PFMDS

Создано системой Doxygen 1.8.14

Оглавление

Глава 1

Список задач

Подпрограмма md_neighbours::find_neighbour_distances (nl, atoms, group1, group2, box)

Сделать sqrt(dr2) быстрее.

Модуль md_read_write

Добавить чтение файла настроек.

Глава 2

Указатель модулей

2.1 Указатель модулей

Аннотированный списокмодулей:

Глава 3

Типы данных

3.1 Типы данных

Аннотированный список типов данных:

Глава 4

Список файлов

4.1 Файлы

Полный список файлов.

Глава 5

Группы

5.1 Модуль cut_off_function

Функции/подпрограммы

- real function f_cut (r, R1, R2)
- real function df_cut (r, R1, R2)

5.1.1 Функции/подпрограммы

5.1.1.1 df_cut()

```
real function cut_off_function::df_cut (
                    real r,
                    real R1,
                    real R2 )
```

См. определение в файле cut_off_function.f90 строка 19

```
19    real df_cut,r,R1,R2,pi
20    pi=3.14159265358979
21    if(r<r1) then
22        df_cut = 0.
23    elseif(r<r2) then
24        df_cut = -sin( pi*(r-r1)/(r2-r1) )*pi/(r2-r1)/r/2
25    else
26        df_cut = 0.
27    endif
```

5.1.1.2 f_cut()

```
real function cut_off_function::f_cut (
                real r,
                real R1,
                real R2 )
```

См. определение в файле cut_off_function.f90 строка 7

```
7     real f_cut,r,R1,R2,pi
8     pi=3.14159265358979
9     if(r<r1) then
10        f_cut = 1.
11    elseif(r<r2) then
12        f_cut = (1.+cos( pi*(r-r1)/(r2-r1) ))/2
13    else
14        f_cut = 0.
15    endif
```

5.2 Модуль cut_off_poly

Функции/подпрограммы

- real function f_cut (r, R1, R2)
- real function df_cut (r, R1, R2)
- pure subroutine f_dfr_cut (f, dfr, r, R1, R2)

5.2.1 Функции/подпрограммы

5.2.1.1 df_cut()

```
real function cut_off_poly::df_cut (
                real r,
                real R1,
                real R2 )
```

См. определение в файле cut_off_poly.f90 строка 20

```
20    real df_cut
21    real :: r,R1,R2
22
23    if(r>r1 .and. r<r2) then
24        df_cut = (-30.*(r-r1)**2*(r2-r1)**2+60.*(r-r1)**3*(r2-r1)-30.*(r-r1)**4)/(r2-r1)**5/r
25    else
26        df_cut = 0.
27    endif
```

5.2.1.2 f_cut()

```
real function cut_off_poly::f_cut (
                real r,
                real R1,
                real R2 )
```

См. определение в файле cut_off_poly.f90 строка 7

```
7     real f_cut
8     real :: r,R1,R2
9
10     if(r>r1 .and. r<r2) then
11        f_cut = 1.+(-10.*(r-r1)**3*(r2-r1)**2+15.*(r-r1)**4*(r2-r1)-6.*(r-r1)**5)/(r2-r1)**5
12     elseif(r<r1) then
13        f_cut = 1.
14     else
15        f_cut = 0.
16     endif
```

5.2.1.3 f_dfr_cut()

```
pure subroutine cut_off_poly::f_dfr_cut (
                real, intent(out) f,
                real, intent(out) dfr,
                real, intent(in) r,
                real, intent(in) R1,
                real, intent(in) R2 )
```

См. определение в файле cut_off_poly.f90 строка 31

```
31     real, intent(in) :: r,R1,R2
32     real, intent(out) :: f,dfr
33     real :: tempr,x,x2
34
35     tempr = r
36     tempr = max(tempr,r1)
37     tempr = min(tempr,r2)
38     x = (tempr-r1)/(r2-r1)
39     x2 = x*x
40     f = 1.+x2*x*(-10.+15.*x-6.*x2)
41     dfr = tempr*x2*(-30.+60.*x-30.*x2)
42
```

5.3 Модуль fit_gr_moire

Функции/подпрограммы

- subroutine calc_error (error, from_init_xyz, params)
- subroutine set_fitting_parameters (fitting_parameters_file_name, init_min_params, init_↩ max_params)

Переменные

- integer sim_num
- integer out_period
- integer num_of_omp_treads
- integer out_id
- integer final_out_id
- integer oid
- integer, dimension(2) ar_c_num
- character(len=256) interaction_name
- character(len=256), dimension(2) ar_settings_filename
- character(len=256) output_prefix
- character(len=256) input_path
- character(len=256) out_path
- character(len=256), dimension(2) ar_final_file
- character(len=256) param_file
- character(len=256), dimension(2) ar_start_xyz_file
- character(len=256), dimension(2) ar_xyz_file
- real z
- real, dimension(2) ar_zero_energy_level
- real be0
- real, dimension(2) ar_grd0
- real, dimension(2) rcut
- logical simplified
- character(len=80) line

5.3.1 Функции/подпрограммы

5.3.1.1 calc_error()

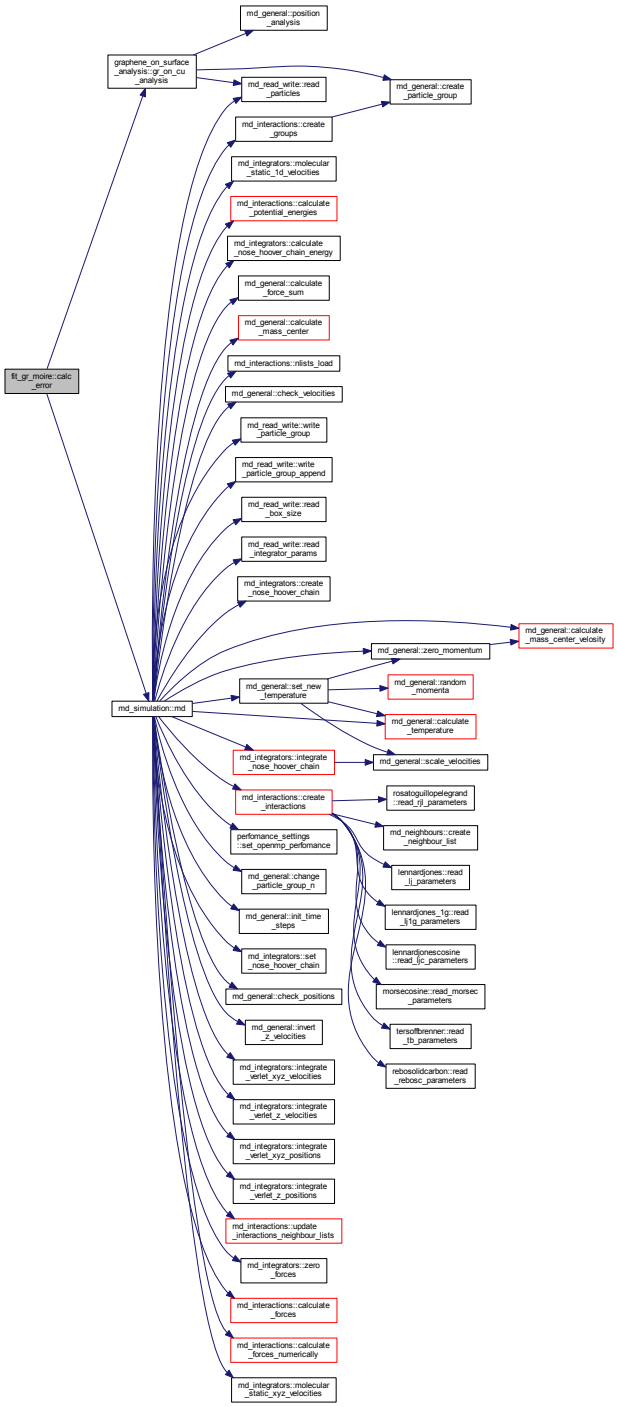
```
subroutine fit_gr_moire::calc_error (
                real error,
                logical from_init_xyz,
                real, dimension(4) params )
```

См. определение в файле fit_gr_moire.f90 строка 17

```
17 character(len=256)      :: settings_filename,start_xyz_file,xyz_file,final_file,str,tempstring,op,prevop
18 real                    :: params(4),error,arr1(3),arr2(3),be,bd,grd
19 logical                 :: from_init_xyz
20 integer                 :: i,rnm_err
21
22     error = 0
23     sim_num = sim_num+1
24     write(out_id,*)
25     write(out_id,*) trim(input_path)//trim(param_file)
26
27     if(interaction_name=='ljc') then
28         write(out_id,*) sim_num,params(3),params(1),params(2)
29         write(oid,'(i6,3f21.6)',advance='no') sim_num,params(3),params(1),params(2)
30         open(1234,file=trim(input_path)//trim(param_file))
31         write(1234,*) params(3),params(1),params(2)
32         write(1234,*) rcut(1),rcut(2)
33         write(1234,*) simplified
34         close(1234)
35     endif
36
37     if(interaction_name=='morsec') then
```

```
38        write(out_id,*) sim_num,params(3),params(1),params(4),params(2)
39        write(oid,'(i6,4f21.6)',advance='no') sim_num,params(3),params(1),params(4),params(2)
40        open(1234,file=trim(input_path)//trim(param_file))
41        write(1234,*) params(3),params(1),params(4),params(2)
42        write(1234,*) rcut(1),rcut(2)
43        write(1234,*) simplified
44        close(1234)
45    endif
46
47    do i=1,2
48
49        settings_filename = trim(ar_settings_filename(i))!; write(out_id,*) settings_filename
50        start_xyz_file = trim(ar_start_xyz_file(i))!;       write(out_id,*) start_xyz_file
51        xyz_file = trim(ar_xyz_file(i))!;                   write(out_id,*) xyz_file
52        final_file = trim(ar_final_file(i))!;               write(out_id,*) final_file
53
54        if(from_init_xyz) then
55            rnm_err = rename(trim(input_path)//trim(start_xyz_file),trim(input_path)//trim(xyz_file))
56        else
57            write(str,'(i6.6)') sim_num-1
58            write(prevop,'(A)') trim(out_path)//trim(output_prefix)//trim(str)//'_'
59            rnm_err = rename(trim(prevop)//'final_'//trim(xyz_file),trim(input_path)//trim(xyz_file))
60        endif
61
62        write(str,'(i6.6)') sim_num
63        write(op,'(A)') trim(out_path)//trim(output_prefix)//trim(str)//'_'
64        open(final_out_id,file=trim(op)//trim(final_file))
65        write(out_id,'(A)') line
66        call md(out_id,final_out_id,input_path,settings_filename,op,out_period,num_of_omp_treads)
67        write(out_id,'(A)') line
68        write(tempstring,'(A)') trim(op)//'final_'//trim(xyz_file)
69        call gr_on_cu_analysis(arr1,arr2,tempstring,z)
70        write(out_id,*) 'gr_on_cu_analysis:',arr1,arr2
71        write(final_out_id,'(3f16.6)') arr1-arr2(1)
72        close(final_out_id)
73
74        open(final_out_id,file=trim(op)//trim(final_file))
75        read(final_out_id,'(A61,f20.9,A)') tempstring,be,tempstring
76        close(final_out_id)
77
78        if(from_init_xyz) then
79            rnm_err = rename(trim(input_path)//trim(xyz_file),trim(input_path)//trim(start_xyz_file))
80        else
81            rnm_err = rename(trim(input_path)//trim(xyz_file),trim(prevop)//'final_'//trim(xyz_file))
82        endif
83
84        bd = arr1(1)-arr2(1)
85        grd = arr1(3)-arr1(2)
86        be = (be-ar_zero_energy_level(i))/ar_c_num(i)
87
88        write(oid,'(3f21.6)',advance='no') be,bd,grd
89
90        if(i==1) then
91            error = (be/be0-1.)**2+(grd/ar_grd0(i)-1.)**2
92            write(oid,'(2f21.6)',advance='no') (be/be0-1.)**2,(grd/ar_grd0(i)-1.)**2
93        endif
94        if(i==2) then
95            error = error+(grd/ar_grd0(i)-1.)**2
96            write(oid,'(f21.6)',advance='no') (grd/ar_grd0(i)-1.)**2
97        endif
98
99    enddo
100
101     write(oid,'(3f21.6)') error
102     rnm_err = rename(trim(input_path)//trim(param_file),trim(input_path)//trim(str)//trim(param_file))
103
```

Граф вызовов:



5.3.1.2 set_fitting_parameters()

subroutine fit_gr_moire::set_fitting_parameters (
character(len=256) fitting_parameters_file_name,

real, dimension(4) init_min_params,
real, dimension(4) init_max_params)

См. определение в файле fit_gr_moire.f90 строка 107

```
107 real                    :: init_min_params(4),init_max_params(4)
108 character(len=256)      :: fitting_parameters_file_name,min_param_file,max_param_file,str
109 integer                 :: i
110
111     final_out_id = 1211
112     oid = 111
113     do i=1,80
114         line(i:i) = '_'
115     enddo
116
117     open(9,file=fitting_parameters_file_name)
118     read(9,'(A16,A)') str,input_path;       write(out_id,*) trim(str),'  ',trim(input_path)
119     read(9,'(A16,A)') str,out_path;         write(out_id,*) trim(str),'  ',trim(out_path)
120     read(9,*) str,interaction_name;         write(out_id,*) trim(str),'  ',trim(interaction_name)
121     read(9,*) str,output_prefix;            write(out_id,*) trim(str),'  ',trim(output_prefix)
122     read(9,*) str,ar_settings_filename(1); write(out_id,*) trim(str),'  ',trim(ar_settings_filename(1))
123     read(9,*) str,ar_start_xyz_file(1);     write(out_id,*) trim(str),'  ',trim(ar_start_xyz_file(1))
124     read(9,*) str,ar_c_num(1);              write(out_id,*) trim(str),'  ',ar_c_num(1)
125     read(9,*) str,ar_zero_energy_level(1); write(out_id,*) trim(str),'  ',ar_zero_energy_level(1)
126     read(9,*) str,ar_final_file(1);         write(out_id,*) trim(str),'  ',trim(ar_final_file(1))
127     read(9,*) str,ar_settings_filename(2); write(out_id,*) trim(str),'  ',trim(ar_settings_filename(2))
128     read(9,*) str,ar_start_xyz_file(2);     write(out_id,*) trim(str),'  ',trim(ar_start_xyz_file(2))
129     read(9,*) str,ar_c_num(2);              write(out_id,*) trim(str),'  ',ar_c_num(2)
130     read(9,*) str,ar_zero_energy_level(2); write(out_id,*) trim(str),'  ',ar_zero_energy_level(2)
131     read(9,*) str,ar_final_file(2);         write(out_id,*) trim(str),'  ',trim(ar_final_file(2))
132     read(9,*) str,min_param_file;           write(out_id,*) trim(str),'  ',trim(min_param_file)
133     read(9,*) str,max_param_file;           write(out_id,*) trim(str),'  ',trim(max_param_file)
134     read(9,*) str,z;                      write(out_id,*) trim(str),'  ',z
135     read(9,*) str,be0;                     write(out_id,*) trim(str),'  ',be0
136     read(9,*) str,ar_grd0(1);               write(out_id,*) trim(str),'  ',ar_grd0(1)
137     read(9,*) str,ar_grd0(2);               write(out_id,*) trim(str),'  ',ar_grd0(2)
138     close(9)
139
140
141     simplified = .false.
142     if(interaction_name=='ljc') then
143         open(1234,file=trim(input_path)//trim(min_param_file))
144         read(1234,*) init_min_params(3),init_min_params(1),init_min_params(2)
145         read(1234,*) rcut(1),rcut(2)
146         close(1234)
147         open(1234,file=trim(input_path)//trim(max_param_file))
148         read(1234,*) init_max_params(3),init_max_params(1),init_max_params(2)
149         close(1234)
150         init_min_params(4) = 0.
151         init_max_params(4) = 0.
152     endif
153     if(interaction_name=='morsec') then
154         open(1234,file=trim(input_path)//trim(min_param_file))
155         read(1234,*) init_min_params(3),init_min_params(1),init_min_params(4),init_min_params(2)
156         read(1234,*) rcut(1),rcut(2)
157         close(1234)
158         open(1234,file=trim(input_path)//trim(max_param_file))
159         read(1234,*) init_max_params(3),init_max_params(1),init_max_params(4),init_max_params(2)
160         close(1234)
161     endif
162
163     sim_num = 0
164
165     open(1234,file=trim(input_path)//trim(ar_settings_filename(1)))
166     read(1234,*);read(1234,*);read(1234,*) str,ar_xyz_file(1)
167     do while(.true.)
168         read(1234,'(A128)') str
169         if(str(1:3)==trim(interaction_name)) then
170             read(str(4:),*) param_file
171             exit
172         endif
173         if(str(1:6)==trim(interaction_name)) then
174             read(str(7:),*) param_file
175             exit
176         endif
177     enddo
178     close(1234)
179     open(1234,file=trim(input_path)//trim(ar_settings_filename(2)))
180     read(1234,*);read(1234,*);read(1234,*) str,ar_xyz_file(2)
181     close(1234)
182
```

5.3.2 Переменные

5.3.2.1 ar_c_num

integer, dimension(2) fit_gr_moire::ar_c_num

См. определение в файле fit_gr_moire.f90 строка 7

```
7 integer                :: ar_c_num(2)
```

