Concerted rotations

Author : Luca Tubiana, luca.tubiana@univie.ac.at
Last revision : 04.06.2018

This notebook implement the equations needed to compute a concerted rotation of 7 dihedral angles on a protein backbone. The algorithm adopted is the one proposed by Zamuner et al, PLoSOne 10(3):e0118342.

The basic idea is to assign to each bond of the chain an orthonormal base, following the Denavit-Hartenberg convention. The constraints of moving a series of bonds while keeping the first and last of them fixed can then be
interpreted as a constraint on the base transformation matrix bringing the base attached to the first bond into the one attached to the last bond.

Such constraints correspond to 6 independent equations: 3 for the orientation and 3 for the translation- for the last bond. Given a set of $n > 6$ variables, the constraint equations identify a manifold of dimension $n - 6$ in $\mathbb{R}^6$. Given
a solution to the equations, a new one can then be found through a random displacement on the tangent space of the manifold followed by a minimization to project the new point on the tangent space back to the manifold.
Every such operation correspond to a concerted rotation of the selected portion of the chain. The tangent space can be identifying applying Dini's theorem once a free variable is chosen.

Here we consider only a variation of 7 torsion angles, giving us a one-dimensional manifold. Our approach can nonetheless be generalized to a different set of 7 variables without a significative effort, and to a set of more than seven variables with some investment.

Main differences with respect to Zamuner's approach:

1. In this version, we do not consider rescaling factors. These must be introduced when one wishes to consider a mix of angles and lengths as free variables for the move.
2. We compute the jacobian explicitly. However, being precomputed, it does not impact the speed of our code.

Organization of this notebook

The notebook is divided in three sections, corresponding to the three stages to create a C library for concerted rotations.

1. Short theoretical introduction. A very brief introduction to the method. For more details please see Zamuner et al, or our epje article**.**
2. Equations implementing the move. All constraints equations are written explicitly with respect to the chosen set of free variables, called t_1, ..., t_7.
3. Printing the equations. Generates all files needed by the python wrapper to put together the C library. Defines the substitution rule for sines and cosines.
4. Call the wrapper.

Notebook general parameters

```
SetDirectory[NotebookDirectory[]]
```

```
/home/tubiana/Science/Projects/ABM/concerted_rotations/Code/generic_move
```

```
Directory[]
```

```
/home/tubiana/Science/Projects/ABM/concerted_rotations/Code/generic_move
```

A quick introduction to the method

Following Zamuner et al, we implement the concerted rotations by following the DH convention. For every link (chain bond) involved in the move we construct an orthonormal basis $\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$ as follows:
$\mathbf{e}_{z,i} = \boldsymbol{b}_i / |\boldsymbol{b}_i|$, where $\boldsymbol{b}_i$ is the bond vector to which the i – th base is attached;
$\mathbf{e}_{x,i} = \mathbf{e}_{z,i-1} \times \mathbf{e}_{z,i}$;
$\mathbf{e}_{y,i} = \mathbf{e}_{z,i} \times \mathbf{e}_{x,i}$.
The origin of reference frame O_i is placed on the first atom of the corresponding bond if $\mathbf{e}_{z,i}$, $\mathbf{e}_{z,i-1}$ are coplanar, or along $\mathbf{e}_{z,i}$ at the minimum distance from $\mathbf{e}_{z,i-1}$ otherwise.

A vector a_i in the reference frame O_i can be expressed in the reference frame O_{i-1} as:
$\boldsymbol{a}_{i-1} = \boldsymbol{R}^i{}_{i-1}\, \boldsymbol{a}_i + \boldsymbol{S}^i{}_{i-1}$
where $\boldsymbol{R}^i{}_{i-1}$ is the transformation matrix expressing the i – th basis in terms of the $(i-1)$ – th basis and $\boldsymbol{S}^i{}_{i-1}$ is the displacement vector between their origins.

