rexpon_types.h

Parameters that are new in reax_types.h

```
typedef struct{
    // bond energy parameters
    double bom;
    // Van der Waals interaction parameters*
    double reqm;
    double beta;
    double repa0, repr0, repn, repscal, reps, reps1;
}three_body_header;
typedef struct{
  double r0_hb, p_hb1, p_hb2, p_hb3; // these are also in Reax
  double p_hb4 ~~~ p_hb27; // new in RexPoN;
} hbond_parameters;
```

rexpon_ffield.cpp

This module contains a single function (~800 lines)

`char Read_Force_Field( FILE *fp, rexpon_interaction *rexpon, control_params *control )`

Parameters

fp: The RexPoN parameter file, e.g. *ffield.RexPoN*

rexpon: The data structure holding all rexpon parameters, including num_atom_types (total number of atom types), global_parameters gp, single_body_parameters *sbp; two_body_parameters **tbp; three-body-parameters ***thbp; hbond_parameters ***hbp; four_body_header ****fbp; This function will allocate spaces for sbp, tbp, thbp, hbp, and fbp using function *scalloc()* (defined in rexpon_tool_box.h/.cpp) but have not found how these memory are freed (should be freed via *sfree()*, which is also defined in rexpon_tool_box.h/.cpp)

control: This function will read the lgflag from *control*, and write bo_cut, nonb_low, nonb_cut to *control*

Return Value Always returns *SUCESS* (=1, as defined in *rexpon_defs.h*)

Local Variables

`char *s;` A line of text read from file

`char **tmp;` Words split from a line

`char ****tor_flag;` A 4-d array of bool flags. tor_flag[i][j][k][l] (as well as tor_flag[l][k][j][i]) records whether a torsion between i-j-k-l is described in the parameters file. However, this flag is set but NEVER used. On the other hand, rexpon->fbp[i][j][k][l] is directly written once a four-body parameter is read.

`int c, i, j, k, l, m, n, o, p, cnt;`
c is the number of words when a line is split via *Tokenize()* (defined in *rexpon_tool_box.cpp*) i, j, k, l are loop variables. m,n,o,p,cnt are temp ints

`int lgflag = control->lgflag;` copied from control->lgflag

`int errorflag = 1;` output error message when vdw type is inconsistent

`double val;` temp variable for floats

`MPI_Comm comm; int me;` MPI comm, *me* is MPI rank of current process.