5.3.2.2 ar_final_file

character(len=256), dimension(2) fit_gr_moire::ar_final_file

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.3 ar_grd0

real, dimension(2) fit_gr_moire::ar_grd0

См. определение в файле fit_gr_moire.f90 строка 10

5.3.2.4 ar_settings_filename

character(len=256), dimension(2) fit_gr_moire::ar_settings_filename

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.5 ar_start_xyz_file

character(len=256), dimension(2) fit_gr_moire::ar_start_xyz_file

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.6 ar_xyz_file

character(len=256), dimension(2) fit_gr_moire::ar_xyz_file

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.7 ar_zero_energy_level

real, dimension(2) fit_gr_moire::ar_zero_energy_level

См. определение в файле fit_gr_moire.f90 строка 10

5.3.2.8 be0

real fit_gr_moire::be0

См. определение в файле fit_gr_moire.f90 строка 10

5.3.2.9 final_out_id

integer fit_gr_moire::final_out_id

См. определение в файле fit_gr_moire.f90 строка 6

5.3.2.10 input_path

character(len=256) fit_gr_moire::input_path

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.11 interaction_name

character(len=256) fit_gr_moire::interaction_name

См. определение в файле fit_gr_moire.f90 строка 8

```
8 character(len=256)      :: interaction_name,ar_settings_filename(2),output_prefix,input_path,out_path,&
9                           ar_final_file(2),param_file,ar_start_xyz_file(2),ar_xyz_file(2)
```

5.3.2.12 line

character(len=80) fit_gr_moire::line

См. определение в файле fit_gr_moire.f90 строка 12

```
12 character(len=80)       :: line
```

5.3.2.13 num_of_omp_treads

integer fit_gr_moire::num_of_omp_treads

См. определение в файле fit_gr_moire.f90 строка 6

5.3.2.14 oid

integer fit_gr_moire::oid

См. определение в файле fit_gr_moire.f90 строка 6

5.3.2.15 out_id

integer fit_gr_moire::out_id

См. определение в файле fit_gr_moire.f90 строка 6

5.3.2.16 out_path

character(len=256) fit_gr_moire::out_path

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.17 out_period

integer fit_gr_moire::out_period

См. определение в файле fit_gr_moire.f90 строка 6

5.3.2.18 output_prefix

character(len=256) fit_gr_moire::output_prefix

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.19 param_file

character(len=256) fit_gr_moire::param_file

См. определение в файле fit_gr_moire.f90 строка 8

5.3.2.20 rcut

real, dimension(2) fit_gr_moire::rcut

См. определение в файле fit_gr_moire.f90 строка 10

5.3.2.21 sim_num

integer fit_gr_moire::sim_num

См. определение в файле fit_gr_moire.f90 строка 6

```
6 integer                 :: sim_num,out_period,num_of_omp_treads,out_id,final_out_id,oid
```

5.3.2.22 simplified

logical fit_gr_moire::simplified

См. определение в файле fit_gr_moire.f90 строка 11

```
11 logical                :: simplified
```

5.3.2.23 z

real fit_gr_moire::z

См. определение в файле fit_gr_moire.f90 строка 10

```
10 real                 :: z,ar_zero_energy_level(2),be0,ar_grd0(2),Rcut(2)
```

5.4 Модуль graphene_on_surface_analysis

Функции/подпрограммы

- subroutine gr_on_cu_analysis (arr1, arr2, filename, z)

5.4.1 Функции/подпрограммы

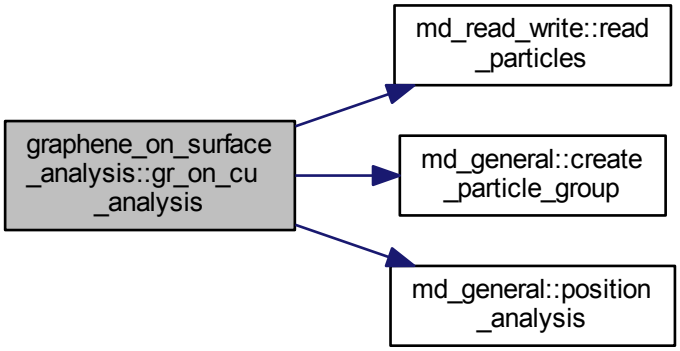
5.4.1.1 gr_on_cu_analysis()

```
subroutine graphene_on_surface_analysis::gr_on_cu_analysis (
                real, dimension(3) arr1,
                real, dimension(3) arr2,
                character(len=256) filename,
                real z )
```

См. определение в файле graphene_on_surface_analysis.f90 строка 9

```
9    type(particles)                    :: atoms
10    type(particle_group)                  :: groupC,groupCU
11    character(len=256)                    :: filename
12    character(len=32)                     :: type_names(3)
13    real                               :: z,arr1(3),arr2(3)
14
15    call read_particles(atoms,filename)
16    type_names(1) = 'C';    type_names(2) = 'C_a';      type_names(3) = 'C_b'
17    call create_particle_group(groupc,type_names,atoms)
18    call position_analysis(arr1(1),arr1(2),arr1(3),atoms,groupc,3,z,1000.)
19    type_names(1) = 'CU';   type_names(2) = 'CU_fixed'; type_names(3) = '#'
20    call create_particle_group(groupcu,type_names,atoms)
21    call position_analysis(arr2(1),arr2(2),arr2(3),atoms,groupcu,3,z,1000.)
22    deallocate(atoms%positions)
23    deallocate(atoms%velocities)
24    deallocate(atoms%forces)
25    deallocate(atoms%masses)
26    deallocate(atoms%atom_types)
27    deallocate(groupc%indexes)
28    deallocate(groupcu%indexes)
29
```

Граф вызовов:



5.5 Модуль graphenenorm

Функции/подпрограммы

- subroutine find_gr_nearest_neighbors (nl_nn, nl)
- subroutine find_norm_in_graphene (gr_norm, dr_nn)
- subroutine update_nearest_neighbours_in_graphene (md_step, nl_nn, nl, atoms, group, box)

5.5.1 Функции/подпрограммы

5.5.1.1 find_gr_nearest_neighbors()

```
subroutine graphenenorm::find_gr_nearest_neighbors (
                type(neighbour_list) nl_nn,
                type(neighbour_list) nl )
```

См. определение в файле graphenenorm.f90 строка 9

```
9     type(neighbour_list):: nl,nl_nn
10     integer:: nnum_nn,i,p,k
11
12     nnum_nn = 3
13     if(nl_nn%neighb_num_max/=nnum_nn) then; write(*,*) 'error: nl_nn%neighb_num_max/=nnum_nn'; stop; endif
14     !$OMP PARALLEL firstprivate(i,p,k)
15     !$OMP DO
16     do i=1,nl%N
17        k = 0
18        do p=1,nl%nnum(i)
19           if (nl%moddr(p,i)<nl_nn%r_cut) then
20              k = k+1
21              if (k<=nnum_nn) then
22                 nl_nn%nlist(k,i) = nl%nlist(p,i)
23                 nl_nn%moddr(k,i) = nl%moddr(p,i)
```

```
24                 nl_nn%dr(:,k,i) = nl%dr(:,p,i)
25               else
26                 write(*,*) 'error: too many gr nearest neibs',i,p,nl%moddr(p,i); stop;
27               endif
28             endif
29         enddo
30         if (k/=nnum_nn) then; write(*,*) 'error: not enough gr nearest neibs',i,nl_nn%moddr(:,i),nl_nn
   %nlist(:,i); stop; endif;
31         nl_nn%nnum(i) = nnum_nn
32       enddo
33       !$OMP END DO
34       !$OMP END PARALLEL
35
```

5.5.1.2 find_norm_in_graphene()

subroutine graphenenorm::find_norm_in_graphene (
real, dimension(:,:) gr_norm,
real, dimension(:,:,:) dr_nn)

См. определение в файле graphenenorm.f90 строка 39

```
39     integer:: nnum_nn,i,p,k
40     real:: dr_nn(:,:,:),gr_norm(:,:),drj12(3),drj31(3)
41
42     nnum_nn = 3
43     !$OMP PARALLEL firstprivate(i,p,k,drj12,drj31)
44     !$OMP DO
45     do i=1,size(gr_norm(1,:))
46         drj12 = dr_nn(:,2,i)-dr_nn(:,1,i)
47         drj31 = dr_nn(:,1,i)-dr_nn(:,3,i)
48         do k=1,3
49             gr_norm(k,i) = (drj12(mod(k,3)+1))*(drj31(mod(k+1,3)+1))-(drj12(mod(k+1,3)+1))*(drj31(mod(k,3)+
   1))
50         enddo
51         if (gr_norm(3,i)<0.) gr_norm(:,i)=-gr_norm(:,i)
52         gr_norm(:,i) = gr_norm(:,i)/sqrt(sum(gr_norm(:,i)**2))
53     enddo
54     !$OMP END DO
55     !$OMP END PARALLEL
```

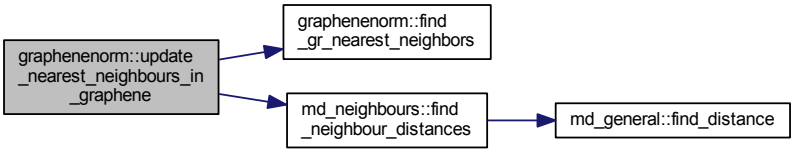
5.5.1.3 update_nearest_neighbours_in_graphene()

subroutine graphenenorm::update_nearest_neighbours_in_graphene (
integer md_step,
type(neighbour_list) nl_nn,
type(neighbour_list) nl,
type(particles) atoms,
type(particle_group) group,
type(simulation_cell) box)

См. определение в файле graphenenorm.f90 строка 59

```
59     type(particles):: atoms
60     type(neighbour_list):: nl,nl_nn
61     type(particle_group):: group
62     type(simulation_cell):: box
63     integer:: md_step
64
65     if (mod(md_step,nl%update_period)==0) then
66         call find_gr_nearest_neighbors(nl_nn,nl)
67     else
68         call find_neighbour_distances(nl_nn,atoms,group,group,box)
69     endif
70
```

Граф вызовов:



5.6 Модуль ifport

Заглушка для удобства компиляции. В IFPORT находится функция rand() при компиляции с ifort. При компиляции с gfortran такого модуля нет. Этот пустой модуль нужен чтобы не убирать use IFPORT в md_general при компиляции с gfortran.

5.6.1 Подробное описание

Заглушка для удобства компиляции. В IFPORT находится функция rand() при компиляции с ifort. При компиляции с gfortran такого модуля нет. Этот пустой модуль нужен чтобы не убирать use IFPORT в md_general при компиляции с gfortran.

5.7 Модуль lennardjones

Типы данных

- type lennardjones_parameters

Функции/подпрограммы

- subroutine read_lj_parameters (LJp, filename)
- subroutine lj_energy (energy, nl, LJp)
- subroutine lj_forces (atoms, nl, LJp)

5.7.1 Функции/подпрограммы

5.7.1.1 lj_energy()

```
subroutine lennardjones::lj_energy (
                real energy,
                type(neighbour_list) nl,
                type(lennardjones_parameters) LJp )
```

См. определение в файле LennardJones.f90 строка 24

```
24    type(neighbour_list):: nl
25    type(LennardJones_parameters):: LJp
26    integer:: i,p
27    real:: energy,energy_priv,V
28
29    energy = 0.
30    energy_priv = 0.
31    !$OMP PARALLEL firstprivate(energy_priv,i,p,V)
32    !$OMP DO
33    do i=1,nl%N
34       do p=1,nl%nnum(i)
35          if (nl%moddr(p,i)<ljp%R2) then
36             v = (ljp%sig/nl%moddr(p,i))**6
37             energy_priv = energy_priv+4*ljp%eps*v*(v-1.)*f_cut(nl%moddr(p,i),ljp%R1,ljp%R2)
38          endif
39       enddo
40    enddo
41    !$OMP END DO
42    !$OMP ATOMIC
43       energy = energy+energy_priv
44    !$OMP END PARALLEL
45
```

Граф вызовов:

5.7.1.2 lj_forces()

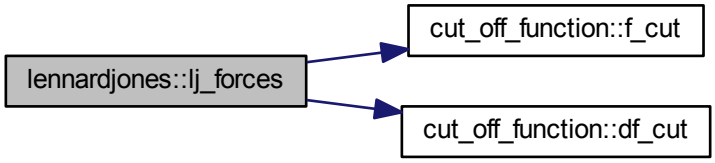
```
subroutine lennardjones::lj_forces (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(lennardjones_parameters) LJp )
```

См. определение в файле LennardJones.f90 строка 49

```
49    type(particles):: atoms
50    type(neighbour_list):: nl
51    type(LennardJones_parameters):: LJp
52    integer:: i,p
53    real:: V
54
55    !$OMP PARALLEL firstprivate(i,p,V)
56    !$OMP DO
```

```
57    do i=1,nl%N
58       do p=1,nl%nnum(i)
59          if (nl%moddr(p,i)<ljp%R2) then
60             v = (ljp%sig/nl%moddr(p,i))**6
61             atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))-4.*ljp%eps*&
62             (v*(12.*v-6.)/nl%moddr(p,i)**2*f_cut(nl%moddr(p,i),ljp%R1,ljp%R2)-v*(v-1.)*df_cut(nl%moddr(
   p,i),ljp%R1,ljp%R2))*nl%dr(:,p,i)
63          endif
64       enddo
65    enddo
66    !$OMP END DO
67    !$OMP END PARALLEL
68
```

Граф вызовов:



5.7.1.3 read_lj_parameters()

```
subroutine lennardjones::read_lj_parameters (
                type(lennardjones_parameters) LJp,
                character(*) filename )
```

См. определение в файле LennardJones.f90 строка 13

```
13    type(LennardJones_parameters):: LJp
14    character(*):: filename
15
16    open(1,file=filename)
17    read(1,*) ljp%eps,ljp%sig
18    read(1,*) ljp%R1,ljp%R2
19    close(1)
20
```

5.8 Модуль lennardjones_1g

Типы данных

- type lennardjones1g_parameters

Функции/подпрограммы

- subroutine read_lj1g_parameters (LJp, filename)
- subroutine lj1g_energy (energy, nl, LJp)
- subroutine lj1g_forces (atoms, nl, LJp)
- real function scalar_lj_force (r, R1, R2, c12, c6, c12t12, c6t6)

5.8.1 Функции/подпрограммы

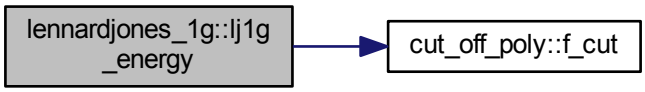
5.8.1.1 lj1g_energy()

```
subroutine lennardjones_1g::lj1g_energy (
                real energy,
                type(neighbour_list) nl,
                type(lennardjones1g_parameters) LJp )
```

См. определение в файле LennardJones_1g.f90 строка 29

```
29    type(neighbour_list):: nl
30    type(LennardJones1g_parameters):: LJp
31    integer:: i,p
32    real:: energy,energy_priv,U
33
34    energy = 0.
35    energy_priv = 0.
36    !$OMP PARALLEL firstprivate(energy_priv) private(i,p,U)
37    !$OMP DO schedule(dynamic,chunk_size)
38    do i=1,nl%N
39       do p=1,nl%lessnnum(i)!do p=1,nl%nnum(i)!if (nl%nlist(p,i)>i) exit
40          u = 1./(nl%moddr(p,i)*nl%moddr(p,i)*nl%moddr(p,i)*nl%moddr(p,i)*nl%moddr(p,i)*nl%moddr(p,i))
41          energy_priv = energy_priv+u*(ljp%c12*u-ljp%c6)*f_cut(nl%moddr(p,i),ljp%R1,ljp%R2)
42       enddo
43    enddo
44    !$OMP END DO
45    !$OMP ATOMIC
46       energy = energy+energy_priv
47    !$OMP END PARALLEL
48
```

Граф вызовов:



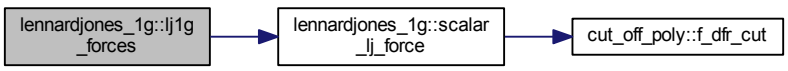
5.8.1.2 lj1g_forces()

```
subroutine lennardjones_1g::lj1g_forces (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(lennardjones1g_parameters) LJp )
```

См. определение в файле LennardJones_1g.f90 строка 52

```
52    type(particles) ::  atoms
53    type(neighbour_list) :: nl
54    type(LennardJones1g_parameters) :: LJp
55    integer:: i,p,k,ind,jnd
56    real,allocatable:: priv_force(:,:),F(:),fp(:,:)
57
58    !$OMP PARALLEL private(i,p,k,ind,jnd,F,fp,priv_force)
59    if(.not. allocated(priv_force)) allocate(priv_force(3,atoms%N))!realloc if N changed
60    if(.not. allocated(f)) allocate(f(nl%neighb_num_max))!realloc if neighb_num_max changed
61    if(.not. allocated(fp)) allocate(fp(3,nl%neighb_num_max))!realloc if neighb_num_max changed
62    f = 0.
63    fp = 0.
64    priv_force = 0.
65    !$OMP DO SCHEDULE(dynamic,chunk_size)
66    do i=1,nl%N
67       do p=1,nl%lessnnum(i)
68          f(p) = scalar_lj_force(nl%moddr(p,i),ljp%R1,ljp%R2,ljp%c12,ljp%c6,ljp%c12t12,ljp%c6t6)
69       enddo
70       do p=1,nl%lessnnum(i)
71          fp(1,p) = f(p)*nl%dr(1,p,i)
72          fp(2,p) = f(p)*nl%dr(2,p,i)
73          fp(3,p) = f(p)*nl%dr(3,p,i)
74       enddo
75       do p=1,nl%lessnnum(i)
76          ind = nl%particle_index(i)
77          jnd = nl%particle_index(nl%nlist(p,i))
78          priv_force(1,ind) = priv_force(1,ind)-fp(1,p)
79          priv_force(2,ind) = priv_force(2,ind)-fp(2,p)
80          priv_force(3,ind) = priv_force(3,ind)-fp(3,p)
81          priv_force(1,jnd) = priv_force(1,jnd)+fp(1,p)
82          priv_force(2,jnd) = priv_force(2,jnd)+fp(2,p)
83          priv_force(3,jnd) = priv_force(3,jnd)+fp(3,p)
84       enddo
85    enddo
86    !$OMP END DO
87    do i=1,nl%N
88       do k=1,3
89          !$OMP ATOMIC
90             atoms%forces(k,i) = atoms%forces(k,i)+priv_force(k,i)
91       enddo
92    enddo
93    !$OMP END PARALLEL
94
```