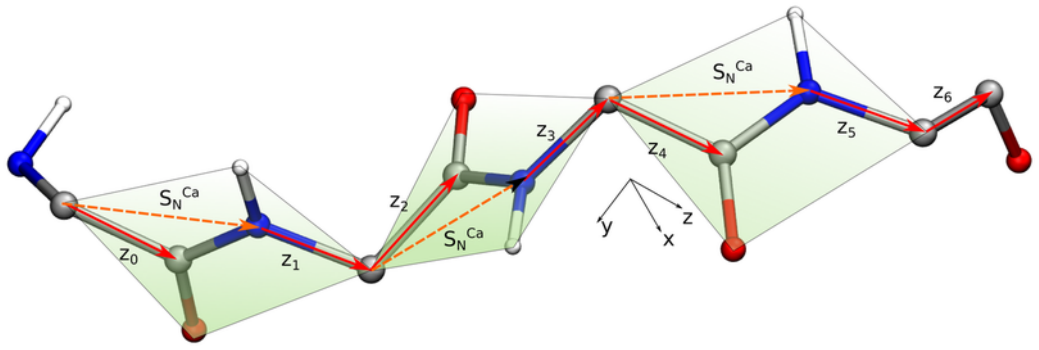
Introducing the vector $\boldsymbol{a}' = (\boldsymbol{a}, 1)^T$, we can rewrite the previous expression in a more compact form as:
$\boldsymbol{a}'_{i-1} = \boldsymbol{T}^i{}_{i-1}\, \boldsymbol{a}'_i$,
where

$$\boldsymbol{T}^i{}_{i-1} = \begin{pmatrix} \boldsymbol{R}^i{}_{i-1} & \boldsymbol{S}^i{}_{i-1} \\ 0 & 1 \end{pmatrix}$$

Our *Mathematica* notebook implements these matrices in terms of the 4 DH parameters:
* the ***link twist*** α_i, defined as the angle between $\mathbf{e}_{z,i}$ and $\mathbf{e}_{z,i-1}$, measured around $\mathbf{e}_{x,i}$;
* the ***joint angle*** θ_i, defined as the angle between $\mathbf{e}_{x,i}$ and $\mathbf{e}_{x,i-1}$, measured around $\mathbf{e}_{z,i-1}$;
* the ***link offset*** d_i, given by $d_i = \left|\boldsymbol{S}^i{}_{i-1} - \left(\boldsymbol{S}^i{}_{i-1} \cdot \mathbf{e}_{z,i-1}\right) \mathbf{e}_{z,i-1}\right|$
* the ***link length*** r_i, given by $r_i = \boldsymbol{S}^i{}_{i-1} \cdot \mathbf{e}_{z,i-1}$

Composing the T matrices we can impose a constraint for the move of a portion of polymer back-bone, by requesting that a vector $\boldsymbol{a}'$ is tranformed in the same way both by the matrices T before the move and by the ones modified by the move:

$$\boldsymbol{a}'_i = \boldsymbol{T}^{i+1}{}_i\, \boldsymbol{T}^{i+2}{}_{i+1} \ldots. \boldsymbol{T}^j{}_{j-1}\, \boldsymbol{a}'_j \longrightarrow \left(\boldsymbol{T}^{i+1}{}_i\, \boldsymbol{T}^{i+2}{}_{i+1} \ldots. \boldsymbol{T}^j{}_{j-1}\right)_{\text{old}} - \left(\boldsymbol{T}^{i+1}{}_i\, \boldsymbol{T}^{i+2}{}_{i+1} \ldots. \boldsymbol{T}^j{}_{j-1}\right)_{\text{new}} = 0$$



The figure above provides a graphical example for the concerted rotation of three consecutive residues on a polypeptide chain. This involves seven T matrices. The matrices, the constraint function and the derivatives necessary to define the tangent space to the constraint manifold are obtained in the rest of the *Mathematica* notebook.

Equations implementing the move

Transformation matrix

We consider a general transformation matrix.

