3

Table 2: *List of elements that can be used in the input and output sections for the different MD simulation programs supported by ForConX.*

MD Program	**Element**
AMBER	<lib file="FILE.lib"/>
	<frcmod file="amber_new.frcmod"/>
CHARMM	<topology file="FILE.rtf"/>
	<parameter file="FILE.prm"/>
	<crd file="FILE.crd"/>
DLPOLY	<field file="FIELD"/>
	<config file="CONFIG" />
GROMACS	<top file="FILE.top"/>
LAMMPS[a]	<command file="input.lmp"/>
	<data file="data.lmp"/>
	<pair file="pair_coeffs.lmp"/>
	<molecule ext=".mol"/>

[a] only the element "command" is required to read the original LAMMPS input files

3.2 <molecule> ... </molecule>

To perform a MD program 1 → XML → MD program conversion, it is necessary to indicate each molecule to be converted as:

```
<molecule name="MOLNAME" nmol="..."/>
```

where "MOLNAME" is the name of the desired molecular species in the input files, followed by the number of molecules, "nmol", in the configuration file. It is possible to define several molecules

```
<molecule name="MOLNAME1" nmol="..."/>
<molecule name="MOLNAME2" nmol="..."/>
```

Here, the atom names within a particular molecule have to be unique but may be the very same in different molecules.

After phase 1 of conversion, ForConX augments the XML file which now looks like this:

```
<molecule name="..." nmol="...">
    <atom name="..." type="..." mass="..." charge="..."
          alpha="..." />
    <virtual name="..." type="..." charge="..."
             zmatrix="... ... ..." r="..."
             theta="..." phi="..." />
    <bond name="... ..." />
    ...
    <angle name="... ..." />
    ...
    <dihedral name="... ... ... ..." />
    ...
    <improper name="... ... ... ..." central="..." />
    ...
</molecule>
```

All atoms have a unique name, an atom type, a mass and a partial charge. The polarizability alpha is optional. Virtual atoms possess no mass but can be defined using a z-matrix style. r is the distance between the virtual atom and the first atom in zmatrix. $theta$ is the angle between the virtual atom and the first and second atom in zmatrix and phi the dihedral angle between

the virtual atom and all atoms defined in zmatrix. All bonds between atoms are defined by <bond> with alphabetically sorted atomnames. This bond information is very important since the autogeneration of angle and dihedrals rely on a complete definition of all bonds within a molecule. Furthermore, during the check of the XML structure, dihedral and impropers are only accepted, if all corresponding bonds are present.

Only bonds, angles, dihedrals and improper torsions defined in the <molecule>-section are taken into account for the conversion, even if additional potentials in the <bonds>,<angles>, <dihedrals> and <impropers> section exists.

3.3 <bonds> ... </bonds>

This section contains all force field parameters for bond potentials U_{bonds}. A particular bond potential can be defined as a harmonic or Morse potential (but not both):

$$U_{\mathrm{bonds}} = \sum_{\xi}^{\mathrm{bonds}} k_{\xi}(r - r_{\xi}^{0})^2 + \sum_{\xi}^{\mathrm{bonds}} D_{\xi}^{0}(1 - e^{\beta(r - r_{\xi}^{0})})^2 \tag{1}$$

Bonds are stored in the XML structure in the following way:

```
<bonds>
    <harm type="... ..." r0="..." k="..." />
    <mors type="... ..." r0="..." D0="..." beta="..." />
</bonds>
```

In contrast to the bonds in <molecule>, the identifiers are the atom types of the bonding partners (separated by a space, e.g. type="CL HC"). As mentioned before, harmonic bonds and Morse potentials cannot be defined for the very same type="... ...".

3.4 <angles> ... </angles>

ForConX knows harmonic angles and Urey-Bradley potentials:

$$U_{\text{angles}} = \sum_{\xi}^{\text{angles}} k_\xi(\theta - \theta_\xi^0)^2 + K_\xi(r_{jk}(\theta) - r_{jk}^0)^2. \tag{2}$$

```
<angles>
    <harm type="... ... ..." k="..." theta0="..." />
    <urey type="... ... ..." k="..." r0="..." />
</angles>
```

Again, they type identifier is composed of the types of the three atoms enclosing an angle, separated by spaces (e.g. type="HC CL HC"). Since Urey-Bradley potentials are used to introduce anharmonicity to harmonic angles, a particular angle can possess a harmonic AND an Urey-Breadley potential. Thus the type is not an unique identifier in this section.

3.5 <dihedrals> ... </dihedrals>

Dihedral potentials are represented as Cosine torsions or Ryckaert-Bellemans potentials in FORCONX:

$$U_{\text{dihedrals}} = \sum_{\xi}^{\text{dihedrals}} k_\xi\Big(1 + \cos(n_\xi\phi - \delta_\xi)\Big) + \sum_{\xi}^{\text{dihedrals}} \sum_{n=0}^{5} c_{\xi n} \cos^n(\phi)$$

```
<dihedrals>
    <cos  type="... ... ... ..." n="... ... ..."
          delta="... ... ..." k="... ... ..." />
    <ryck type="... ... ... ..." k="... ... ... ... ... ..."/>
</dihedrals>
```

However, these potentials are mutually exclusive and hence have unique type identifiers (which again are composed of the four atom types comprising the dihedral, e.g. type="HC CL CL HC").