Граф вызовов:



5.8.1.3 read_lj1g_parameters()

```
subroutine lennardjones_1g::read_lj1g_parameters (
                type(lennardjones1g_parameters) LJp,
                character(*) filename )
```

См. определение в файле LennardJones_1g.f90 строка 14

```
14    type(LennardJones1g_parameters):: LJp
15    character(*):: filename
16
17    open(1,file=filename)
18    read(1,*) ljp%eps,ljp%sig
19    read(1,*) ljp%R1,ljp%R2
```

```
20     close(1)
21     ljp%c6 = 4.*ljp%eps*ljp%sig**6
22     ljp%c12 = 4.*ljp%eps*ljp%sig**12
23     ljp%c6t6 = 6.*4.*ljp%eps*ljp%sig**6
24     ljp%c12t12 = 12.*4.*ljp%eps*ljp%sig**12
25
```

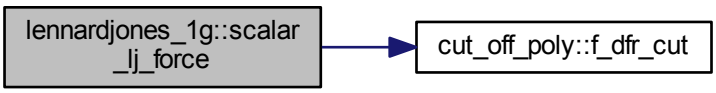
5.8.1.4 scalar_lj_force()

```
real function lennardjones_1g::scalar_lj_force (
                real, intent(in) r,
                real, intent(in) R1,
                real, intent(in) R2,
                real, intent(in) c12,
                real, intent(in) c6,
                real, intent(in) c12t12,
                real, intent(in) c6t6 )
```

См. определение в файле LennardJones_1g.f90 строка 98

```
98     !$OMP DECLARE SIMD(scalar_lj_force) UNIFORM(R1,R2,c12,c6,c12t12,c6t6)
99     real, intent(in) :: r
100     real, intent(in) :: R1,R2,c12,c6,c12t12,c6t6
101     real scalar_lj_force
102     real :: invr2,U,fcut,dfrcut
103
104     invr2 = 1./(r*r)
105     u = invr2*invr2*invr2
106     call f_dfr_cut(fcut,dfrcut,r,r1,r2)
107     scalar_lj_force = u*invr2*((c12t12*u-c6t6)*fcut-(c12*u-c6)*dfrcut)
108
```

Граф вызовов:



5.9 Модуль lennardjonescosine

Типы данных

- type lennardjonescosine_parameters

Функции/подпрограммы

- subroutine read_ljc_parameters (LJCp, filename)
- subroutine ljc_energy (energy, nl, LJCp)
- subroutine ljc_forces_for_graphene (atoms, nl, nl_nn, LJCp)
- subroutine ljc_forces_for_other_atoms (atoms, nl, LJCp)

5.9.1 Функции/подпрограммы

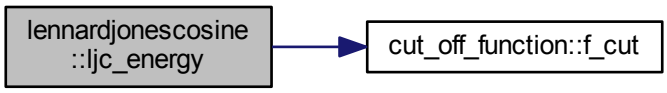
5.9.1.1 ljc_energy()

```
subroutine lennardjonescosine::ljc_energy (
                real energy,
                type(neighbour_list) nl,
                type(lennardjonescosine_parameters) LJCp )
```

См. определение в файле LennardJonesCosine.f90 строка 27

```
27    type(neighbour_list):: nl
28    type(LennardJonesCosine_parameters):: LJCp
29    integer:: i,p
30    real:: energy,energy_priv,V1,V2,V3
31
32    energy = 0.
33    energy_priv = 0.
34    !$OMP PARALLEL firstprivate(energy_priv,i,p,V1,V2,V3)
35    !$OMP DO
36    do i=1,nl%N
37       do p=1,nl%nnum(i)
38          if (nl%moddr(p,i)<ljcp%R2) then
39             v2 = (ljcp%sig/nl%moddr(p,i))**6
40             v1 = v2**2
41             v3 = (abs(sum(ljcp%gr_norm(:,i)*nl%dr(:,p,i)))/nl%moddr(p,i))**ljcp%delt
42             energy_priv = energy_priv+4*ljcp%eps*(v1-v2*v3)*f_cut(nl%moddr(p,i),ljcp%R1,ljcp%R2)
43          endif
44       enddo
45    enddo
46    !$OMP END DO
47    !$OMP ATOMIC
48       energy = energy+energy_priv
49    !$OMP END PARALLEL
50
```

Граф вызовов:



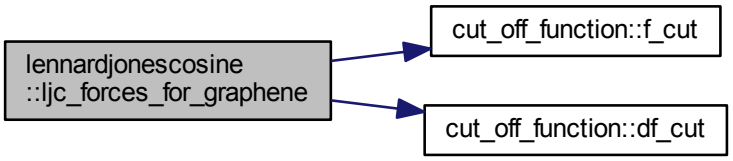
5.9.1.2 ljc_forces_for_graphene()

```
subroutine lennardjonescosine::ljc_forces_for_graphene (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(neighbour_list) nl_nn,
                type(lennardjonescosine_parameters) LJCp )
```

См. определение в файле LennardJonesCosine.f90 строка 54

```
54    type(particles)::   atoms
55    type(neighbour_list):: nl,nl_nn
56    type(LennardJonesCosine_parameters):: LJCp
57    integer:: i,p,q,l1,l2,l3,j,k,nnum_nn
58    real:: drj12(3),drj31(3),drj23(3),V1,V2,V3,f_c,df_c
59
60    if (nl%N/=nl_nn%N .and. nl%N/=0) then; write(*,*) 'error: nl%N/=nl_nn%N',nl%N,nl_nn%N; stop; endif
61    nnum_nn = 3
62    if (ljcp%simplified) then; nnum_nn = 0; ljcp%gr_norm = 0.; ljcp%gr_norm(3,:) = 1.; endif!optimize
63
64    !$OMP PARALLEL firstprivate(i,p,q,l1,l2,l3,j,k,drj12,drj31,drj23,V1,V2,V3,f_c,df_c)
65    !$OMP DO
66    do i=1,nl%N
67       do p=1,nl%nnum(i)
68          if (nl%moddr(p,i)<ljcp%R2) then
69             v2 = (ljcp%sig/nl%moddr(p,i))**6
70             v1 = v2**2
71             v3 = (abs(sum(ljcp%gr_norm(:,i)*nl%dr(:,p,i)))/nl%moddr(p,i))**ljcp%delt
72             f_c = f_cut(nl%moddr(p,i),ljcp%R1,ljcp%R2)
73             df_c = df_cut(nl%moddr(p,i),ljcp%R1,ljcp%R2)
74             atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))-4.*ljcp%eps*&
75             (((12.*v1-(6.+ljcp%delt)*v2*v3)/nl%moddr(p,i)**2*f_c-(v1-v2*v3)*df_c)*nl%dr(:,p,i)+&
76             (ljcp%delt*v2*v3/sum(ljcp%gr_norm(:,i)*nl%dr(:,p,i))*f_c)*ljcp%gr_norm(:,i))
77          endif
78       enddo
79    enddo
80    !$OMP END DO
81    !$OMP DO
82    do i=1,nl%N
83       do q=1,nnum_nn
84          j = nl_nn%nlist(q,i)
85          do l1=1,nnum_nn
86             if (nl_nn%nlist(l1,j)==i) exit
87          enddo
88          if(l1>3) then; write(*,*) 'l1>3'; stop; endif
89          l2 = mod(l1,3)+1
90          l3 = mod(l1+1,3)+1
91          drj12 = nl_nn%dr(:,l2,j)-nl_nn%dr(:,l1,j)
92          drj31 = nl_nn%dr(:,l1,j)-nl_nn%dr(:,l3,j)
93          drj23 = nl_nn%dr(:,l3,j)-nl_nn%dr(:,l2,j)
94          do p=1,nl%nnum(j)
95             if (nl%moddr(p,j)<ljcp%R2) then
96                v2 = (ljcp%sig/nl%moddr(p,j))**6
97                v1 = v2**2
98                v3 = (abs(sum(ljcp%gr_norm(:,j)*nl%dr(:,p,j)))/nl%moddr(p,j))**ljcp%delt
99                f_c = f_cut(nl%moddr(p,j),ljcp%R1,ljcp%R2)
100                atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))&
101                -4.*ljcp%eps*ljcp%delt*v2*v3*f_c/(sum(drj12**2)*sum(drj31**2)-sum(drj12*drj31)**2)/sum(
    ljcp%gr_norm(:,j)*nl%dr(:,p,j))*&
102                ljcp%gr_norm(:,j)*sum(nl%dr(:,p,j)*(drj12*sum(drj23*drj31)-drj31*sum(drj23*drj12)))
103             endif
104          enddo
105       enddo
106    enddo
107    !$OMP END DO
108    !$OMP END PARALLEL
109
```

Граф вызовов:



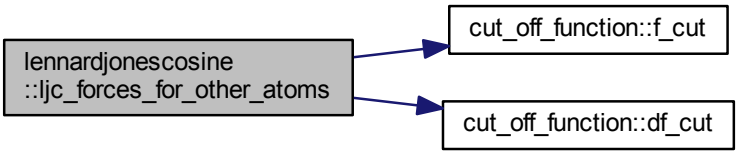
5.9.1.3 ljc_forces_for_other_atoms()

```
subroutine lennardjonescosine::ljc_forces_for_other_atoms (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(lennardjonescosine_parameters) LJCp )
```

См. определение в файле LennardJonesCosine.f90 строка 113

```
113     type(particles)::   atoms
114     type(neighbour_list):: nl
115     type(LennardJonesCosine_parameters):: LJCp
116     integer:: i,p
117     real:: gr_n(3),V1,V2,V3,f_c,df_c
118
119     !$OMP PARALLEL firstprivate(i,p,gr_n,V1,V2,V3,f_c,df_c)
120     !$OMP DO
121     do i=1,nl%N
122        do p=1,nl%nnum(i)
123           if (nl%moddr(p,i)<ljcp%R2) then
124              gr_n = ljcp%gr_norm(:,nl%nlist(p,i))
125              v2 = (ljcp%sig/nl%moddr(p,i))**6
126              v1 = v2**2
127              v3 = (abs(sum(gr_n*nl%dr(:,p,i)))/nl%moddr(p,i))**ljcp%delt
128              f_c = f_cut(nl%moddr(p,i),ljcp%R1,ljcp%R2)
129              df_c = df_cut(nl%moddr(p,i),ljcp%R1,ljcp%R2)
130              atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))-4.*ljcp%eps*&
131              (((12.*v1-(6.+ljcp%delt)*v2*v3)/nl%moddr(p,i)**2*f_c-(v1-v2*v3)*df_c)*nl%dr(:,p,i)+&
132              (ljcp%delt*v2*v3/sum(gr_n*nl%dr(:,p,i))*f_c)*gr_n)
133           endif
134        enddo
135     enddo
136     !$OMP END DO
137     !$OMP END PARALLEL
138
```

Граф вызовов:



5.9.1.4 read_ljc_parameters()

```
subroutine lennardjonescosine::read_ljc_parameters (
                type(lennardjonescosine_parameters) LJCp,
                character(*) filename )
```

См. определение в файле LennardJonesCosine.f90 строка 15

```
15      type(LennardJonesCosine_parameters):: LJCp
16      character(*):: filename
17
18      open(1,file=filename)
19      read(1,*) ljcp%eps,ljcp%sig,ljcp%delt
20      read(1,*) ljcp%R1,ljcp%R2
21      read(1,*) ljcp%simplified
22      close(1)
23
```

5.10 Модуль md_general

Типы данных

- type integrator_params
- type neighbour_list
- type nose_hoover_chain
- type particle_group
- type particles
- type simulation_cell
- type time_steps

Функции/подпрограммы

- subroutine init_time_steps (dt, delta_t)
- subroutine create_particle_group (group, type_names, atoms)
- subroutine change_particle_group_n (group, md_step, change_ts1, change_ts2, change_frec, init_group)
- subroutine scale_velocities (atoms, group, s)
- subroutine random_velocities (atoms, group)
- subroutine random_momenta (atoms, group)
- subroutine calculate_kinetic_energy (ke, atoms, group)
- subroutine calculate_mass_center (mc, atoms, group)
- subroutine calculate_mass_center_velosity (mcv, atoms, group)
- subroutine zero_momentum (atoms, group)
- subroutine calculate_masses_sum (totm, atoms, group)
- subroutine calculate_force_sum (fs, atoms, group)
- subroutine calculate_temperature (temp, ke, atoms, group)
- subroutine set_new_temperature (atoms, group, temp)
- subroutine check_positions (out_id, atoms, box)
- subroutine check_velocities (out_id, atoms)
- subroutine invert_z_velocities (atoms, z_low_border, z_high_border)
- subroutine position_analysis (av, mi, ma, atoms, group, direction, minimum, maximum)
- pure subroutine find_distance (dr, dr2, vec1, vec2, box)

5.10.1 Функции/подпрограммы

5.10.1.1 calculate_force_sum()

```
subroutine md_general::calculate_force_sum (
                real, dimension(3) fs,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 279

```
279     type(particles)::   atoms
280     type(particle_group):: group
281     real:: fs(3),fs_priv(3)
282     integer::       i,ind,k
283
284     fs = 0.
285     fs_priv = 0.
286     !$OMP PARALLEL firstprivate(fs_priv,i,ind,k)
287     !$OMP DO
288     do ind=1,group%N
289         i = group%indexes(ind)
290         fs_priv = fs_priv+atoms%forces(:,i)
291     enddo
292     !$OMP END DO
293     do k=1,3
294         !$OMP ATOMIC
295             fs(k) = fs(k)+fs_priv(k)
296     enddo
297     !$OMP END PARALLEL
298
299     return
```

5.10.1.2 calculate_kinetic_energy()

```
subroutine md_general::calculate_kinetic_energy (
                real ke,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 163

```
163     type(particles)::   atoms
164     type(particle_group):: group
165     real:: ke,ke_priv
166     real,parameter:: mass_coef=1.6605389217/1.6021765654*10.**(2)
167     integer::       i,ind
168
169     ke = 0.
170     ke_priv = 0.
171     !$OMP PARALLEL firstprivate(ke_priv,i,ind)
172     !$OMP DO
173     do ind=1,group%N
174         i = group%indexes(ind)
175         ke_priv = ke_priv+atoms%masses(i)*sum(atoms%velocities(:,i)**2)/2*mass_coef
176     enddo
177     !$OMP END DO
178     !$OMP ATOMIC
179         ke = ke+ke_priv
180     !$OMP END PARALLEL
181
182     return
```

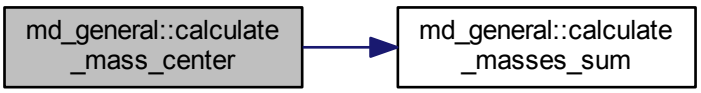
5.10.1.3 calculate_mass_center()

```
subroutine md_general::calculate_mass_center (
                real, dimension(3) mc,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 186

```
186     type(particles)::   atoms
187     type(particle_group):: group
188     real:: mc(3),mc_priv(3),totm
189     integer::       i,ind,k
190
191     mc = 0.
192     mc_priv = 0.
193     !$OMP PARALLEL firstprivate(mc_priv,i,ind,k)
194     !$OMP DO
195     do ind=1,group%N
196        i = group%indexes(ind)
197        mc_priv = mc_priv+atoms%masses(i)*atoms%positions(:,i)
198     enddo
199     !$OMP END DO
200     do k=1,3
201        !$OMP ATOMIC
202           mc(k) = mc(k)+mc_priv(k)
203     enddo
204     !$OMP END PARALLEL
205     call calculate_masses_sum(totm,atoms,group)
206     mc = mc/totm
207
208     return
```

Граф вызовов:



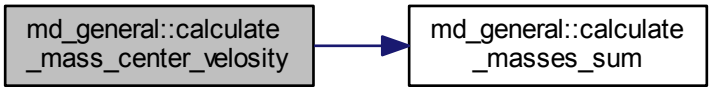
5.10.1.4 calculate_mass_center_velosity()

```
subroutine md_general::calculate_mass_center_velosity (
                real, dimension(3) mcv,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 212

```
212     type(particles)::   atoms
213     type(particle_group):: group
214     real:: mcv(3),mcv_priv(3),totm
215     integer::       i,ind,k
216
217     mcv = 0.
218     mcv_priv = 0.
219     !$OMP PARALLEL firstprivate(mcv_priv,i,ind,k)
220     !$OMP DO
221     do ind=1,group%N
222        i = group%indexes(ind)
223        mcv_priv = mcv_priv+atoms%masses(i)*atoms%velocities(:,i)
224     enddo
225     !$OMP END DO
226     do k=1,3
227        !$OMP ATOMIC
228           mcv(k) = mcv(k)+mcv_priv(k)
229     enddo
230     !$OMP END PARALLEL
231     call calculate_masses_sum(totm,atoms,group)
232     mcv = mcv/totm
233
234     return
```

Граф вызовов:



5.10.1.5 calculate_masses_sum()

```
subroutine md_general::calculate_masses_sum (
                real totm,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 257

```
257     type(particles)::   atoms
258     type(particle_group):: group
259     real:: totm,totm_priv
260     integer::       i,ind
261
262     totm = 0.
263     totm_priv = 0.
264     !$OMP PARALLEL firstprivate(totm_priv,i,ind)
265     !$OMP DO
266     do ind=1,group%N
267        i = group%indexes(ind)
268        totm_priv = totm_priv+atoms%masses(i)
269     enddo
270     !$OMP END DO
271     !$OMP ATOMIC
272        totm = totm+totm_priv
273     !$OMP END PARALLEL
274
275     return
```

5.10.1.6 calculate_temperature()