In the following, α is the link twist, θ the joint angle, r=$\vec{S}\cdot\hat{z}$ is the z component of $\vec{S}$, and d=$\|\vec{S}-(\vec{S}\cdot\hat{z})\,\hat{z}\|$, with $\vec{S}$ the vector joinint the base i-1 with the base i .

```
T[θ_, α_, d_, r_] := {{Cos[θ], -Cos[α] Sin[θ], Sin[θ] Sin[α], d * Sin[θ]},
  {Sin[θ], Cos[θ] Cos[α], -Cos[θ] Sin[α], -d * Cos[θ]},
  {0., Sin[α], Cos[α], r},
  {0., 0., 0., 1.}}
```

```
T[θ, α, d, r] // MatrixForm
```

```
( Cos[θ]  -Cos[α] Sin[θ]   Sin[α] Sin[θ]   d Sin[θ]  )
( Sin[θ]   Cos[α] Cos[θ]  -Cos[θ] Sin[α]  -d Cos[θ]  )
(   0.         Sin[α]          Cos[α]          r      )
(   0.           0.              0.            1.     )
```

Tranformation matrix for seven dihedral angles.

The Final transformation matrix is obtained by composing the 7 elementary transformation matrices given above.

For example, going from one N to a C, 4 Cα apart, I move 7 dihedrals and the transformation Matrix is:

```
T7[t1_, t2_, t3_, t4_, t5_, t6_, t7_] :=
 T[t1, α1, d1, r1].T[t2, α2, d2, r2].T[t3, α3, d3, r3].
  T[t4, α4, d4, r4].T[t5, α5, d5, r5].T[t6, α6, d6, r6].T[t7, α7, d7, r7]
```

Please note that the user is free to choose a different set of seven variables here. For examples link twists or a mixture of link twists and joint angles. The number of matrices can be lower than 7 in that case.

To mix angles and lengths, it will be necessary to take into account rescaling factors in the following, a feature which is currently mixing.

Constraints equations.

The elements chosen for the constrain equations are the following: [[1, 3]], [[2, 3]], [[1,2]], [[1, 4]], [[2, 4]], [[3, 4]].
The first three fix the orientation of the vector z in the last ref. system according to the first ref. system. The last Three its spatial position.

```
T7eqs[t1_, t2_, t3_, t4_, t5_, t6_, t7_] :=
 {(T7[t1, t2, t3, t4, t5, t6, t7])[[1, 2]],
  (T7[t1, t2, t3, t4, t5, t6, t7])[[1, 3]],
  (T7[t1, t2, t3, t4, t5, t6, t7])[[2, 3]],
  (T7[t1, t2, t3, t4, t5, t6, t7])[[1, 4]],
  (T7[t1, t2, t3, t4, t5, t6, t7])[[2, 4]],
  (T7[t1, t2, t3, t4, t5, t6, t7])[[3, 4]]}
```

Derivative of T7 for different dihedral angles.

These are needed to apply Dini’s theorem.

```
DT7t1 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t1}];
DT7t2 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t2}];
DT7t3 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t3}];
DT7t4 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t4}];
DT7t5 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t5}];
DT7t6 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t6}];
DT7t7 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {t7}];
```

Jacobians

To apply Dini’s theorem, we have to choose one free variable, compute the jacobian of the equations with respect to the other 6 variables and invert it. Inversion will be done numerically. Nonetheless, it will be possible to print an expression for the determinant of the jacobian, allowing one to check if the inversion is possible.

```
jacT7t1 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t2, t3, t4, t5, t6, t7}}];
jacT7t2 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t3, t4, t5, t6, t7}}];
jacT7t3 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t4, t5, t6, t7}}];
jacT7t4 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t3, t5, t6, t7}}];
jacT7t5 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t3, t4, t6, t7}}];
jacT7t6 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t3, t4, t5, t7}}];
jacT7t7 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t3, t4, t5, t6}}];
```

```
JACT7 = D[T7eqs[t1, t2, t3, t4, t5, t6, t7], {{t1, t2, t3, t4, t5, t6, t7}}];
```

Printing equations in C format

All the equations and functions derived before must be plugged in a C code. We do so by printing them in C form. For the C code to be fast and reliable, we need to do two things:

1. substitute all sines and cosines with some variables c and s, which will be calculated beforehand and passed to the function.

