C++ implementation of the extended Lennard-Jones Potential energy function

Yu Zhai*[1]

[1] *Laboratory of Theoretical and Computational Chemistry, Institute of Theoretical Chemistry, Jilin University, 2519 Jiefang Road, Changchun 130023, People's Republic of China*

28th March 2019

Following Hajigeorgiou's work[1], in this document, I will describe how I implemented the function in C++.

1 The extended Lennard-Jones function

The extended Lennard-Jones (ELJ) function is defined as

$$V_{\mathrm{ELJ}} = \mathscr{D}_{\mathrm{e}} \left[1 - C(r) \left(\frac{r_{\mathrm{e}}}{r} \right)^{n(r)} \right]^2 , \tag{1}$$

where C is defined as

$$C(r) = \left(1 - \frac{f_{\mathrm{LR}}(r)}{f_{\mathrm{LR}}(r_{\mathrm{e}})} \right) \left(\frac{1}{2\mathscr{D}_{\mathrm{e}}} \right) \sum_{i=0}^{M} \frac{C_{n_i}}{r_{\mathrm{e}}^{n_i}} \left(\frac{r_{\mathrm{e}}}{r} \right)^{\Delta n} + \frac{f_{\mathrm{LR}}(r)}{f_{\mathrm{LR}}(r_{\mathrm{e}})}, \tag{2}$$

where $\Delta n = n_i - n_0$ and

$$f_{\mathrm{LR}}(r) = \left[1 + \exp\left(\delta_{\mathrm{LR}} \left(r - R_{\mathrm{LR}} \right) \right) \right]^{-1} . \tag{3}$$

The n function is defined as

$$n(r) = \begin{cases} \left(\sum_{i=0}^{N} \rho_i \zeta^i \right) f_{\mathrm{n}}(r) + n_0 (1 - f_{\mathrm{n}}(r)) & r > R_{\mathrm{c}}, \\ a e^{br} & r \le R_{\mathrm{c}}, \end{cases} \tag{4}$$

and

$$f_{\mathrm{n}}(r) = \left[1 + \exp\left(\delta_{\mathrm{n}} \left(r - R_{\mathrm{n}} \right) \right) \right]^{-1} . \tag{5}$$

ζ in Eq.(4) can be of two types:

*Email: `yuzhai@mail.huiligroup.org`

- Dunham type

$$\xi = (r - r_\mathrm{e})/r_{r_\mathrm{e}}, \tag{6}$$

- Ogilvie-Tipping type

$$z = 2(r - r_\mathrm{e})/(r + r_\mathrm{e}). \tag{7}$$

2 Implementation details

The ELJ function is implemented as a class in C++ `ELJ`. The evaluation of potential energy is regarded as reload of operator `()`.

I also define the following member variables to store the values (Table 1).

Acknowledgement

This work is sponsored by nobody...

References

[1] Photos G. Hajigeorgiou. The extended lennard-jones potential energy function: A simpler model for direct-potential-fit analysis. *Journal of Molecular Spectroscopy*, 330:4–13, dec 2016. doi: 10.1016/j.jms.2016.06.014.

Table 1: Members in class `ELJ`. For all C++ variables in the left column, namespace code `ELJ::` is omitted.

C++ variable	Formula	Notes
0. Interatomic distance		
`double r`	r	
1. Parameters		
`const double D_e`	$\mathscr{D}_\mathrm{e}$	
`const double r_e`	r_e	
`const int n[]`	n_i	
`const double C_n[]`	C_{n_i}	
`const double delta_LR`	δ_LR	
`const double R_LR`	R_LR	
`const double R_c`	R_c	
`const double rho[]`	ρ_i	
`const double delta_n`	δ_n	
`const double R_n`	R_n	
`const double a`	a	
`const double b`	b	
2. Terms		
`const double fLRre`	$f_\mathrm{LR}(r_\mathrm{e})$	
`double ffLR`	$\frac{f_\mathrm{LR}(r)}{f_\mathrm{LR}(r_\mathrm{e})}$	
`double rr`	$\frac{r_\mathrm{e}}{r}$	
`double Cnre[]`	$\frac{C_{n_i}}{r_\mathrm{e}^{n_i}}$	
`const double over2De`	$\frac{1}{2\mathscr{D}_\mathrm{e}}$	
3. Functions		
`double C()const`	$C(r)$	Eq. (2)
`double n()const`	$n(r)$	Eq. (4)
`double f(double delta, double r)const`	$f_\mathrm{LR}(r)$ and $f_\mathrm{LR}(r)$	Eqs. (3)(5)
`double xi()const`	ξ	Eq. (6)
`double z()const`	z	Eq. (7)