```
subroutine md_general::calculate_temperature (
                real temp,
                real ke,
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 303

```
303     type(particles)::   atoms
304     type(particle_group):: group
305     real,parameter::    kt_a_degree =1.3806488/1.6021765654*10.**(-4)
306     real:: temp,ke
307
308     call calculate_kinetic_energy(ke,atoms,group)
309     temp = 2*ke/kt_a_degree/(3*(group%N))
310
311     return
```

Граф вызовов:

5.10.1.7 change_particle_group_n()

```
subroutine md_general::change_particle_group_n (
                type(particle_group) group,
                integer md_step,
                integer change_ts1,
                integer change_ts2,
                integer change_frec,
                type(particle_group) init_group )
```

См. определение в файле md_general.f90 строка 83

```
83     type(particle_group):: group,init_group
84     integer:: md_step,change_ts1,change_ts2,change_frec
85
86     if(md_step<=change_ts1) then
87        group%N = init_group%N
88        if(md_step==change_ts1) group%N = group%N+1
89     else
90        if(md_step<change_ts2 .and. mod(md_step-change_ts1,change_frec)==0) group%N = group%N+1
91     endif
92     if(group%N>size(group%indexes)) group%N = size(group%indexes)
93
```

5.10.1.8 check_positions()

```
subroutine md_general::check_positions (
                integer out_id,
                type(particles) atoms,
                type(simulation_cell) box )
```

См. определение в файле md_general.f90 строка 327

```
327     type(particles)::   atoms
328     type(simulation_cell):: box
329     integer:: out_id,i,k,p
330     real:: tolerance=0.0000001
331
332     !$OMP PARALLEL private(i,k,p)
333     p=0
334     !$OMP DO
335     do i=1,atoms%N
336         do k=1,3
337             if( .not.(atoms%positions(k,i)>(0.-tolerance) .and. atoms%positions(k,i)<(box%box_size(k)+
    tolerance)) ) then
338                 write(out_id,*) i,' particle out of cell ',atoms%positions(:,i)
339                 p = p+1
340             endif
341         enddo
342     enddo
343     !$OMP END DO
344     !$OMP BARRIER
345     if(p>0) stop
346     !$OMP END PARALLEL
347
```

5.10.1.9 check_velocities()

```
subroutine md_general::check_velocities (
                integer out_id,
                type(particles) atoms )
```

См. определение в файле md_general.f90 строка 351

```
351     type(particles)::   atoms
352     integer:: out_id,i
353     real:: maxvel2,v2
354
355     maxvel2 = -1.
356     !$OMP PARALLEL DO private(i,v2) REDUCTION(max:maxvel2)
357     do i=1,atoms%N
358         v2 = sum(atoms%velocities(:,i)**2)
359         if(maxvel2<v2) maxvel2 = v2
360     enddo
361     !$OMP END PARALLEL DO
362     write(out_id,'(A,f16.8,A)') ' max velocity: ',sqrt(maxvel2),' A/fs '
363
```

5.10.1.10 create_particle_group()

```
subroutine md_general::create_particle_group (
                type(particle_group) group,
                character(len=32), dimension(:) type_names,
                type(particles) atoms )
```

См. определение в файле md_general.f90 строка 58

```
58     type(particles)::   atoms
59     type(particle_group):: group
60     character(len=32):: type_names(:)
61     integer:: i,j,ind
62
63     group%N = 0
64     do j=1,size(type_names)
65         group%N = group%N+count(atoms%atom_types==type_names(j))
66     enddo
```

```
67     allocate(group%indexes(group%N))
68
69     ind=0
70     do j=1,size(type_names)
71        do i=1,atoms%N
72           if (atoms%atom_types(i)==type_names(j)) then
73              ind=ind+1
74              group%indexes(ind) = i
75           endif
76        enddo
77     enddo
78
79     return
```

5.10.1.11 find_distance()

```
pure subroutine md_general::find_distance (
                real, dimension(3), intent(out) dr,
                real, intent(out) dr2,
                real, dimension(3), intent(in) vec1,
                real, dimension(3), intent(in) vec2,
                type(simulation_cell), intent(in) box )
```

См. определение в файле md_general.f90 строка 408

```
408     type(simulation_cell), intent (in) :: box
409     real, intent (in) :: vec1(3),vec2(3)
410     real, intent (out) :: dr(3),dr2
411
412     dr(1) = vec2(1)-vec1(1)
413     dr(2) = vec2(2)-vec1(2)
414     dr(3) = vec2(3)-vec1(3)
415     dr(1) = dr(1)-box%half_box_size(1)*(sign(1.,dr(1)-box%half_box_size(1))+sign(1.,dr(1)+box%half_box_size
    (1)))
416     dr(2) = dr(2)-box%half_box_size(2)*(sign(1.,dr(2)-box%half_box_size(2))+sign(1.,dr(2)+box%half_box_size
    (2)))
417     dr(3) = dr(3)-box%half_box_size(3)*(sign(1.,dr(3)-box%half_box_size(3))+sign(1.,dr(3)+box%half_box_size
    (3)))
418     dr2 = dr(1)*dr(1)+dr(2)*dr(2)+dr(3)*dr(3)
419
420     return
```

5.10.1.12 init_time_steps()

```
subroutine md_general::init_time_steps (
                type(time_steps) dt,
                real delta_t )
```

См. определение в файле md_general.f90 строка 46

```
46     type(time_steps):: dt
47     integer:: i
48     real:: delta_t
49
50     do i=1,size(dt%ts)
51        dt%ts(i) = delta_t/2**(i-1)
52     enddo
53
54     return
```

5.10.1.13 invert_z_velocities()

```
subroutine md_general::invert_z_velocities (
                type(particles) atoms,
                real z_low_border,
                real z_high_border )
```

См. определение в файле md_general.f90 строка 367

```
367     type(particles)::   atoms
368     integer:: i
369     real:: z_low_border,z_high_border
370
371     !$OMP PARALLEL DO private(i)
372     do i=1,atoms%N
373        if( (atoms%positions(3,i)>z_low_border .and.&
374           atoms%positions(3,i)<(z_low_border+z_high_border)/2 .and.&
375           atoms%velocities(3,i)>0.) .or.&
376           (atoms%positions(3,i)<z_high_border .and.&
377           atoms%positions(3,i)>(z_low_border+z_high_border)/2 .and.&
378           atoms%velocities(3,i)<0.)   )   atoms%velocities(3,i) = -atoms%velocities(3,i)
379     enddo
380     !$OMP END PARALLEL DO
381
```

5.10.1.14 position_analysis()

```
subroutine md_general::position_analysis (
                real av,
                real mi,
                real ma,
                type(particles) atoms,
                type(particle_group) group,
                integer direction,
                real minimum,
                real maximum )
```

См. определение в файле md_general.f90 строка 385

```
385     type(particles)::   atoms
386     type(particle_group):: group
387     real:: minimum,maximum,av,mi,ma
388     integer::       i,ind,direction,k
389
390     k = 0
391     av = 0.
392     mi = maximum
393     ma = minimum
394     do ind=1,group%N
395        i = group%indexes(ind)
396        if (atoms%positions(direction,i)<maximum .and. atoms%positions(direction,i)>minimum) then
397           av = av+atoms%positions(direction,i)
398           k = k+1
399           if (atoms%positions(direction,i)>ma) ma = atoms%positions(direction,i)
400           if (atoms%positions(direction,i)<mi) mi = atoms%positions(direction,i)
401        endif
402     enddo
403     av = av/k
404
```

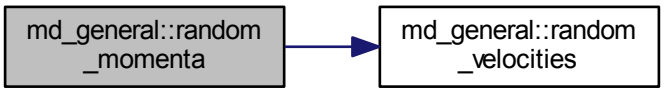
5.10.1.15 random_momenta()

```
subroutine md_general::random_momenta (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 143

```
143     type(particles)::   atoms
144     type(particle_group):: group
145     real,parameter:: coef = 1.3806488/1.6605389217*10.**(-6)
146     integer::       i,ind
147
148     call random_velocities(atoms,group)
149
150     !$OMP PARALLEL firstprivate(i,ind)
151     !$OMP DO
152     do ind=1,group%N
153         i = group%indexes(ind)
154         atoms%velocities(:,i) = atoms%velocities(:,i)*sqrt(coef/atoms%masses(i))
155     enddo
156     !$OMP END DO
157     !$OMP END PARALLEL
158
159     return
```

Граф вызовов:



5.10.1.16 random_velocities()

```
subroutine md_general::random_velocities (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 115

```
115     type(particles)::   atoms
116     type(particle_group):: group
117     real a1,a2,b
118     integer::       i,k,ind
119     integer omp_get_thread_num
120
121     !$OMP PARALLEL firstprivate(i,ind,k,a1,a2,b)
122     call srand(omp_get_thread_num())
123     !$OMP DO
124     do ind=1,group%N
125         i = group%indexes(ind)
126         do k=1,3,1
127             b = 2.
128             do while (b>=1.)
129                 a1 = 2.*rand()-1.
```

```
130             a2 = 2.*rand()-1.
131             b = a1**2+a2**2
132           enddo
133           atoms%velocities(k,i) = a1*sqrt(-2.*log(b)/b)
134         enddo
135       enddo
136       !$OMP END DO
137       !$OMP END PARALLEL
138
139       return
```

5.10.1.17 scale_velocities()

```
subroutine md_general::scale_velocities (
                type(particles) atoms,
                type(particle_group) group,
                real s )
```

См. определение в файле md_general.f90 строка 97

```
97    type(particles)::   atoms
98    type(particle_group):: group
99    real:: s
100    integer::       i,ind
101
102    !$OMP PARALLEL firstprivate(i,ind)
103    !$OMP DO
104    do ind=1,group%N
105       i = group%indexes(ind)
106       atoms%velocities(:,i) = atoms%velocities(:,i)*s
107    enddo
108    !$OMP END DO
109    !$OMP END PARALLEL
110
111    return
```

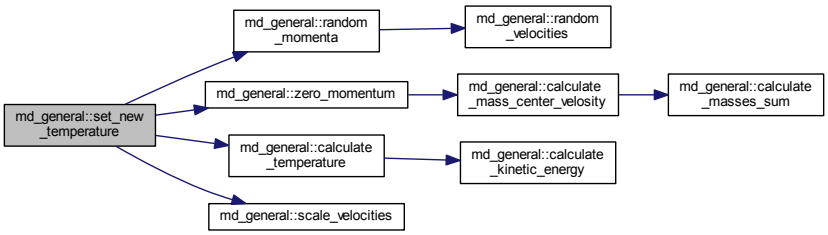
5.10.1.18 set_new_temperature()

```
subroutine md_general::set_new_temperature (
                type(particles) atoms,
                type(particle_group) group,
                real temp )
```

См. определение в файле md_general.f90 строка 315

```
315    type(particles)::   atoms
316    type(particle_group):: group
317    real:: temp,ke,temperature
318
319    call random_momenta(atoms,group)
320    call zero_momentum(atoms,group)
321    call calculate_temperature(temperature,ke,atoms,group)
322    call scale_velocities(atoms,group,sqrt(temp/temperature))
323
```

Граф вызовов:



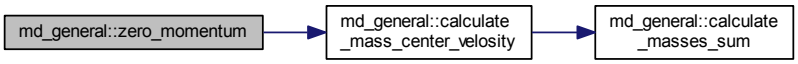
5.10.1.19 zero_momentum()

```
subroutine md_general::zero_momentum (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_general.f90 строка 238

```
238     type(particles)::   atoms
239     type(particle_group):: group
240     integer::       i,ind
241     real:: mcv(3)
242
243     call calculate_mass_center_velosity(mcv,atoms,group)
244     !$OMP PARALLEL firstprivate(mcv,i,ind)
245     !$OMP DO
246     do ind=1,group%N
247        i = group%indexes(ind)
248        atoms%velocities(:,i) = atoms%velocities(:,i)-mcv
249     enddo
250     !$OMP END DO
251     !$OMP END PARALLEL
252
253     return
```

Граф вызовов:



5.11 Модуль md_integrators

Функции/подпрограммы

- subroutine integrate_verlet_xyz_positions (atoms, group, steps, box)

- subroutine integrate_verlet_z_positions (atoms, group, steps, box)
- subroutine integrate_verlet_xyz_velocities (atoms, group, steps)
- subroutine integrate_verlet_z_velocities (atoms, group, steps)
- subroutine molecular_static_xyz_velocities (atoms, group)
- subroutine molecular_static_1d_velocities (atoms, group)
- subroutine zero_forces (atoms, group)
- subroutine create_nose_hoover_chain (nhc)
- subroutine set_nose_hoover_chain (nhc, temp, q1, l)
- subroutine integrate_nose_hoover_chain (nhc, atoms, group, dt)
- subroutine calculate_nose_hoover_chain_energy (nhc)

5.11.1 Функции/подпрограммы

5.11.1.1 calculate_nose_hoover_chain_energy()

```
subroutine md_integrators::calculate_nose_hoover_chain_energy (
                type(nose_hoover_chain) nhc )
```

См. определение в файле md_integrators.f90 строка 242

```
242     type(nose_hoover_chain):: nhc
243     real:: kt
244     integer::       i
245
246     kt=1.3806488/1.6021765654*10.**(-4)*nhc%temperature
247
248     nhc%e = nhc%q(1)/2*nhc%v(1)**2+3.*nhc%L*kt*nhc%x(1)
249     do i=2,nhc%M,1
250        nhc%e = nhc%e+nhc%q(i)/2*nhc%v(i)**2+kt*nhc%x(i)
251     enddo
252
253     return
```

5.11.1.2 create_nose_hoover_chain()

```
subroutine md_integrators::create_nose_hoover_chain (
                type(nose_hoover_chain) nhc )
```

См. определение в файле md_integrators.f90 строка 167

```
167     type(nose_hoover_chain):: nhc
168
169     allocate(nhc%x(nhc%M))
170     allocate(nhc%v(nhc%M))
171     allocate(nhc%q(nhc%M))
172     nhc%x = 0.
173     nhc%v = 0.
174     nhc%q = 0.
175     nhc%e = 0.
176     nhc%s = 1.
177
178     return
```

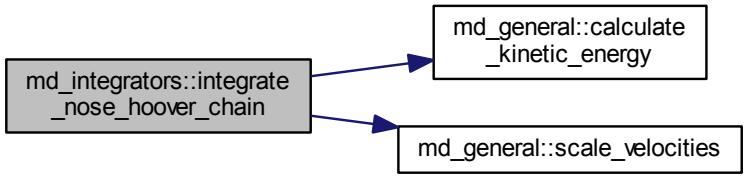
5.11.1.3 integrate_nose_hoover_chain()

```
subroutine md_integrators::integrate_nose_hoover_chain (
                type(nose_hoover_chain) nhc,
                type(particles) atoms,
                type(particle_group) group,
                type(time_steps) dt )
```

См. определение в файле md_integrators.f90 строка 197

```
197     type(particles)::   atoms
198     type(particle_group):: group
199     type(time_steps):: dt
200     type(nose_hoover_chain):: nhc
201     real:: kedif,ke,kt,b
202     integer::       i
203
204     call calculate_kinetic_energy(ke,atoms,group)
205     kt=1.3806488/1.6021765654*10.**(-4)*nhc%temperature
206     kedif = 2.*ke-3.*nhc%L*kt
207
208     if (nhc%M==1) then
209         nhc%v(1)=nhc%v(1)+kedif/nhc%q(1)*dt%ts(3)
210     else
211         nhc%v(nhc%M)=nhc%v(nhc%M)+(nhc%q(nhc%M-1)*nhc%v(nhc%M-1)**2-kt)/nhc%q(nhc%M)*dt%ts(3)
212         do i=nhc%M-1,2,-1
213             b=exp(-nhc%v(i+1)*dt%ts(4))
214             nhc%v(i)=nhc%v(i)*b**2+(nhc%q(i-1)*nhc%v(i-1)**2-kt)/nhc%q(i)*dt%ts(3)*b
215         enddo
216         b=exp(-nhc%v(2)*dt%ts(4))
217         nhc%v(1)=nhc%v(1)*b**2+kedif/nhc%q(1)*dt%ts(3)*b
218     endif
219
220     nhc%s=exp(-nhc%v(1)*dt%ts(2))
221     call scale_velocities(atoms,group,nhc%s)
222     kedif = 2.*ke*nhc%s**2-3.*nhc%L*kt
223     do i=1,nhc%M,1
224         nhc%x(i)=nhc%x(i)+nhc%v(i)*dt%ts(2)
225     enddo
226
227     if (nhc%M==1) then
228         nhc%v(1)=nhc%v(1)+kedif/nhc%q(1)*dt%ts(3)
229     else
230         nhc%v(1)=nhc%v(1)*b**2+kedif/nhc%q(1)*dt%ts(3)*b
231         do i=2,nhc%M-1,1
232             b=exp(-nhc%v(i+1)*dt%ts(4))
233             nhc%v(i)=nhc%v(i)*b**2+(nhc%q(i-1)*nhc%v(i-1)**2-kt)/nhc%q(i)*dt%ts(3)*b
234         enddo
235         nhc%v(nhc%M)=nhc%v(nhc%M)+(nhc%q(nhc%M-1)*nhc%v(nhc%M-1)**2-kt)/nhc%q(nhc%M)*dt%ts(3)
236     endif
237
238     return
```

Граф вызовов:



5.11.1.4 integrate_verlet_xyz_positions()

```
subroutine md_integrators::integrate_verlet_xyz_positions (
                type(particles) atoms,
                type(particle_group) group,
                type(time_steps) steps,
                type(simulation_cell) box )
```

См. определение в файле md_integrators.f90 строка 8