2. Put the equations in the simplest possible polynomial form. This is achieved by using the Horner-Form routine.

Point 1. above is achieved by using the following substitution rule.

```
substitRule = {Cos[t1] → c1, Cos[t2] → c2, Cos[t3] → c3,
   Cos[t4] → c4, Cos[t5] → c5, Cos[t6] → c6, Cos[t7] → c7,
   Sin[t1] → s1, Sin[t2] → s2, Sin[t3] → s3, Sin[t4] → s4,
   Sin[t5] → s5, Sin[t6] → s6, Sin[t7] → s7,
   Cos[α1] → ca1, Cos[α2] → ca2, Cos[α3] → ca3, Cos[α4] → ca4,
   Cos[α5] → ca5, Cos[α6] → ca6, Cos[α7] → ca7,
   Sin[α1] → sa1, Sin[α2] → sa2, Sin[α3] → sa3, Sin[α4] → sa4,
   Sin[α5] → sa5, Sin[α6] → sa6, Sin[α7] → sa7
  };
```

All equations are printed one line per file. A python wrapper then puts together the C headers and source files.

Export T7

```
Export["T7_12.c", CForm[
   HornerForm[T7[t1, t2, t3, t4, t5, t6, t7][[1, 2]] /. substitRule]], "Text"];
Export["T7_13.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[1, 3]] /. substitRule]], "Text"];
Export["T7_23.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[2, 3]] /. substitRule]], "Text"];
Export["T7_14.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[1, 4]] /. substitRule]], "Text"];
Export["T7_24.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[2, 4]] /. substitRule]], "Text"];
Export["T7_34.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[3, 4]] /. substitRule]], "Text"];
Export["T7_33.c", CForm[HornerForm[
    T7[t1, t2, t3, t4, t5, t6, t7][[3, 3]] /. substitRule]], "Text"];
```

Export DT7

```
Do[Module[{bname = "DT7_t1"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t1[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t2"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t2[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t3"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t3[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t4"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t4[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t5"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t5[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t6"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t6[[j]]] /. substitRule], "Text"]], {j, 6}]
```

```
Do[Module[{bname = "DT7_t7"},
  fname = bname <> "_" <> ToString[j] <> ".c";
Export[fname, CForm[HornerForm[DT7t7[[j]]] /. substitRule], "Text"]], {j, 6}]
```

Export the Jacobians

```
Do[
 Do[
   Module[{bname = "jac_t1"},
    el = CForm[HornerForm[jacT7t1[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]
```

```
Do[
 Do[
   Module[{bname = "jac_t2"},
    el = CForm[HornerForm[jacT7t2[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]
```

```
Do[
 Do[
   Module[{bname = "jac_t3"},
    el = CForm[HornerForm[jacT7t3[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]
```

```
Do[
 Do[
   Module[{bname = "jac_t4"},
    el = CForm[HornerForm[jacT7t4[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]
```

```
j
```

```
j
```

```
Do[
 Do[
   Module[{bname = "jac_t5"},
    el = CForm[HornerForm[jacT7t5[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]

Do[
 Do[
   Module[{bname = "jac_t6"},
    el = CForm[HornerForm[jacT7t6[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]

Do[
 Do[
   Module[{bname = "jac_t7"},
    el = CForm[HornerForm[jacT7t7[[i, j]]] /. substitRule];
    fname = bname <> "_" <> ToString[i] <> ToString[j] <> ".c";
    Export[fname, el, "Text"]
   ], {j, 6}],
 {i, 6}]
```

Call wrapper

```
Run["python ./Precompiler.py"]
(*Run["rm *_t*.c"]*)
(*Run["./create_C_files_LUdecomp_inner_check.sh"]*)
```