3.6 <impropers> ... </impropers>

Improper torsions are represented as cosine torsions, Ryckaert-Bellemans or harmonic angle potentials:

$$U_{\text{impropers}} = \sum_{\xi}^{\text{impropers}} k_\xi \Big(1 + \cos(n_\xi \phi - \delta_\xi)\Big) + \sum_{\xi}^{\text{impropers}} \sum_{n=0}^{5} c_{\xi n} cos^n(\phi) + \sum_{\xi}^{\text{impropers}} k_\xi (\theta - \theta_\xi^0)^2$$

```
<impropers>
    <cos delta="... ... ..." k="... ... ..." n="... ... ..."
         type="... ... ... ..."/>
    <ryck type="... ... ... ..." k="... ... ... ... ... ..."/>
    <harm type="... ... ... ..." k="..." theta0="..."   />
</impropers>
```

Again, these potentials are mutually exclusive and hence have a unique atom identifier, comprised of types of the four atoms involved (e.g. type="CR NA NA HA"). For impropers, the atom sequence is different in various MD programs. Therefore, the central atom of the improper is stored in the corresponding improper subelement of the <molecule> section, e.g:

```
<molecule name="IM21" nmol="1">
    ...
    ...
    <improper central="NA" name="NA CR CW C1" type="NA CR CW C1
       "/>
</impropers>
```

.

In the <impropers> section, the first atom type in the identifier is the central atom.

3.7 <nonbonded> ... </nonbonded>

As partial charges are already defined in the atom-entries of <molecule>, only van-der-Waals interaction parameters are stored in <nonbonded>. FORCONX can describe only standard Lennard-Jones potentials:

$$U_{\text{vdWs}} = \sum_{\xi} \sum_{\xi<\lambda} 4\epsilon_{\xi\lambda} \left(\left(\frac{\sigma_{\xi\lambda}}{r_{\xi\lambda}}\right)^{12} - \left(\frac{\sigma_{\xi\lambda}}{r_{\xi\lambda}}\right)^{6} \right) \tag{3}$$

```
<nonbonded>
    <mixing_rules epsilon="geometric" sigma="arithmetic"/>
    <atom type="... ..." epsilon="..." sigma="..."
          vdw14="..." elec14="..." />
      ...
    <vdw type="... ..." epsilon="..." sigma="..."
         vdw14="..." elec14="..." />
      ...
</nonbonded>
```

"Geometric" and "arithmetic" define the method of averaging and affects <atom> only when pairwise Lennard-Jones interactions are computed. For Lennard-Jones pairs in <vdw> the mixing rules are obsolete. Special caution is necessary when 1-4 interactions are concerned, as they are defined in different ways in different force fields: AMBER, DL-POLY and GROMACS store this information with the dihedral angle, whereas CHARMM possesses an extra non-bonded section. For flexibility reasons, 1-4-scaling factors for Coulomb (elec14) and Lennard-Jones (vdw14) interactions can be defined seperately for each <atom> and <vdw>. However, for an atom pair $\xi\lambda$ in <vdw> connected by a dihedral angle, only $\epsilon_{\xi\lambda}$ is scaled, but not $\sigma_{\xi\lambda}$.

4 Developers guide

ForConX is written in Python 2. An overview of all ForConX code files and where to find them is given in Fig. 2. The ./potentials directory contains python files for the energy potentials used. All core routines of ForConX are stored in the ./md_xml directory and should not be changed. In addition to these directories, each MD program has its own directory, *e.g.* ./charmm.