```
8     type(particles)::   atoms
9     type(particle_group):: group
10    type(time_steps):: steps
11    type(simulation_cell):: box
12    integer::       i,k,ind
13
14    !$OMP PARALLEL firstprivate(i,ind,k)
15    !$OMP DO
16    do ind=1,group%N
17       i = group%indexes(ind)
18       do k=1,3
19          atoms%positions(k,i) = atoms%positions(k,i)+atoms%velocities(k,i)*steps%ts(1)
20          if (atoms%positions(k,i)>box%box_size(k)) then
21             atoms%positions(k,i) = atoms%positions(k,i)-box%box_size(k)
22          elseif (atoms%positions(k,i)<0.) then
23             atoms%positions(k,i) = atoms%positions(k,i)+box%box_size(k)
24          endif
25       enddo
26    enddo
27    !$OMP END DO
28    !$OMP END PARALLEL
29
30    return
```

5.11.1.5 integrate_verlet_xyz_velocities()

```
subroutine md_integrators::integrate_verlet_xyz_velocities (
                type(particles) atoms,
                type(particle_group) group,
                type(time_steps) steps )
```

См. определение в файле md_integrators.f90 строка 59

```
59    type(particles)::   atoms
60    type(particle_group):: group
61    type(time_steps):: steps
62    real,parameter:: mass_coef=1.6605389217/1.6021765654*10.**(2)
63    integer::       i,ind,k
64
65    !$OMP PARALLEL firstprivate(i,ind,k)
66    !$OMP DO
67    do ind=1,group%N
68       i = group%indexes(ind)
69       do k=1,3
70          atoms%velocities(k,i) = atoms%velocities(k,i)+atoms%forces(k,i)/atoms%masses(i)/mass_coef*steps
   %ts(2)
71       enddo
72    enddo
73    !$OMP END DO
74    !$OMP END PARALLEL
75
76    return
```

5.11.1.6 integrate_verlet_z_positions()

```
subroutine md_integrators::integrate_verlet_z_positions (
                type(particles) atoms,
                type(particle_group) group,
                type(time_steps) steps,
                type(simulation_cell) box )
```

См. определение в файле md_integrators.f90 строка 34

```
34    type(particles)::   atoms
35    type(particle_group):: group
36    type(time_steps):: steps
37    type(simulation_cell):: box
38    integer::       i,k,ind
39
40    !$OMP PARALLEL firstprivate(i,ind,k)
41    !$OMP DO
42    do ind=1,group%N
43       i = group%indexes(ind)
44       k = 3
45          atoms%positions(k,i) = atoms%positions(k,i)+atoms%velocities(k,i)*steps%ts(1)
46          if (atoms%positions(k,i)>box%box_size(k)) then
47             atoms%positions(k,i) = atoms%positions(k,i)-box%box_size(k)
48          elseif (atoms%positions(k,i)<0.) then
49             atoms%positions(k,i) = atoms%positions(k,i)+box%box_size(k)
50          endif
51    enddo
52    !$OMP END DO
53    !$OMP END PARALLEL
54
55    return
```

5.11.1.7 integrate_verlet_z_velocities()

```
subroutine md_integrators::integrate_verlet_z_velocities (
                type(particles) atoms,
                type(particle_group) group,
                type(time_steps) steps )
```

См. определение в файле md_integrators.f90 строка 80

```
80    type(particles)::   atoms
81    type(particle_group):: group
82    type(time_steps):: steps
83    real,parameter:: mass_coef=1.6605389217/1.6021765654*10.**(2)
84    integer::       i,ind,k
85
86    !$OMP PARALLEL firstprivate(i,ind,k)
87    !$OMP DO
88    do ind=1,group%N
89       i = group%indexes(ind)
90       k = 3
91          atoms%velocities(k,i) = atoms%velocities(k,i)+atoms%forces(k,i)/atoms%masses(i)/mass_coef*steps
    %ts(2)
92    enddo
93    !$OMP END DO
94    !$OMP END PARALLEL
95
96    return
```

5.11.1.8 molecular_static_1d_velocities()

```
subroutine md_integrators::molecular_static_1d_velocities (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_integrators.f90 строка 126

```
126     type(particles)::   atoms
127     type(particle_group):: group
128     real:: fv
129     integer:: i,k,ind
130
131     !$OMP PARALLEL firstprivate(i,k,ind,fv)
132     !$OMP DO
133     do ind=1,group%N
134         i = group%indexes(ind)
135         fv = sum(atoms%forces(:,i)*atoms%velocities(:,i))
136         if (fv>0.) then
137         else
138             atoms%velocities(:,i) = 0.
139         endif
140     enddo
141     !$OMP END DO
142     !$OMP END PARALLEL
143
144     return
```

5.11.1.9 molecular_static_xyz_velocities()

```
subroutine md_integrators::molecular_static_xyz_velocities (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_integrators.f90 строка 100

```
100     type(particles)::   atoms
101     type(particle_group):: group
102     real:: fv,ff
103     integer:: i,k,ind
104
105     !$OMP PARALLEL firstprivate(i,k,ind,fv,ff)
106     !$OMP DO
107     do ind=1,group%N
108         i = group%indexes(ind)
109         fv = sum(atoms%forces(:,i)*atoms%velocities(:,i))
110         ff = sum(atoms%forces(:,i)**2)
111         if (fv>0. .and. ff>10.**(-12) ) then
112             do k=1,3
113                 atoms%velocities(k,i) = fv/ff*atoms%forces(k,i)
114             enddo
115         else
116             atoms%velocities(:,i) = 0.
117         endif
118     enddo
119     !$OMP END DO
120     !$OMP END PARALLEL
121
122     return
```

5.11.1.10 set_nose_hoover_chain()

```
subroutine md_integrators::set_nose_hoover_chain (
                type(nose_hoover_chain) nhc,
                real temp,
                real q1,
                integer l )
```

См. определение в файле md_integrators.f90 строка 182

```
182     type(nose_hoover_chain):: nhc
183     integer:: l,i
184     real::  q1,temp
185
186     nhc%L = l
187     nhc%temperature = temp
188     nhc%q(1) = q1
189     do i=2,nhc%M,1
190         nhc%q(i) = nhc%q(1)/(3.d0*nhc%L)
191     enddo
192
193     return
```

5.11.1.11 zero_forces()

```
subroutine md_integrators::zero_forces (
                type(particles) atoms,
                type(particle_group) group )
```

См. определение в файле md_integrators.f90 строка 148

```
148     type(particles)::   atoms
149     type(particle_group):: group
150     integer:: i,ind,k
151
152     !$OMP PARALLEL firstprivate(i,ind,k)
153     !$OMP DO
154     do ind=1,group%N
155         i = group%indexes(ind)
156         do k=1,3
157             atoms%forces(k,i) = 0.
158         enddo
159     enddo
160     !$OMP END DO
161     !$OMP END PARALLEL
162
```

5.12 Модуль md_interactions

Типы данных

- type interaction
- type interaction_parameters

Функции/подпрограммы

- subroutine create_groups (groups, file_id, out_id, atoms)
- subroutine create_interactions (interactions, groups, file_id, out_id, input_path)
- subroutine update_interactions_neighbour_lists (md_step, interactions, atoms, groups, cell, exe↩ _time_nlsearch, exe_time_nldistance)
- subroutine allocate_graphene_norm (interactions)
- subroutine update_norm_in_graphene (interactions)
- subroutine calculate_forces (atoms, interactions)
- subroutine energy (inter_name, e, nl, p)
- subroutine calculate_potential_energies (interactions)
- subroutine calculate_forces_numerically (atoms, interactions)
- subroutine create_truncated_nl (tnl, nl)
- subroutine destroy_truncated_nl (tnl)
- subroutine calculate_truncated_nl (tnl, nl, i, n)
- subroutine shift_drs (tnl, inl, k, nl_n, dx)
- subroutine shift_gr_norm (gr_norm, nl_nn, inl, k, dx)
- subroutine nlists_load (out_id, interactions)

5.12.1 Функции/подпрограммы

5.12.1.1 allocate_graphene_norm()

```
subroutine md_interactions::allocate_graphene_norm (
                type(interaction), dimension(:) interactions )
```

См. определение в файле md_interactions.f90 строка 181

```
181     type(interaction):: interactions(:)
182     integer:: i
183
184     do i=1,size(interactions)
185         select case (interactions(i)%interaction_name)
186         case('ljc')
187             allocate(interactions(i)%parameters%LJC(1)%gr_norm(3,interactions(i)%nl(3)%N))
188         case('morsec')
189             allocate(interactions(i)%parameters%MorseC(1)%gr_norm(3,interactions(i)%nl(3)%N))
190         end select
191     enddo
192
```

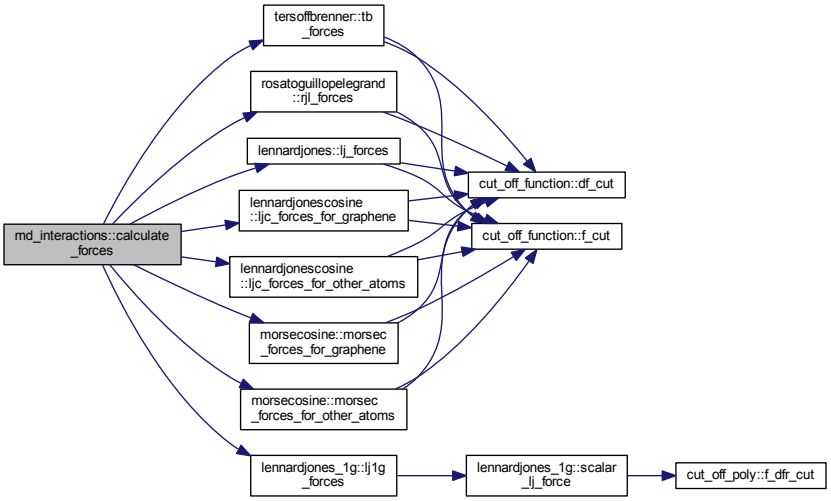
5.12.1.2 calculate_forces()

subroutine md_interactions::calculate_forces (
type(particles) atoms,
type(interaction), dimension(:) interactions)

См. определение в файле md_interactions.f90 строка 211

```
211     type(interaction):: interactions(:)
212     type(particles):: atoms
213     integer:: i
214
215     do i=1,size(interactions)
216         if(.not. interactions(i)%numerical_force) then
217             select case (interactions(i)%interaction_name)
218             case('lj')
219                 call lj_forces(atoms,interactions(i)%nl(1),interactions(i)%parameters%LJ(1))
220                 call lj_forces(atoms,interactions(i)%nl(2),interactions(i)%parameters%LJ(1))
221             case('lj1g')
222                 call lj1g_forces(atoms,interactions(i)%nl(1),interactions(i)%parameters%LJ1g(1))
223             !case('morse')
224             !   call Morse_forces(atoms,interactions(i)%nl(1),interactions(i)%parameters%Morse(1))
225             !   call Morse_forces(atoms,interactions(i)%nl(2),interactions(i)%parameters%Morse(1))
226             case('ljc')
227                 call ljc_forces_for_graphene(atoms,interactions(i)%nl(1),interactions(i)%nl(3),interactions
    (i)%parameters%LJC(1))
228                 call ljc_forces_for_other_atoms(atoms,interactions(i)%nl(2),interactions(i)%parameters%LJC(
    1))
229             case('morsec')
230                 call morsec_forces_for_graphene(atoms,interactions(i)%nl(1),interactions(i)%nl(3),
    interactions(i)%parameters%MorseC(1))
231                 call morsec_forces_for_other_atoms(atoms,interactions(i)%nl(2),interactions(i)%parameters
    %MorseC(1))
232             case('tb')
233                 call tb_forces(atoms,interactions(i)%nl(1),interactions(i)%parameters%TB(1))
234             case('rebosc')
235             !
236             case('rjl')
237                 call rjl_forces(atoms,interactions(i)%nl(1),interactions(i)%parameters%RJL(1))
238             end select
239         endif
240     enddo
241
```

Граф вызовов:



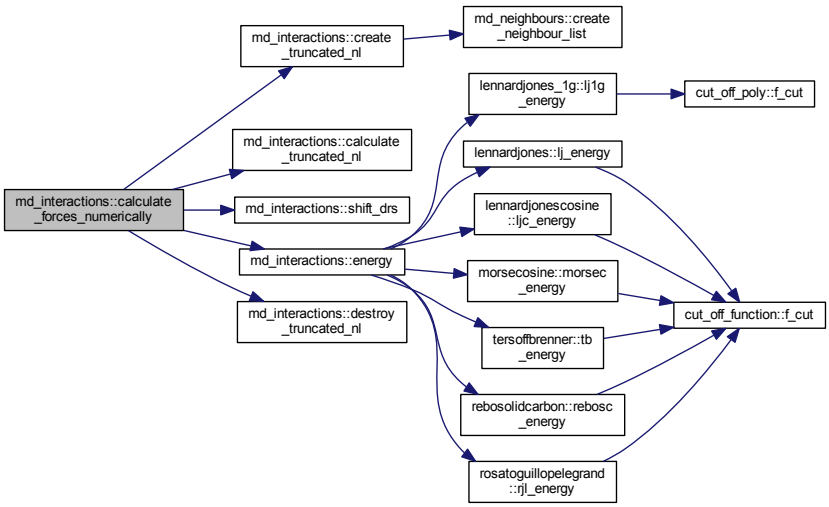
5.12.1.3 calculate_forces_numerically()

```
subroutine md_interactions::calculate_forces_numerically (
                type(particles) atoms,
                type(interaction), dimension(:) interactions )
```

См. определение в файле md_interactions.f90 строка 274

```
274     type(interaction):: interactions(:)
275     type(particles):: atoms
276     type(neighbour_list):: tnl
277     real:: e1,e2
278     integer:: i,k,inl,j
279     real,parameter:: dx = 10.**(-6)
280
281     do i=1,size(interactions)
282         if(interactions(i)%numerical_force) then
283             select case (interactions(i)%interaction_name)
284             case('ljc','morsec')!nl(1) - calc_tnl(0),add nearest neibls,shift_nl,shift_gr_norm !nl(2) -
    calc_tnl(0),shift_nl
285             case('lj','morse','tb','rebosc','rjl')
286                 do j=1,interactions(i)%nl_n
287                 !$OMP PARALLEL firstprivate(k,inl,e1,e2) private(tnl)
288                 call create_truncated_nl(tnl,interactions(i)%nl(j))
289                 !$OMP DO
290                 do inl=1,interactions(i)%nl(j)%N
291                     call calculate_truncated_nl(tnl,interactions(i)%nl(j),inl,interactions(i)
    %neib_order)
292                     do k=1,3
293                         call shift_drs(tnl,inl,k,interactions(i)%nl_n,-dx)
294                         call energy(interactions(i)%interaction_name,e1,tnl,interactions(i)%parameters)
295                         call shift_drs(tnl,inl,k,interactions(i)%nl_n,2*dx)
296                         call energy(interactions(i)%interaction_name,e2,tnl,interactions(i)%parameters)
297                         if(k/=3) call shift_drs(tnl,inl,k,interactions(i)%nl_n,-dx)
298                         atoms%forces(k,interactions(i)%nl(j)%particle_index(inl)) =&
299                         atoms%forces(k,interactions(i)%nl(j)%particle_index(inl))+(e1-e2)/2/dx
300                     enddo
301                 enddo
302                 !$OMP END DO
303                 call destroy_truncated_nl(tnl)
304                 !$OMP END PARALLEL
305             enddo
306         end select
307       endif
308     enddo
309
```

Граф вызовов:



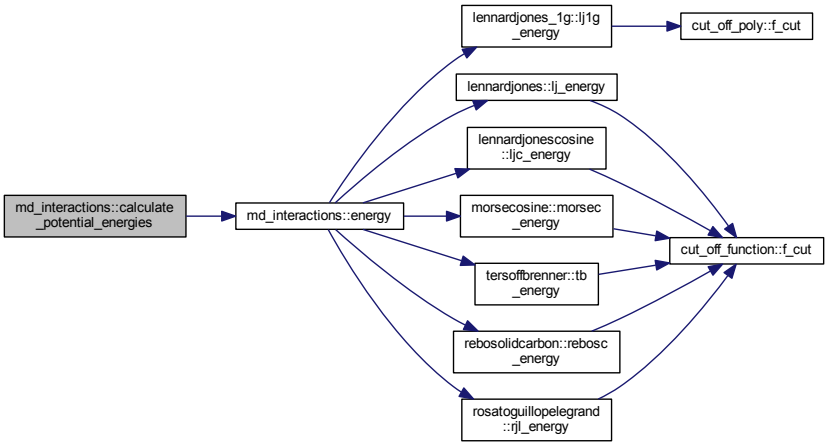
5.12.1.4 calculate_potential_energies()

subroutine md_interactions::calculate_potential_energies (
type(interaction), dimension(:) interactions)

См. определение в файле md_interactions.f90 строка 264

```
264     type(interaction):: interactions(:)
265     integer:: i
266
267     do i=1,size(interactions)
268         call energy(interactions(i)%interaction_name,interactions(i)%energy,interactions(i)%nl(1),
    interactions(i)%parameters)
269     enddo
270
```

Граф вызовов:



5.12.1.5 calculate_truncated_nl()

subroutine md_interactions::calculate_truncated_nl (
type(neighbour_list) tnl,
type(neighbour_list) nl,
integer i,
integer n)

См. определение в файле md_interactions.f90 строка 335

```
335     type(neighbour_list):: tnl,nl
336     integer:: n,i,ni,p,q,j
337
338     tnl%particle_index = 0; tnl%nnum = 0; !tnl%nlist = 0!tnl%dr = 0.!tnl%moddr = 0.
339
340     tnl%N = nl%N
341     tnl%neighb_num_max = nl%neighb_num_max
342     tnl%nlist(:nl%nnum(i),i) = nl%nlist(:nl%nnum(i),i)
343     tnl%dr(:,:nl%nnum(i),i) = nl%dr(:,:nl%nnum(i),i)
344     tnl%moddr(:nl%nnum(i),i) = nl%moddr(:nl%nnum(i),i)
```