The format of RexPoN forcefield parameters file

Below is an example for RexPoN forcefield description file

```
Reactive MD-force field. ! Comment Line
39  ! Number of general parameters.
50.0000  !p(boc1)
9.5469   ! p(boc2)
26.5405  !p(coa2)
3.0000   ! 1.5105 !p(trip4) **
6.5000   ! 6.6630 !p(trip3) **
0.0000   ! 70.0000 !kc2     **
1.0588   ! 1.0588 !p(ovun6) **
9.0000   ! 4.6000 !p(trip2) **
12.1176  ! p(ovun7)
13.3056  ! p(ovun8)
0.0000   ! -70.1292 !p(trip1) **
0.0000   ! Lower Taper-radius (swa)
12.0000  ! 10.0000 !Upper Taper-radius (swb) **
2.8793   ! 0.0000 !not used **
33.8667  !p(val7)
6.0891   !p(lp1)
1.0563   !p(val9)
2.0384   !p(val10)
6.1431   !not used
6.9290   !p(pen2)
0.3989   !p(pen2)
3.9954   !p(pen4)
1.0000   ! 0.0000 !not used  **
5.7796   !p(tor2)
10.0000  !p(tor2)
1.9487   !p(tor2)
-1.2327  !0.0000 !not used   **
2.1645   !p(cot2)
1.5591   !p(cot2)
10.0100  !0.1000 !Cutoff for bond order*100 (cutoff)
2.1365   !p(coa4)
1.5001   !0.6991 !p(ovun4)  **
3.2593   !50.0000 !p(ovun3)
1.8512   !p(val8)
0.5000   !0.0000 !not used  **
```

```
1.0000   !0.0000 ! factor in tbp[i][j].lgre   **
5.0000   !0.0000 !not used  **
0.0000   !0.0000 !not used  **
2.6962   !p(coa3)
2      ! Nr of atoms; atomID;ro(sigma); Val;atom
mass;Rvdw;Dij;gamma;ro(pi);Val(e)
line 1 !alfa;gamma(w);Val(angle);p(ovun5);n.u.;chiEEM;etaEEM;n.u.
line 2 !ro(pipi);p(lp2);Heat increment;p(boc4);p(boc3);p(boc5),n.u.;n.u.
line 3 !p(ovun2);p(val3);n.u.;Val(boc);p(val5);n.u.;n.u.;n.u.
Hw 0.8930 0.9000 1.0080 1.3550 0.0000 0.8203 -0.1000 1.0000
8.2230 0.0000 0.9000 0.0000 121.1250 3.7248 9.6093 2.0000
-0.1000 0.0000 55.1878 6.1413 17.2017 17.0003 1.0698 0.0000
-19.4571 4.2733 1.0338 0.9000 2.8793 1.0000 0.0000 1.0000
239.4803 1.6346 2.5848
Ow 1.2450 1.8000 15.9990 2.3890 0.0000 1.0898 -1.0548 2.0000
9.7300 0.0000 1.8000 0.0000 116.0768 8.5000 8.3122 1.0000
-0.9049 0.0000 68.0152 6.1413 17.2017 17.0003 0.9745 0.0000
-3.5500 3.3562 1.0493 1.8000 2.9225 1.0000 0.0000 1.0000
634.7066 1.4969 3.4012
3  ! Nr of bonds;
at1;at2;De(sigma);De(pi);De(pipi);p(be1);p(bo5);13corr;n.u.;p(bo6),p(ovun1
)
line 1 ! p(be2);p(bo3);p(bo4);n.u.;p(bo1);p(bo2)
1 1 109.4853 0.0000 0.0000 1.0000 0.0000 1.0000 6.0000 0.0000
4.0000 1.0000 0.0000 1.0000 2.3026 1.3532 0.0000 0.0000
1 2 113.250 0.0000 0.0000 0.1073 0.0000 1.0000 6.0000 0.0000
33.335203 1.0000 0.0000 1.00 2.3026 1.4428 0.0000 0.0000
2 2 0.0000 100.0000 0.0000 1.0785 0.0000 1.0000 6.0000 0.0000
8.1798 -0.1055 9.0000 0.6190 2.1972 1.7063 0.0000 0.0000
3  ! Aij, -alpha, beta, nu, nu, nu, gamma, nij, eta, delta, nu, Rvdw/2, C6
1 1 0.0212 2.4002 10.5460 0 0 0 -0.8113 0.0014 -0.2607 -0.5640 0 1.6346
239.4803
1 2 0.0906 3.1455 10.0823 0 0 0 -0.0007 -1.000 0.00000 0.00000 0 1.56425
389.8714
2 2 0.0272 3.6468 14.5278 0 0 0 -0.3235 0.0305 0.03310 -0.5688 0 1.4969
634.7066
1   ! Nr of angles.
at1;at2;at3;Thetao,o;p(val1);p(val2);p(coa1);p(val7);p(pen1);p(val4)
1 2 1 98.8511 31.6132 2.2720 0.0000 2.8635 0.0000 3.8453
0   ! Nr of torsions
1    ! Nr of hydrogen bonds. at1;at2;at3;r(hb);p(hb1);p(hb2);p(hb3)
2 1 2 2.1200 0.765 0.0 0.0 10.2511 2.0214 0.0 15.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-22.5 12.7434 1.1345 0.7861 3.5071 1.1693 3.6922 -0.8921
```

Detailed explanation of each parameter

Global Parameters

```
Reactive MD-force field. ! Comment Line
39 ! Number of general parameters.
```

This number is stored in rexpon->gp.n_global. An array *rexpon->gp.l* is then allocated to store the following

(39) parameters. In Addition, the 30th parameter (10.01 above) is written to control->bo_cut; the 12th and 13th parameters (0 and 12 above) are written to *control->nonb_low* and *control->nonb_cut*.

```
50.0000  !p(boc1)
......
2.6962   !p(coa3)
```

Single Atom Parameters

Global parameters are followed by single atom parameters.

```
2      ! Nr of atoms; atomID;ro(sigma); Val;atom
mass;Rvdw;Dij;gamma;ro(pi);Val(e)
....3 lines of comments ...
Hw 0.8930 0.9000 1.0080 1.3550 0.0000 0.8203 -0.1000 1.0000
....4 more lines for type Hw
....5 lines for the next atom type ...
```

The number *2* here in the 1st line is the number of atom types, which is written to *rexpon->num_atom_types*. This number is also used in memory allocations of rexpon->sbp, tbp, thbp, hbp, and fbp, which are 1-, 2-, 3-, 3-, and 4- dimensional arrays, respectively. The length of all dimensions are exactly *rexpon->num_atom_types* read from the 1st line above.