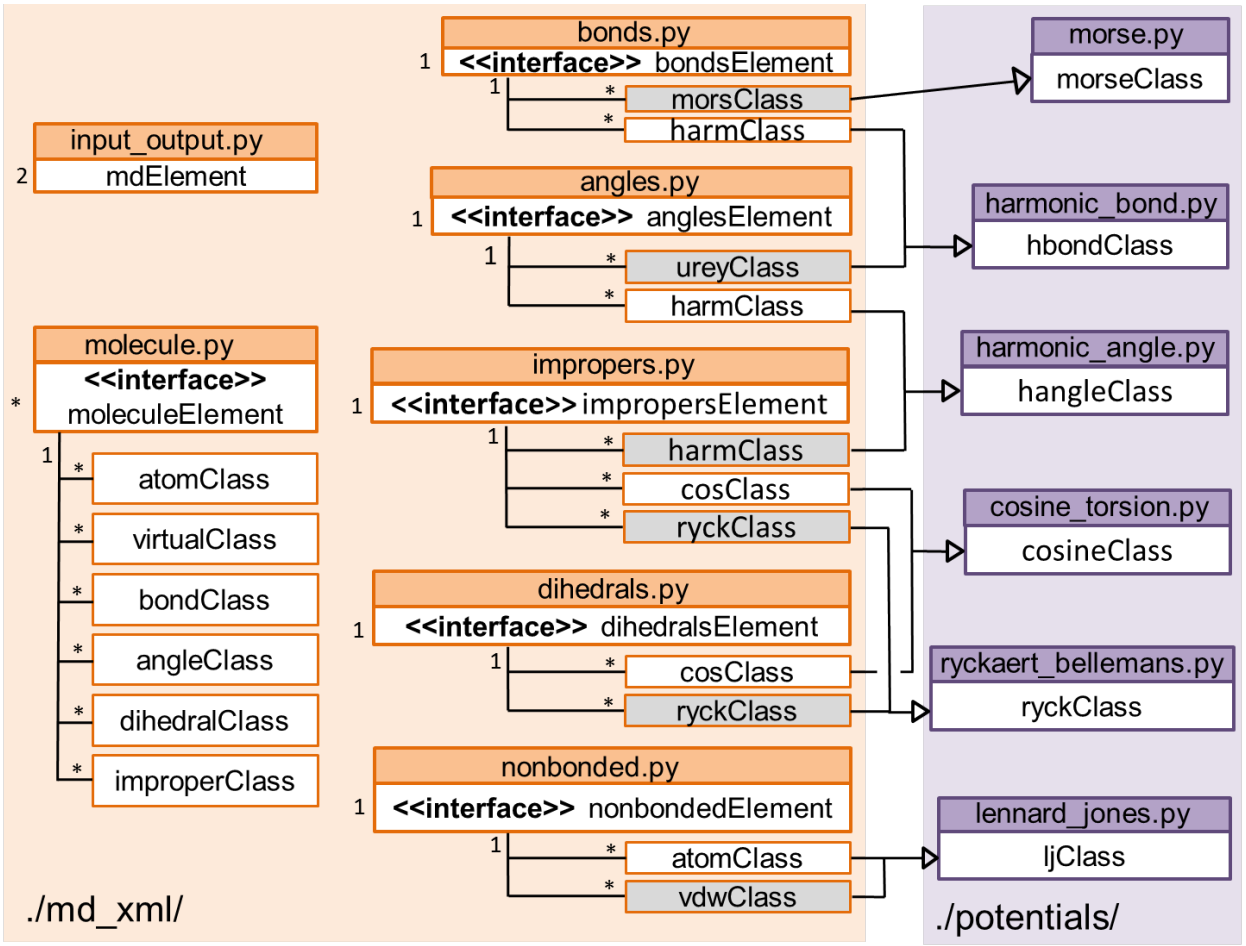


Figure 2: *Python classes in* ForConX *and where to find them.*

4.1 ./md_xml

The core routines of ForConX are in this directory:

- input_output.py
- molecule.py
- bonds.py
- angles.py
- dihedrals.py
- impropers.py
- nonbonded.py

Each of these files is responsible for a particular segment of the XML document which can be deduced from their names. Except input_output.py, these Python files contain an ≪interface≫ to all classes which is also defined in the same Python file. Some of these classes have a parent class defined in ./potentials.

4.2 ./potentials

This directory contains Python files for each energy potential used in ForConX. Overloading ensures that not only a value of a particular property can be set but also that the XML structure is immediately updated.

4.3 MD program

Each MD program has its own directory which contains at least three files: md2xml.py and xml2md.py for the force field conversion and xyz.py for the production of a coordinate file on a basis of a temporary forconx.pdb. Consequently, xyz.py is able to convert the MD specific coordinate file to a PDB file and vice versa. md2xml.py and xml2md.py convert the force field files using the Python objects described in the next section.

4.4 XML structure

ForConX uses the xml module of Python which is documented at

```
https://docs.python.org/2/library/xml.etree.elementtree.html
```

root is the root element of the XML tree. The present molecules can then be found in the following way:

```
import xml.etree.ElementTree as ET
import sys

xmlfile = sys.argv[1]
root =  ET.parse(xmlfile).getroot()
for mol in root.findall('molecule'):
  molname = mol.get('name')
```

4.5 Elements and Classes

In order to gain access to all properties of the atom and virtual atoms of that molecule, molecule.py offers moleculeElement as ≪interface≫ class:

```
from ..md_xml import molecule

current_molecule = molecule.moleculeElement(mol, molname)
for atomname in current_molecule.list("ATOM VIRTUAL"):
  print atomname
```

The atomname and the pointer of the respective molecule, mol, are sufficient to get access to the atom and its properties:

```
current_atom = molecule.atomClass(mol,atomname)
print current_atom.type
print current_atom.charge
print current_atom.mass
```

These properties can also be changed:

```
current_atom.charge = -0.5
```

which results not only in the assignment of the new partial charge but also updates the current XML structure thanks to overloading. Please use these classes to convert your MD force field to the XML and back. The overwhelming rest of your code is formatted reading and writing the force field files.