```
345     tnl%nnum(i) = nl%nnum(i)
346     tnl%particle_index(i) = nl%particle_index(i)
347     do ni=1,n
348        if(ni<n) then
349           do j=1,tnl%N
350              do p=1,tnl%nnum(j)
351                 q = tnl%nlist(p,j)
352                 if(tnl%nnum(q)==0) then
353                    tnl%nlist(:nl%nnum(q),q) = nl%nlist(:nl%nnum(q),q)
354                    tnl%dr(:,:nl%nnum(q),q) = nl%dr(:,:nl%nnum(q),q)
355                    tnl%moddr(:nl%nnum(q),q) = nl%moddr(:nl%nnum(q),q)
356                    tnl%particle_index(q) = nl%particle_index(q)
357                 endif
358              enddo
359           enddo
360           do j=1,tnl%N
361              if(tnl%particle_index(j)/=0) tnl%nnum(j) = nl%nnum(j)
362           enddo
363        else
364           do j=1,tnl%N
365              do p=1,tnl%nnum(j)
366                 q = tnl%nlist(p,j)
367                 if(tnl%nnum(q)==0) then
368                    tnl%nnum(q) = tnl%nnum(q)+1
369                    tnl%nlist(tnl%nnum(q),q) = j
370                    tnl%dr(:,tnl%nnum(q),q) = -nl%dr(:,p,j)
371                    tnl%moddr(tnl%nnum(q),q) = nl%moddr(p,j)
372                    tnl%particle_index(q) = nl%particle_index(q)
373                 endif
374              enddo
375           enddo
376        endif
377     enddo
378
```

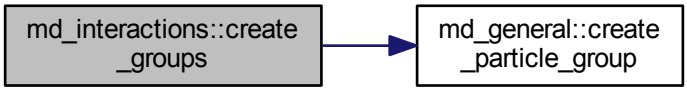
5.12.1.6 create_groups()

```
subroutine md_interactions::create_groups (
                type(particle_group), dimension(:), allocatable groups,
                integer file_id,
                integer out_id,
                type(particles) atoms )
```

См. определение в файле md_interactions.f90 строка 39

```
39     type(particle_group),allocatable:: groups(:)
40     type(particles):: atoms
41     character(len=32),allocatable:: type_names(:,:)
42     character(len=128):: str,frmt
43     integer:: i,particle_types_num,groups_num,file_id,out_id
44
45     read(file_id,*) str,particle_types_num
46     write(out_id,'(A32,i12)') str,particle_types_num
47     read(file_id,*) str,groups_num
48     write(out_id,'(A32,i12)') str,groups_num
49     allocate(groups(groups_num),type_names(particle_types_num,groups_num))
50     write(frmt,'("(i6,A,",i0,"A12,i9)")') particle_types_num
51     do i=1,groups_num
52        read(file_id,*) str,type_names(:,i)
53        call create_particle_group(groups(i),type_names(:,i),atoms)
54        write(out_id,frmt) i,' ',type_names(:,i),groups(i)%N
55     enddo
56
```

Граф вызовов:



5.12.1.7 create_interactions()

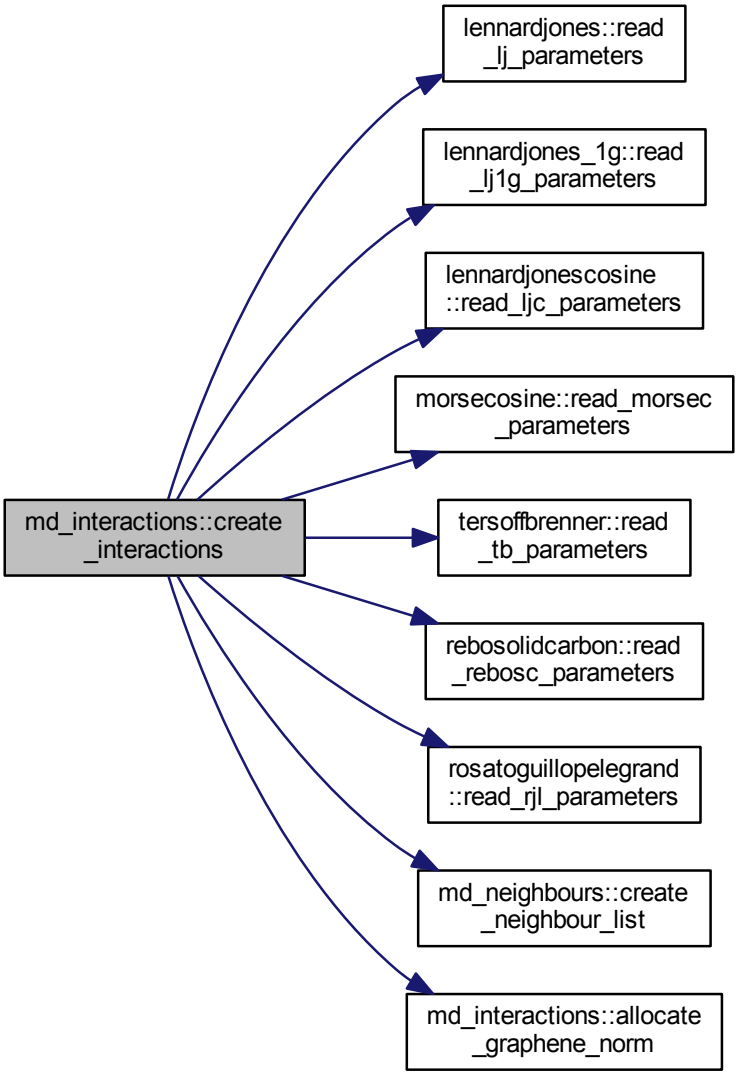
```
subroutine md_interactions::create_interactions (
                type(interaction), dimension(:), allocatable interactions,
                type(particle_group), dimension(:) groups,
                integer file_id,
                integer out_id,
                character(len=128) input_path )
```

См. определение в файле md_interactions.f90 строка 60

```
60    type(interaction),allocatable:: interactions(:)
61    type(particle_group):: groups(:)
62    integer:: file_id,out_id,i,j,inter_n
63    character(len=128):: input_path,str
64
65    read(file_id,*) str,inter_n
66    write(out_id,'(A32,i12)') str,inter_n
67    allocate(interactions(inter_n))
68    do i=1,inter_n
69        read(file_id,*) interactions(i)%interaction_name,interactions(i)%parameters_file
70        select case (interactions(i)%interaction_name)
71        case('lj')
72            allocate(interactions(i)%parameters%LJ(1))
73            call read_lj_parameters(interactions(i)%parameters%LJ(1),trim(input_path)//interactions(i)
    %parameters_file)
74            interactions(i)%nl_n = 2
75            interactions(i)%neib_order = 0
76            interactions(i)%numerical_force = .false.
77        case('lj1g')
78            allocate(interactions(i)%parameters%LJ1g(1))
79            call read_lj1g_parameters(interactions(i)%parameters%LJ1g(1),trim(input_path)//interactions(i)
    %parameters_file)
80            interactions(i)%nl_n = 1
81            interactions(i)%neib_order = 0
82            interactions(i)%numerical_force = .false.
83        !case('morse')
84        !   allocate(interactions(i)%parameters%Morse(1))
85        !   call
     read_Morse_parameters(interactions(i)%parameters%Morse(1),trim(input_path)//interactions(i)%parameters_file)
86        !   interactions(i)%nl_n = 2
87        !   interactions(i)%neib_order = 0
88        !   interactions(i)%numerical_force = .false.
89        case('ljc')
90            allocate(interactions(i)%parameters%LJC(1))
91            call read_ljc_parameters(interactions(i)%parameters%LJC(1),trim(input_path)//interactions(i)
    %parameters_file)
92            interactions(i)%nl_n = 3
93            interactions(i)%neib_order = 0
94            interactions(i)%numerical_force = .false.
95        case('morsec')
96            allocate(interactions(i)%parameters%MorseC(1))
97            call read_morsec_parameters(interactions(i)%parameters%MorseC(1),trim(input_path)//interactions
    (i)%parameters_file)
```

```
98            interactions(i)%nl_n = 3
99            interactions(i)%neib_order = 0
100             interactions(i)%numerical_force = .false.
101         case('tb')
102             allocate(interactions(i)%parameters%TB(1))
103             call read_tb_parameters(interactions(i)%parameters%TB(1),trim(input_path)//interactions(i)
    %parameters_file)
104             interactions(i)%nl_n = 1
105             interactions(i)%neib_order = 2
106             interactions(i)%numerical_force = .false.
107         case('rebosc')
108             allocate(interactions(i)%parameters%REBOsc(1))
109             call read_rebosc_parameters(interactions(i)%parameters%REBOsc(1),trim(input_path)//interactions
    (i)%parameters_file)
110             interactions(i)%nl_n = 1
111             interactions(i)%neib_order = 3
112             interactions(i)%numerical_force = .true.
113         case('rjl')
114             allocate(interactions(i)%parameters%RJL(1))
115             call read_rjl_parameters(interactions(i)%parameters%RJL(1),trim(input_path)//interactions(i)
    %parameters_file)
116             interactions(i)%nl_n = 1
117             interactions(i)%neib_order = 2
118             interactions(i)%numerical_force = .false.
119         case default
120             print*,'error: unknown interaction name'
121         end select
122         write(out_id,'(2A32,i6)') interactions(i)%interaction_name,interactions(i)%parameters_file,
    interactions(i)%nl_n
123         allocate(interactions(i)%group_nums(2*interactions(i)%nl_n))
124         allocate(interactions(i)%nl(interactions(i)%nl_n))
125         do j=1,interactions(i)%nl_n
126             read(file_id,*) interactions(i)%group_nums(j*2-1:j*2),&
127             interactions(i)%nl(j)%neighb_num_max,interactions(i)%nl(j)%r_cut,interactions(i)%nl(j)
    %update_period
128             write(out_id,'(3i6,f16.6,i9)') interactions(i)%group_nums(j*2-1:j*2),&
129             interactions(i)%nl(j)%neighb_num_max,interactions(i)%nl(j)%r_cut,interactions(i)%nl(j)
    %update_period
130             interactions(i)%nl(j)%N = groups(interactions(i)%group_nums(j*2-1))%N
131             call create_neighbour_list(interactions(i)%nl(j))
132         enddo
133     enddo
134     call allocate_graphene_norm(interactions)
135
```

Граф вызовов:



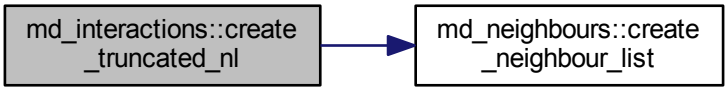
5.12.1.8 create_truncated_nl()

```
subroutine md_interactions::create_truncated_nl (
                type(neighbour_list) tnl,
                type(neighbour_list) nl )
```

См. определение в файле md_interactions.f90 строка 313

```
313     type(neighbour_list):: tnl,nl
314
315     tnl%N = nl%N
316     tnl%neighb_num_max = nl%neighb_num_max
317     call create_neighbour_list(tnl)
318
```

Граф вызовов:



5.12.1.9 destroy_truncated_nl()

```
subroutine md_interactions::destroy_truncated_nl (
                type(neighbour_list) tnl )
```

См. определение в файле md_interactions.f90 строка 322

```
322     type(neighbour_list):: tnl
323
324     tnl%N = 0
325     tnl%neighb_num_max = 0
326     if(allocated(tnl%particle_index)) deallocate(tnl%particle_index)
327     if(allocated(tnl%nnum)) deallocate(tnl%nnum)
328     if(allocated(tnl%nlist)) deallocate(tnl%nlist)
329     if(allocated(tnl%dr)) deallocate(tnl%dr)
330     if(allocated(tnl%moddr)) deallocate(tnl%moddr)
331
```

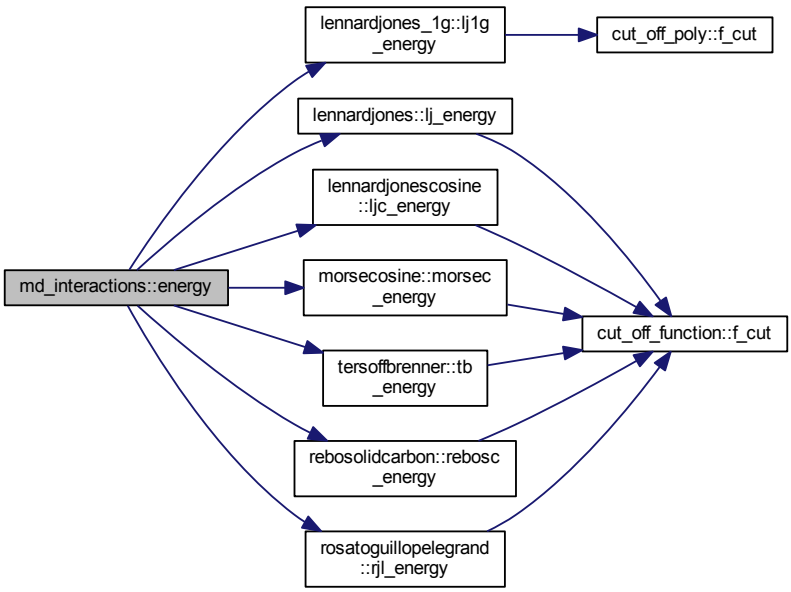
5.12.1.10 energy()

```
subroutine md_interactions::energy (
                character(len=32) inter_name,
                real e,
                type(neighbour_list) nl,
                type(interaction_parameters) p )
```

См. определение в файле md_interactions.f90 строка 245

```
245     type(interaction_parameters):: p
246     type(neighbour_list):: nl
247     character(len=32):: inter_name
248     real:: e
249
250     select case (inter_name)
251     case('lj');      call lj_energy(e,nl,p%LJ(1))
252     case('lj1g');    call lj1g_energy(e,nl,p%LJ1g(1))
253     !case('morse');
254     case('ljc');     call ljc_energy(e,nl,p%LJC(1))
255     case('morsec'); call morsec_energy(e,nl,p%MorseC(1))
256     case('tb');      call tb_energy(e,nl,p%TB(1))
257     case('rebosc'); call rebosc_energy(e,nl,p%REBOsc(1))
258     case('rjl');     call rjl_energy(e,nl,p%RJL(1))
259     end select
260
```

Граф вызовов:



5.12.1.11 nlists_load()

```
subroutine md_interactions::nlists_load (
                integer out_id,
                type(interaction), dimension(:) interactions )
```

См. определение в файле md_interactions.f90 строка 428

```
428     type(interaction):: interactions(:)
429     integer:: i,j,out_id
430
431     if(size(interactions)>0) write(out_id,'(A)') ' neib lists load:'
432     do i=1,size(interactions)
433        do j=1,interactions(i)%nl_n
434           if(interactions(i)%nl(j)%N>0) then
435              write(out_id,'(A12,i6,i6,A,i6)') trim(interactions(i)%interaction_name),j,&
436              maxval(interactions(i)%nl(j)%nnum),'/',interactions(i)%nl(j)%neighb_num_max
437           else
438              write(out_id,'(A12,i6,i6,A,i6)') trim(interactions(i)%interaction_name),j,&
439              0,'/',interactions(i)%nl(j)%neighb_num_max
440           endif
441        enddo
442     enddo
443
```

5.12.1.12 shift_drs()

```
subroutine md_interactions::shift_drs (
                type(neighbour_list) tnl,
                integer inl,
                integer k,
                integer nl_n,
                real dx )
```

См. определение в файле md_interactions.f90 строка 382

```
382     type(neighbour_list):: tnl
383     real:: dx
384     integer:: p,k,inl,q,j,nl_n
385
386     do p=1,tnl%nnum(inl)
387        tnl%dr(k,p,inl) = tnl%dr(k,p,inl)-dx
388        tnl%moddr(p,inl) = sqrt(sum(tnl%dr(:,p,inl)**2))
389     enddo
390     if(nl_n==1) then
391        do j=1,tnl%N
392           do p=1,tnl%nnum(j)
393              q = tnl%nlist(p,j)
394              if(q==inl) then
395                 tnl%dr(k,p,j) = tnl%dr(k,p,j)+dx
396                 tnl%moddr(p,j) = sqrt(sum(tnl%dr(:,p,j)**2))
397              endif
398           enddo
399        enddo
400     endif
401
```

5.12.1.13 shift_gr_norm()

```
subroutine md_interactions::shift_gr_norm (
                real, dimension(:,:) gr_norm,
                type(neighbour_list) nl_nn,
                integer inl,
                integer k,
                real dx )
```

См. определение в файле md_interactions.f90 строка 405

```
405     real:: gr_norm(:,:)
406     type(neighbour_list):: nl_nn
407     real:: dx,drj12(3),drj31(3)
408     integer:: k,inl,p,q,j,l
409
410     do p=1,nl_nn%nnum(inl)
411        j = nl_nn%nlist(p,inl)
412        do q=1,nl_nn%nnum(j)
413           if(nl_nn%nlist(q,j)==inl) exit
414        enddo
415        drj12 = nl_nn%dr(:,mod(q+1,3)+1,j)-nl_nn%dr(:,mod(q,3)+1,j)
416        drj31 = nl_nn%dr(:,mod(q,3)+1,j)-nl_nn%dr(:,q,j)
417        drj31(k) = drj31(k)+dx
418        do l=1,3
419           gr_norm(l,j) = (drj12(mod(l,3)+1))*(drj31(mod(l+1,3)+1))-(drj12(mod(l+1,3)+1))*(drj31(mod(l,3)+
    1))
420        enddo
421        if (gr_norm(3,j)<0.) gr_norm(:,j)=-gr_norm(:,j)
422        gr_norm(:,j) = gr_norm(:,j)/sqrt(sum(gr_norm(:,j)**2))
423     enddo
424
```

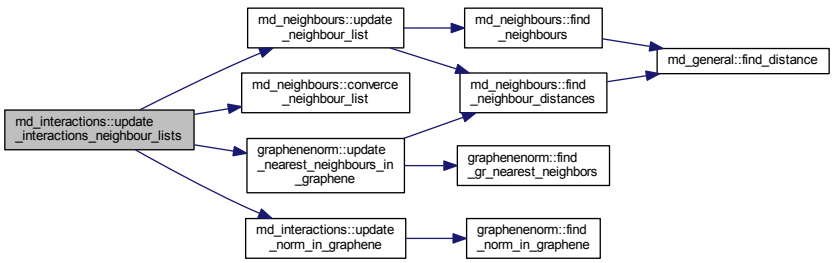
5.12.1.14 update_interactions_neighbour_lists()