Following three lines of comments, there are 5 lines for each atom type (4 lines for the original Reax FF). Taking the Hw type as an example, these 5 lines are:
`Hw 0.8930 0.9000 1.0080 1.3550 0.0000 0.8203 -0.1000 1.0000`
The atom type `Hw` will be converted to upper case before written to *rexpon->sbp[i].name*, where *i* is the index for atom types. The following 8 numbers are stored in members of *rexpon->sbp[i]*, specifically, to *r_s, valency, mass, r_vdw, epsilon, gamma, r_pi, valency_e*, respectively. Another member, *nlp_opt* is calculated from *0.5*(valency_e - valency)*

`8.2230 0.0000 0.9000 0.0000 121.1250 3.7248 9.6093 2.0000`
alpha, gamma_w, valency_boc, p_ovun5, not used, chi, eta (= 2 the given value), p_hbond*

`-0.1000 0.0000 55.1878 6.1413 17.2017 17.0003 1.0698 0.0000`
r_pi_pi, p_lp2, not used, b_o_131, b_o_132, b_o_133, not used, not used

`-19.4571 4.2733 1.0338 0.9000 2.8793 1.0000 0.0000 1.0000`
p_ovun2, p_val3, not used valency_val, p_val5, rcore2, ecore2, acore2

`239.4803 1.6346 2.5848`
p_ovun2, p_val3, not used valency_val, p_val5, rcore2, ecore2, acore2 Only when `lgflag == 1`, this line will be read (reminder: lgflag is read from *control->lgflag*)
But interestingly, only the first two numbers are read and saved to *lgcij* and *lgre*, the last number is ignored.

The code will then judge whether the atoms are with inner wall (rcore2 > 0.01 and acore2 > 0.01) or not (currently both 1.0). As well as whether it is “Shielding vdWaals” by whether gamma_w > 0.5 (currently 0.0).

There are four different situations:
With inner wall and shielding: rexpon->gp.vdw_type = 3; (11 in binary)
With inner wall but no shielding: rexpon->gp.vdw_type = 2; (10 in binary)
Without inner wall but with shielding: rexpon->gp.vdw_type = 1; (01 in binary)
Without inner and no shielding: rejects the input and prints error messages.
But these if-statements are from Reax and may be ignored now.

Two-body parameters

rexpon->tbp[j][k] are read from ffield.RexPoN, one for each bond, for example, there are 8 lines describing 3 bond types (H-H, H-O, O-O) in the lines below:

```
3  ! Nr of bonds;
at1;at2;De(sigma);De(pi);De(pipi);p(be1);p(bo5);13corr;n.u.;p(bo6),p(ovun1
)
line 1 ! p(be2);p(bo3);p(bo4);n.u.;p(bo1);p(bo2)
1 1 109.4853 0.0000 0.0000 1.0000 0.0000 1.0000 6.0000 0.0000
4.0000 1.0000 0.0000 1.0000 2.3026 1.3532 0.0000 0.0000
1 2 113.250 0.0000 0.0000 0.1073 0.0000 1.0000 6.0000 0.0000
33.335203 1.0000 0.0000 1.00 2.3026 1.4428 0.0000 0.0000
2 2 0.0000 100.0000 0.0000 1.0785 0.0000 1.0000 6.0000 0.0000
8.1798 -0.1055 9.0000 0.6190 2.1972 1.7063 0.0000 0.0000
```

1st Line: number of total bonds

2nd Line: Comment

`1 1 109.4853 0.0000 0.0000 1.0000 0.0000 1.0000 6.0000 0.0000`
3rd Line: Parameter for bond between atom type 1 and atom type 1. The 8 numbers are read to members of tbp[k][j], which are *De_s, De_p, De_pp, p_be1, p_bo5, v13cor, p_bo6, and p_ovun1*. Their meanings are same as in Reax.

`4.0000 1.0000 0.0000 1.0000 2.3026 1.3532 0.0000 0.0000`
4th Line: Continuation of line 3 for bond between type 1 and type 2. 8 numbers are *p_be2, p_bo3, p_bo4, bom, p_bo1, p_bo2, ovc, reqm*. The *bom, ovc, reqm* are new in RexPoN.

5-6 Lines: parameters are bond between type 1 and 2. 7-8 Lines are for bond between type 2 and 2. etc.

After reading the bond parameters. *tbp[i][j].r_s, r_pi, r_pi_pi* are set to arithmetic means of *r_s, r_pi, r_pi_pi* of involved atom. *p_boc3, p_boc4, p_boc5, D, alpha* are set to geometric means of *b_o_132, b_o_131, b_o_133, epsilon, alpha* of involved atom, respectively. Inner core parameters: *rcore* are from: rcore_ij=sqrt(rcore2_i rcore2_j). *ecore and acore* are calculated in a similar fashion from ecore2_i and acore2_i. *lgcij* are arithmetic means of atoms. *lgre* are arithmetic means of each atom times 2.0 time the 36th global parameter (1.0 above).

In addition, RexPoN specific parameters, *alpha, r_vdw, gamma_w* are from geometric means of respective parameters of each atom. *gamma* are from g_ij=(g_i g_j)^(-1.5).