```
subroutine md_interactions::update_interactions_neighbour_lists (
                integer md_step,
                type(interaction), dimension(:) interactions,
                type(particles) atoms,
                type(particle_group), dimension(:) groups,
                type(simulation_cell) cell,
                real exe_time_nlsearch,
                real exe_time_nldistance )
```

См. определение в файле md_interactions.f90 строка 139

```
139     type(interaction):: interactions(:)
140     type(particles):: atoms
141     type(particle_group):: groups(:)
142     type(simulation_cell):: cell
143     integer:: i,j,md_step
144     real:: exe_time_nlsearch,exe_time_nldistance
145
146     do i=1,size(interactions)
147
148         call update_neighbour_list(md_step,interactions(i)%nl(1),atoms,&
149         groups(interactions(i)%group_nums(1)),groups(interactions(i)%group_nums(2)),cell,exe_time_nlsearch,
    exe_time_nldistance)
150
151         select case (interactions(i)%interaction_name)
152         case('lj','morse','ljc','morsec')
153             call converce_neighbour_list(interactions(i)%nl(2),&
154             groups(interactions(i)%group_nums(2)),interactions(i)%nl(1))
155         end select
156
157         select case (interactions(i)%interaction_name)
158         case('ljc','morsec')
159             do j=1,size(interactions)
160                 select case (interactions(j)%interaction_name)
161                 case('tb'); exit
162                 case('rebosc'); exit
163                 end select
164             enddo
165             if (j<=size(interactions)) then
166                 call update_nearest_neighbours_in_graphene(md_step,interactions(i)%nl(3),interactions(j)%nl
    (1),atoms,&
167                 groups(interactions(j)%group_nums(1)),cell)
168             else
169                 call update_neighbour_list(md_step,interactions(i)%nl(3),atoms,&
170                 groups(interactions(i)%group_nums(5)),groups(interactions(i)%group_nums(6)),cell,
    exe_time_nlsearch,exe_time_nldistance)
171             endif
172         end select
173
174     enddo
175
176     call update_norm_in_graphene(interactions)
177
```

Граф вызовов:



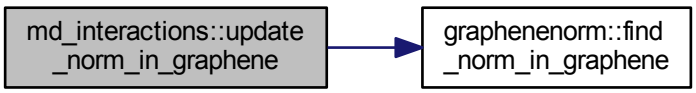
5.12.1.15 update_norm_in_graphene()

```
subroutine md_interactions::update_norm_in_graphene (
                type(interaction), dimension(:) interactions )
```

См. определение в файле md_interactions.f90 строка 196

```
196     type(interaction):: interactions(:)
197     integer:: i
198
199     do i=1,size(interactions)
200        select case (interactions(i)%interaction_name)
201        case('ljc')
202           call find_norm_in_graphene(interactions(i)%parameters%LJC(1)%gr_norm,interactions(i)%nl(3)%dr)
203        case('morsec')
204           call find_norm_in_graphene(interactions(i)%parameters%MorseC(1)%gr_norm,interactions(i)%nl(3)
    %dr)
205        end select
206     enddo
207
```

Граф вызовов:



5.13 Модуль md_neighbours

Модуль содержит подпрограммы относящиеся к спискам соседей частиц

Функции/подпрограммы

- subroutine create_neighbour_list (nl)

  Инициализирует пустой список соседей.
- subroutine update_neighbour_list (md_step, nl, atoms, group1, group2, box, exe_time_nlsearch, exe_time_nldistance)

  Вычисляет расстояния между соседями. Обновляет список соседей с нужной частотой. Расстояния между соседями вычисляются всегда. Список соседей обнавляется с периодом указанным в списке. Также замеряется затраченное время.
- subroutine find_neighbours (nl, atoms, group1, group2, box)

  Ищет соседей для частиц из первой группы среди второй группы. Группы могут совпадать.
- subroutine find_neighbour_distances (nl, atoms, group1, group2, box)

  Пересчитывает взаимное расположение соседей.
- subroutine converce_neighbour_list (cnl, group2, nl)

  Делает список соседей для второй группы из списка соседей для первой группы. Эквивалентно find_neighbours(cnl,atoms,group2,group1,box). Обращенный список соседей заполняется данными из списка полученного подпрогаммой find_neighbours(nl,atoms,group1,group2,box)

5.13.1 Подробное описание

Модуль содержит подпрограммы относящиеся к спискам соседей частиц

5.13.2 Функции/подпрограммы

5.13.2.1 converce_neighbour_list()

```
subroutine md_neighbours::converce_neighbour_list (
                type(neighbour_list) cnl,
                type(particle_group) group2,
                type(neighbour_list) nl )
```

Делает список соседей для второй группы из списка соседей для первой группы. Эквивалентно find_neighbours(cnl,atoms,group2,group1,box). Обращенный список соседей заполняется данными из списка полученного подпрогаммой find_neighbours(nl,atoms,group1,group2,box)

См. определение в файле md_neighbours.f90 строка 129

```
129     type(neighbour_list):: nl,cnl
130     type(particle_group):: group2
131     integer::       i,j,p
132
133     if (group2%N>0 .and. nl%N>0 .and. cnl%N>0) then
134         if (group2%N>cnl%N) then; write(*,*) 'error: group2%N>cnl%N',group2%N,cnl%N; stop; endif
135         !$OMP PARALLEL firstprivate(i)
136         !$OMP DO
137         do i=1,group2%N
138             cnl%particle_index(i) = group2%indexes(i)
139         enddo
140         !$OMP END DO
141         !$OMP DO
142         do i=1,cnl%N
143             cnl%nnum(i) = 0
144         enddo
145         !$OMP END DO
146         !$OMP END PARALLEL
147         do i=1,nl%N !can not be parallel!
148             do p=1,nl%nnum(i)
149                 j = nl%nlist(p,i)
150                 cnl%nnum(j) = cnl%nnum(j)+1
151                 if (cnl%nnum(j)>cnl%neighb_num_max) then; write(*,*) 'error: too many neighbours',j,cnl
    %nnum(j); stop; endif
152                 cnl%nlist(cnl%nnum(j),j) = i
153                 cnl%dr(:,cnl%nnum(j),j) = -nl%dr(:,p,i)
154                 cnl%moddr(cnl%nnum(j),j) = nl%moddr(p,i)
155             enddo
156         enddo
157     endif
158
159     return
```

5.13.2.2 create_neighbour_list()

```
subroutine md_neighbours::create_neighbour_list (
                type(neighbour_list) nl )
```

Инициализирует пустой список соседей.

См. определение в файле md_neighbours.f90 строка 10

```
10    type(neighbour_list):: nl
11
12    if (nl%N>0) then
13       allocate(nl%particle_index(nl%N))
14       allocate(nl%nnum(nl%N))
15       allocate(nl%lessnnum(nl%N))
16       allocate(nl%nlist(nl%neighb_num_max,nl%N))
17       allocate(nl%dr(3,nl%neighb_num_max,nl%N))
18       allocate(nl%moddr(nl%neighb_num_max,nl%N))
19       nl%particle_index = 0
20       nl%nnum = 0
21       nl%lessnnum = 0
22       nl%nlist = 0
23       nl%dr = 0.
24       nl%moddr = 0.
25    endif
26
27    return
```

5.13.2.3 find_neighbour_distances()

```
subroutine md_neighbours::find_neighbour_distances (
                type(neighbour_list) nl,
                type(particles) atoms,
                type(particle_group) group1,
                type(particle_group) group2,
                type(simulation_cell) box )
```

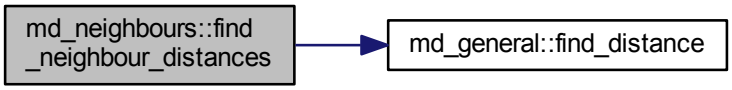
Пересчитывает взаимное расположение соседей.

Необходимо сделать Сделать sqrt(dr2) быстрее.

См. определение в файле md_neighbours.f90 строка 105

```
105   type(particles)::   atoms
106   type(particle_group):: group1,group2
107   type(simulation_cell):: box
108   type(neighbour_list):: nl
109   real:: dr2
110   integer:: i,p
111
112   !$OMP PARALLEL firstprivate(i,p,dr2)
113   !$OMP DO
114   do i=1,nl%N
115      do p=1,nl%nnum(i)
116         call find_distance(nl%dr(:,p,i),dr2,atoms%positions(:,group1%indexes(i)),atoms%positions(:,
    group2%indexes(nl%nlist(p,i))),box)
117         nl%moddr(p,i) = sqrt(dr2)
118      enddo
119   enddo
120   !$OMP END DO
121   !$OMP END PARALLEL
122
123   return
```

Граф вызовов:



5.13.2.4 find_neighbours()

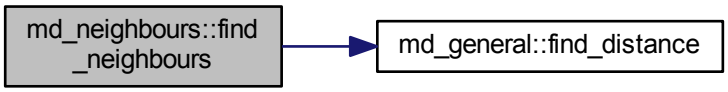
```
subroutine md_neighbours::find_neighbours (
                type(neighbour_list) nl,
                type(particles) atoms,
                type(particle_group) group1,
                type(particle_group) group2,
                type(simulation_cell) box )
```

Ищет соседей для частиц из первой группы среди второй группы. Группы могут совпадать.

См. определение в файле md_neighbours.f90 строка 57

```
57    type(particles):: atoms
58    type(particle_group):: group1,group2
59    type(simulation_cell):: box
60    type(neighbour_list):: nl
61    real:: dr(3),dr2
62    integer:: i,j,k,ind,jnd,nnumind,lessnnumind
63
64    if (group1%N>nl%N) then; write(*,*) 'error: group1%N>nl%N',group1%N,nl%N; stop; endif
65    !$OMP PARALLEL firstprivate(i,j,k,ind,jnd,dr,dr2,nnumind,lessnnumind)
66    !$OMP DO
67    do ind=1,group1%N
68       i = group1%indexes(ind)
69       nl%particle_index(ind) = group1%indexes(ind)
70       !nl%nlist(:nl%nnum(ind),ind) = 0
71       nnumind = 0
72       lessnnumind = -1
73       do jnd=1,group2%N
74          j = group2%indexes(jnd)
75          if (i/=j) then
76             call find_distance(dr,dr2,atoms%positions(:,i),atoms%positions(:,j),box)
77             if (dr2<nl%r_cut*nl%r_cut) then
78                if (lessnnumind==-1 .and. i<j) lessnnumind = nnumind
79                nnumind = nnumind+1
80                if (nnumind>nl%neighb_num_max) then; write(*,*) 'error: too many neighbours',ind,
    nnumind; stop; endif
81                nl%nlist(nnumind,ind) = jnd
82                nl%dr(1,nnumind,ind) = dr(1)
83                nl%dr(2,nnumind,ind) = dr(2)
84                nl%dr(3,nnumind,ind) = dr(3)
85                nl%moddr(nnumind,ind) = sqrt(dr2)
86             endif
87          endif
88       enddo
89       if(lessnnumind==-1) then
90          nl%lessnnum(ind) = nnumind
91       else
92          nl%lessnnum(ind) = lessnnumind
93       endif
94       nl%nnum(ind) = nnumind
95    enddo
96    !$OMP END DO
97    !$OMP END PARALLEL
98
99    return
```

Граф вызовов:



5.13.2.5 update_neighbour_list()

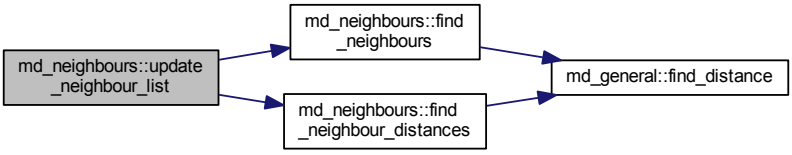
```
subroutine md_neighbours::update_neighbour_list (
                integer md_step,
                type(neighbour_list) nl,
                type(particles) atoms,
                type(particle_group) group1,
                type(particle_group) group2,
                type(simulation_cell) box,
                real exe_time_nlsearch,
                real exe_time_nldistance )
```

Вычисляет расстояния между соседями. Обновляет список соседей с нужной частотой. Расстояния между соседями вычисляются всегда. Список соседей обнавляется с периодом указанным в списке. Также замеряется затраченное время.

См. определение в файле md_neighbours.f90 строка 33

```
33    type(particles):: atoms
34    type(particle_group):: group1,group2
35    type(simulation_cell):: box
36    type(neighbour_list):: nl
37    integer:: md_step
38    real:: exe_t,exe_time_nlsearch,exe_time_nldistance,omp_get_wtime
39
40    if (group1%N>0 .and. group2%N>0 .and. nl%N>0) then
41       if (mod(md_step,nl%update_period)==0) then
42          exe_t = omp_get_wtime()
43          call find_neighbours(nl,atoms,group1,group2,box)
44          exe_time_nlsearch = exe_time_nlsearch+omp_get_wtime()-exe_t
45       else
46          exe_t = omp_get_wtime()
47          call find_neighbour_distances(nl,atoms,group1,group2,box)
48          exe_time_nldistance = exe_time_nldistance+omp_get_wtime()-exe_t
49       endif
50    endif
51
```

Граф вызовов:



5.14 Модуль md_read_write

Модуль ввода вывода .xyz файлов и настроек моделирования.

Функции/подпрограммы

- subroutine read_integrator_params (integr, file_id)

  Читает параметры интегратора.
- subroutine read_box_size (box, filename)

  Читает параметры моделируемой ячейки в файле .xyz. Читает второю строку в xyz файле. В размер ячейки задается тремя векторами. В общем случае ячейка триклинная.
- subroutine read_particles (atoms, filename)

  Читает информацию о частицах из .xyz файла. Первая строка файла содержит количество частиц. Третья и последующие - координаты по X, Y, Z, скорости по X, Y, Z, массу и название частицы. Всего 8 столбцов для каждой частицы.
- subroutine write_particle_group (filename, atoms, group, box)

  Выводит информацию о группе частиц в новый файл.
- subroutine write_particle_group_append (filename, atoms, group, box, md_step)

  Выводит информацию о группе частиц в конец уже имеющегося файла.

5.14.1 Подробное описание

Модуль ввода вывода .xyz файлов и настроек моделирования.

Необходимо сделать Добавить чтение файла настроек.

5.14.2 Функции/подпрограммы

5.14.2.1 read_box_size()

```
subroutine md_read_write::read_box_size (
                type(simulation_cell) box,
                character(*) filename )
```

Читает параметры моделируемой ячейки в файле .xyz. Читает второю строку в xyz файле. В размер ячейки задается тремя векторами. В общем случае ячейка триклинная.

Предупреждения

Данная подпрограмма (как и вся программа) рассчитана только на прямоугольные ячейки.

См. определение в файле md_read_write.f90 строка 23

```
23    type(simulation_cell):: box
24    real:: boxmatrix(9)
25    character(*):: filename
26    character(len=32):: c
27
28    open(1,file=filename)
29    read(1,*)
30    read(1,*) c,boxmatrix
31    close(1)
32    box%box_size(1)=boxmatrix(1)
33    box%box_size(2)=boxmatrix(5)
34    box%box_size(3)=boxmatrix(9)
35    box%half_box_size = 0.5*box%box_size
36
37    return
```

5.14.2.2 read_integrator_params()

```
subroutine md_read_write::read_integrator_params (
                type(integrator_params) integr,
                integer file_id )
```

Читает параметры интегратора.

См. определение в файле md_read_write.f90 строка 11

```
11    type(integrator_params):: integr
12    integer:: file_id
13
14    read(file_id,*) integr%int_name,integr%dt,integr%l,integr%period_snapshot,integr%period_log
15
16    return
```

5.14.2.3 read_particles()

```
subroutine md_read_write::read_particles (
                type(particles) atoms,
                character(*) filename )
```

Читает информацию о частицах из .xyz файла. Первая строка файла содержит количество частиц. Третья и последующие - координаты по X, Y, Z, скорости по X, Y, Z, массу и название частицы. Всего 8 столбцов для каждой частицы.

См. определение в файле md_read_write.f90 строка 43

```
43    type(particles)::   atoms
44    character(*):: filename
45    integer::       i
46
47    open(1,file=filename)
48    read(1,*) atoms%N
49    allocate(atoms%positions(3,atoms%N))
50    allocate(atoms%velocities(3,atoms%N))
51    allocate(atoms%forces(3,atoms%N))
52    allocate(atoms%masses(atoms%N))
53    allocate(atoms%atom_types(atoms%N))
54    read(1,*)
55    do i=1,atoms%N
56        read(1,*) atoms%positions(:,i),atoms%velocities(:,i),atoms%masses(i),atoms%atom_types(i)
57    enddo
58    close(1)
59
60    return
```

5.14.2.4 write_particle_group()

```
subroutine md_read_write::write_particle_group (
                character(*) filename,
                type(particles) atoms,
                type(particle_group) group,
                type(simulation_cell) box )
```

Выводит информацию о группе частиц в новый файл.

Предупреждения

Если файл с таким именем уже существует, он будет заменен новым.

См. определение в файле md_read_write.f90 строка 66