The next section, so-called 'off-diagonal terms', are also read to tbp parameters: (They are different from Reax parameters, they should be Pauli Repulsion parameters for non-bonded interactions in RexPoN. There are still problems in this section...)

```
3  ! Aij, -alpha, beta, nu, nu, nu, gamma, nij, eta, delta, nu, Rvdw/2, C6
1 1 0.0212 2.4002 10.5460 0 0 0 -0.8113 0.0014 -0.2607 -0.5640 0 1.6346
239.4803
1 2 0.0906 3.1455 10.0823 0 0 0 -0.0007 -1.000 0.00000 0.00000 0 1.56425
389.8714
2 2 0.0272 3.6468 14.5278 0 0 0 -0.3235 0.0305 0.03310 -0.5688 0 1.4969
634.7066
```

Parameters in this line has different meaning for different vdw types
There are 13 values following the atom types represent (all are members of tbp[j][k] = tbp[k][j]):
*D, r_vdw/beta, alpha, r_s, r_p, r_pp, repa0, repr0, repn, repscal, reps, lgre (=2*given value), and lgcij*
The *r_vdw* and *beta* are shared one position. The last 7 parameters (starting from repa0) are new in RexPoN.

- In the current implementation, if the program detects the species are water (atom names Hw and Ow), then it will use the VDW method described in the original RexPoN paper [J Chem Phys 149, 174502, 2018] and the 13 numbers are: `Aij, -alpha, beta, nu., nu., nu., gamma, nij, eta, delta, nu., RvdWij/2 ,C6ij`
- If the program detects non-water molecules, it will use the Universal Non-Bond (UNB) method [J Chem Phys 151, 154111, 2019] and the 13 numbers are: `De, Re, L, nu., nu., nu., beta, s0 ~ s5`, for example:

```
3  ! De Re L nu nu nu beta s0 s1 s2 s3 s4 s5 s6
1 1 0.0528   3.2541  0.5241  0 0 0 1.0035 1.0000 1.0201 0.0168 0.0033
0.0037 0.0011
1 2 0.088935032 3.3384078675  0.4774212919 0 0 0 1.0035 1.0000 1.0201
0.0168 0.0033 0.0037 0.0011
2 2 0.1498   3.4249  0.4349  0 0 0 1.0035 1.0000 1.0201 0.0168 0.0033
0.0037 0.0011
```

Three-body parameters

Firstly, *thbp[i][j][k].cnt* are set to 0 for all i,j,k. The parameters are (identical to Reax):
`1 2 1 98.8511 31.6132 2.2720 0.0000 2.8635 0.0000 3.8453`
j, k, m, theta_00, p_val1, p_val2, p_coa1, p_val7, p_pen1, p_val4
When reading each line, the atom types are read at first. Then the cnt of thbp[j][k][m] and thbp[m][k][j] are incremented by 1. All subsequent operations are performed symmetrically on thbp[j][k][m] and thbp[m][k][j]. The following numbers are stored to members of thbp[j][k][m].prm[cnt]. It seems like multiple sets of parameters can be given to a combination of atom types.

Four-body parameters

rexpon->fbp[i][j][k][m]

At first, fbp[i][j][k][m].cnt are all set to 0. tor_flag[i][j][k][l] are also set to 0. They are same as in Reax. Four-body parameters are not involved in the current case. if present, there should be in a format like:
`p k m o V1 V2 V3 p_tor1 p_cot1`

Hydrogen-bond parameters

Hydrogen-bond belong to three-body interactions, but the parameters appear AFTER Four-body paras.
H-Bond paras are NOT symmetric, i.e *rexpon->hbp[i][j][k]* is **not equal** to *rexpon->hbp[k][j][i]*
There is also not a *cnt* flag as in *fbp* to signify the existence of H-Bond paras. In the beginning, *r0_hb* of all *hbp* are set to -1.0. Firstly, hbp[i][j][k].r0_hb for all i,j,k are set to -1.0.

```
1    ! Nr of hydrogen bonds. at1;at2;at3;r(hb);p(hb1);p(hb2);p(hb3)  ????
2 1 2 2.1200 0.765 0.0 0.0 10.2511 2.0214 0.0 15.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-22.5 12.7434 1.1345 0.7861 3.5071 1.1693 3.6922 -0.8921
```

1st Line: 8 numbers after *2 1 2* are: *r0_hb, p_hb1, p_hb2, p_hb3, p_hb4, p_hb5, p_hb6, p_hb7*
2nd Line: *p_hb8 ~ p_hb15*. Why are they all zero?
3rd Line: *p_hb16 ~ p_hb23*.
Note: *p_hb4 ~ p_hb23* are new in RexPoN, and *hbp* actually has *p_hb24~p_hb27* but no code is responsible for reading them (are they calculated somewhere later?)