```
66    type(particles)::   atoms
67    type(particle_group):: group
68    type(simulation_cell):: box
69    character(*):: filename
70    integer::       i,ind
71
72    open(2,file=filename)
73    write(2,*) group%N
74    write(2,'(A,9f16.6,A)') 'Lattice="',box%box_size(1), 0., 0., 0., box%box_size(2), 0., 0., 0., box
      %box_size(3),&
75          '" Properties=pos:R:3:vel:R:3:mass:R:1:species:S:1'
76    do ind=1,group%N
77       i = group%indexes(ind)
78       write(2,'(7f27.16,A,A)') atoms%positions(:,i),atoms%velocities(:,i),atoms%masses(i),'    ',atoms
      %atom_types(i)
79    enddo
80    close(2)
81
82    return
```

5.14.2.5 write_particle_group_append()

```
subroutine md_read_write::write_particle_group_append (
                character(*) filename,
                type(particles) atoms,
                type(particle_group) group,
                type(simulation_cell) box,
                integer md_step )
```

Выводит информацию о группе частиц в конец уже имеющегося файла.

См. определение в файле md_read_write.f90 строка 87

```
87    type(particles)::   atoms
88    type(particle_group):: group
89    type(simulation_cell):: box
90    character(*):: filename
91    integer::       i,ind,md_step
92
93    if(group%N>0) then
94       open(2,file=filename,action='write',position='append')
95       write(2,*) group%N
96       write(2,'(A,i9,A,9f10.4,A)') 'time_step: ',md_step,&
97       '    Lattice="',box%box_size(1), 0., 0., 0., box%box_size(2), 0., 0., 0., box%box_size(3),&
98       '" Properties=pos:R:3:vel:R:3:mass:R:1:species:S:1'
99       do ind=1,group%N
100           i = group%indexes(ind)
101           write(2,'(7f10.4,A,A)') atoms%positions(:,i),atoms%velocities(:,i),atoms%masses(i),'    ',trim(
      atoms%atom_types(i))
102       enddo
103       close(2)
104    endif
105
106    return
```

5.15 Модуль md_simulation

Функции/подпрограммы

- subroutine md (out_id, all_out_id, input_path, settings_filename, output_prefix, out_period, num_of_omp_treads)

5.15.1 Функции/подпрограммы

5.15.1.1 md()

```
subroutine md_simulation::md (
                integer out_id,
                integer all_out_id,
                character(len=128) input_path,
                character(len=128) settings_filename,
                character(len=128) output_prefix,
                integer out_period,
                integer num_of_omp_treads )
```

См. определение в файле md_simulation.f90 строка 10

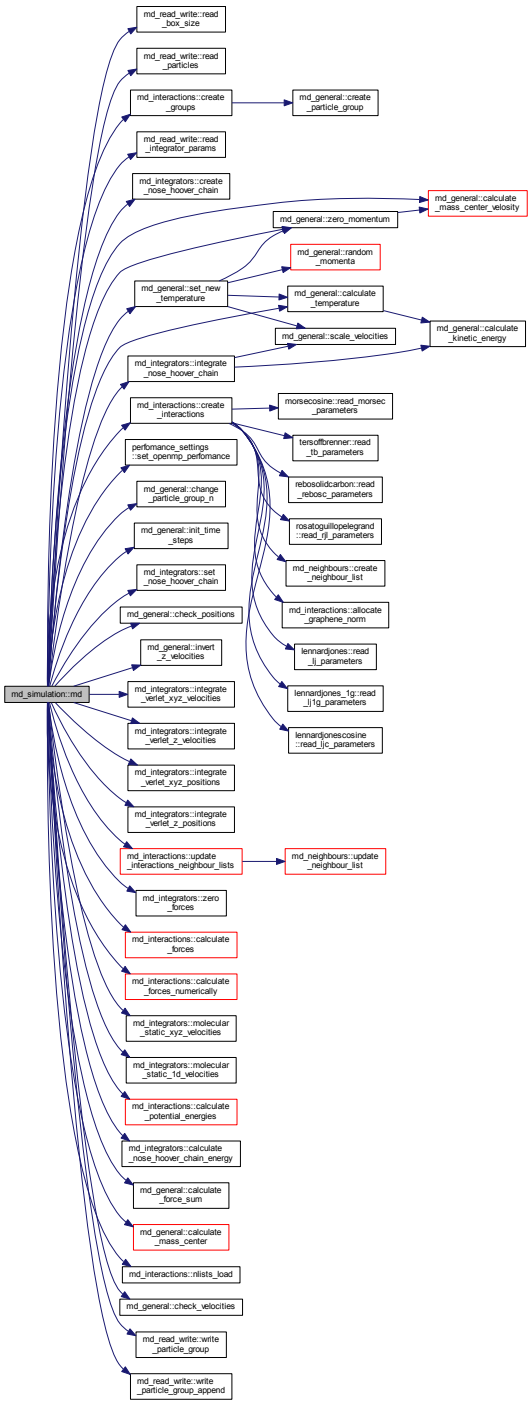
```
10
11 type(simulation_cell)                :: cell
12 type(time_steps)                     :: dt
13 type(particles)                      :: atoms
14 type(particle_group),allocatable     :: groups(:)
15 type(interaction),allocatable        :: interactions(:)
16 type(integrator_params),allocatable  :: integrators(:)
17 type(nose_hoover_chain)              :: nhc
18 real                                 :: exe_t,exe_time_start,exe_time_md,exe_time_pos_vel,&
19                                         exe_time_nlists,exe_time_nlsearch,exe_time_nldistance,
   exe_time_forces,exe_time_energy,&
20                                         conserved_energy,total_energy,kinetic_energy,potential_energy,
   prev_potential_energy,&
21                                         ms_de,fs(3),mcv(3),mc(3),initial_temperature,target_temperature
   ,temperature,nhc_q1
22 integer                              :: i,num_of_omp_treads,md_step,md_step_limit,integrators_num,
   integrator_index,&
23                                         file_id,out_id,log_id,out_period,all_out_id,&
24                                         all_atoms_group_num,termo_atoms_group_num,&
25                                         all_moving_atoms_group_num,xyz_moving_atoms_group_num,
   z_moving_atoms_group_num,&
26                                         traj_group_num,period_traj,change_group_num,&
27                                         zero_momentum_period
28 integer,allocatable                   :: group_change_from(:),group_change_to(:),change_ts1(:),
   change_ts2(:),change_frec(:)
29 character(len=128)                    :: str,filename,logfilename,init_xyz_filename,input_path,
   settings_filename,output_prefix
30 character(len=32)                     :: integrator_name,interactions_energies_format
31 logical                               :: new_velocities,invert_z_vel
32 real                                  :: omp_get_wtime
33
34 exe_time_start = omp_get_wtime()
35 log_id = 108
36 file_id = 2017
37
38 open(file_id,file=trim(input_path)//settings_filename); write(out_id,'(A,A)') 'settings_filename: ',trim(
   settings_filename)
39 read(file_id,*) str,md_step_limit;                    write(out_id,'(A32,i12)') str,md_step_limit
40 read(file_id,*) str,logfilename;                      write(out_id,'(A32,A,A)') str,' ',trim(logfilename)
41 read(file_id,*) str,init_xyz_filename;                write(out_id,'(A32,A,A)') str,' ',trim(
   init_xyz_filename)
42 call read_box_size(cell,trim(input_path)//init_xyz_filename);   write(out_id,'(A,3f16.6)')  'box_size: ',
   cell%box_size
43 call read_particles(atoms,trim(input_path)//init_xyz_filename); write(out_id,'(A,i12)')     'particles_num:
```

```
    ',atoms%N
44 read(file_id,*) str,new_velocities;                    write(out_id,'(A32,l8)') str,new_velocities
45 read(file_id,*) str,zero_momentum_period;              write(out_id,'(A32,i12)') str,zero_momentum_period
46 call create_groups(groups,file_id,out_id,atoms)
47 read(file_id,*) str,all_moving_atoms_group_num;        write(out_id,'(A32,i12)') str,
   all_moving_atoms_group_num
48 read(file_id,*) str,xyz_moving_atoms_group_num;        write(out_id,'(A32,i12)') str,
   xyz_moving_atoms_group_num
49 read(file_id,*) str,z_moving_atoms_group_num;          write(out_id,'(A32,i12)') str,
   z_moving_atoms_group_num
50 read(file_id,*) str,termo_atoms_group_num;            write(out_id,'(A32,i12)') str,termo_atoms_group_num
51 read(file_id,*) str,all_atoms_group_num;              write(out_id,'(A32,i12)') str,all_atoms_group_num
52 read(file_id,*) str,traj_group_num;                   write(out_id,'(A32,i12)') str,traj_group_num
53 read(file_id,*) str,period_traj;                      write(out_id,'(A32,i12)') str,period_traj
54 read(file_id,*) str,change_group_num;                 write(out_id,'(A32,i12)') str,change_group_num
55 allocate(group_change_from(change_group_num),group_change_to(change_group_num),&
56 change_ts1(change_group_num),change_ts2(change_group_num),change_frec(change_group_num))
57 do i=1,change_group_num
58     read(file_id,*) str,group_change_from(i),group_change_to(i);
59     write(out_id,'(A32,2i12)') str,group_change_from(i),group_change_to(i)
60     read(file_id,*) str,change_ts1(i),change_ts2(i),change_frec(i)
61     write(out_id,'(A32,3i12)') str,change_ts1(i),change_ts2(i),change_frec(i)
62 enddo
63 read(file_id,*) str,invert_z_vel;                       write(out_id,'(A32,l8)') str,invert_z_vel
64 read(file_id,*) str,integrators_num;                    write(out_id,'(A32,i12)') str,integrators_num
65 allocate(integrators(0:integrators_num)); integrators(0)%int_name='none'; integrators(0)%dt=0.; integrators
   (0)%l=0
66 read(file_id,'(A)') str;                               write(out_id,'(A)') str
67 do i=1,integrators_num
68     call read_integrator_params(integrators(i),file_id)
69     write(out_id,'(A,A6,f10.5,4i9)') '  ',integrators(i)%int_name,&
70     integrators(i)%dt,integrators(i)%l,integrators(i)%period_snapshot,integrators(i)%period_log
71 enddo
72 read(file_id,*) str,ms_de;                              write(out_id,'(A32,f16.6)') str,ms_de
73 read(file_id,*) str,target_temperature,nhc%M,nhc_q1;    write(out_id,'(A32,f16.6,i6,f16.6)') str,
   target_temperature,nhc%M,nhc_q1
74 call create_nose_hoover_chain(nhc)
75 read(file_id,*) str,initial_temperature;               write(out_id,'(A32,f16.6)') str,initial_temperature
76 if (initial_temperature<0.) initial_temperature=0.
77 if (new_velocities) call set_new_temperature(atoms,groups(all_moving_atoms_group_num),initial_temperature)
78 call create_interactions(interactions,groups,file_id,out_id,input_path)
79 close(file_id)
80
81 write(interactions_energies_format,'("(",i0,"f20.9)")') size(interactions)
82 write(str,'(A,A)') trim(output_prefix),trim(logfilename)
83 open(log_id,file=str)
84 write(out_id,'(A24,1f10.2,A)') 'PREPARATIONS TIME: ',omp_get_wtime()-exe_time_start,' S '
85 write(out_id,'(A24,i6,A)') 'RUNNING ON ',num_of_omp_treads,' OPENMP THREADS'
86 call set_openmp_perfomance(num_of_omp_treads,atoms%N)
87 exe_time_md = omp_get_wtime()
88 exe_time_pos_vel = 0.
89 exe_time_nlists = 0.
90 exe_time_forces = 0.
91 exe_time_energy = 0.
92 integrator_index = 0
93 dt%simulation_time = 0.
94
95 do md_step=0,md_step_limit
96
97     do i=1,change_group_num
98         call change_particle_group_n(groups(group_change_to(i)),md_step,change_ts1(i),change_ts2(i),&
99         change_frec(i),groups(group_change_from(i)))
100     enddo
101
102     if (md_step-1==sum(integrators(0:integrator_index)%l) .or. integrator_index==0) then
103         integrator_index = integrator_index+1
104         do i=integrator_index,integrators_num
105             if (integrators(i)%l>0) then
106                 integrator_name = integrators(i)%int_name
107                 call init_time_steps(dt,integrators(i)%dt)
108                 if (integrator_name=='nvt') call set_nose_hoover_chain(nhc,target_temperature,nhc_q1,groups
    (termo_atoms_group_num)%N)
109                 exit
110             endif
111         enddo
112         if (i>integrators_num .or. (i==integrators_num .and. integrators(integrators_num)%l<1)) then
113             write(out_id,*) 'no more integrators'
114             exit
115         endif
116         integrator_index = i
117     endif
118
119     exe_t = omp_get_wtime()
120     call check_positions(out_id,atoms,cell)
121     if (invert_z_vel) call invert_z_velocities(atoms,0.8*cell%box_size(3),0.9*cell%box_size(3))
122     if (md_step/=0) then
123         if (integrator_name=='nvms') then
```

```
124             if (abs(potential_energy-prev_potential_energy)<=ms_de) then
125                 write(out_id,*) 'potential energy diffrence is small enough'
126                 exit
127             endif
128         endif
129         if (integrator_name=='nvt') call integrate_nose_hoover_chain(nhc,atoms,groups(termo_atoms_group_num
    ),dt)
130         call integrate_verlet_xyz_velocities(atoms,groups(xyz_moving_atoms_group_num),dt)
131         call integrate_verlet_z_velocities(atoms,groups(z_moving_atoms_group_num),dt)
132         call integrate_verlet_xyz_positions(atoms,groups(xyz_moving_atoms_group_num),dt,cell)
133         call integrate_verlet_z_positions(atoms,groups(z_moving_atoms_group_num),dt,cell)
134     endif
135     prev_potential_energy = potential_energy
136     exe_time_pos_vel = exe_time_pos_vel+omp_get_wtime()-exe_t
137
138     exe_t = omp_get_wtime()
139     call update_interactions_neighbour_lists(md_step,interactions,atoms,groups,cell,exe_time_nlsearch,
    exe_time_nldistance)
140     exe_time_nlists = exe_time_nlists+omp_get_wtime()-exe_t
141     exe_t = omp_get_wtime()
142     if (mod(md_step,zero_momentum_period)==0) call zero_momentum(atoms,groups(all_atoms_group_num))
143     call zero_forces(atoms,groups(all_atoms_group_num))
144     call calculate_forces(atoms,interactions)
145     call calculate_forces_numerically(atoms,interactions)
146     exe_time_forces = exe_time_forces+omp_get_wtime()-exe_t
147
148     exe_t = omp_get_wtime()
149     if (md_step/=0) then
150         call integrate_verlet_xyz_velocities(atoms,groups(xyz_moving_atoms_group_num),dt)
151         call integrate_verlet_z_velocities(atoms,groups(z_moving_atoms_group_num),dt)
152         if (integrator_name=='nvt') call integrate_nose_hoover_chain(nhc,atoms,groups(termo_atoms_group_num
    ),dt)
153         if (integrator_name=='nvms') call molecular_static_xyz_velocities(atoms,groups(
    xyz_moving_atoms_group_num))
154         if (integrator_name=='nvms') call molecular_static_1d_velocities(atoms,groups(
    z_moving_atoms_group_num))
155         dt%simulation_time = dt%simulation_time+dt%ts(1)
156     endif
157     exe_time_pos_vel = exe_time_pos_vel+omp_get_wtime()-exe_t
158
159     if (mod(md_step,integrators(integrator_index)%period_log)==0 .or. mod(md_step,out_period)==0 &
160     .or. mod(md_step,out_period)==0 .or. integrator_name=='nvms') then
161
162         exe_t = omp_get_wtime()
163         call calculate_potential_energies(interactions)
164         potential_energy = sum(interactions%energy)
165         call calculate_temperature(temperature,kinetic_energy,atoms,groups(all_moving_atoms_group_num))
166         total_energy = potential_energy+kinetic_energy
167         if (integrator_name=='nvt') call calculate_nose_hoover_chain_energy(nhc)
168         conserved_energy = total_energy+nhc%e
169         exe_time_energy = exe_time_energy+omp_get_wtime()-exe_t
170
171         if (mod(md_step,integrators(integrator_index)%period_log)==0) then
172             write(log_id,'(A6,i9,6f24.6)',advance='no') trim(integrator_name),md_step,dt%simulation_time,&
173             conserved_energy,total_energy,potential_energy,kinetic_energy,temperature
174             write(log_id,interactions_energies_format) interactions%energy
175         endif
176
177         if (mod(md_step,out_period)==0) then
178             call calculate_force_sum(fs,atoms,groups(all_atoms_group_num))
179             call calculate_mass_center(mc,atoms,groups(all_atoms_group_num))
180             call calculate_mass_center_velosity(mcv,atoms,groups(all_atoms_group_num))
181             write(out_id,'(f9.2,A6,i10,f21.9,2e11.4,3f9.4)') omp_get_wtime()-exe_time_start,&
182             trim(integrator_name),md_step,conserved_energy,sqrt(sum(fs**2)),sqrt(sum(mcv**2)),mc
183             call nlists_load(out_id,interactions)
184             call check_velocities(out_id,atoms)
185         endif
186
187     endif
188
189     if (mod(md_step,integrators(integrator_index)%period_snapshot)==0 .and. md_step/=0) then
190         write(filename,'(A,A,i6.6,A)') trim(output_prefix),'snapshot_',md_step,'.xyz'
191         call write_particle_group(filename,atoms,groups(all_atoms_group_num),cell)
192     endif
193
194     if (mod(md_step,period_traj)==0) then
195         write(filename,'(A,A,i2.2,A)') trim(output_prefix),'traj_',traj_group_num,'.xyz'
196         call write_particle_group_append(filename,atoms,groups(traj_group_num),cell,md_step)
197     endif
198
199 enddo
200
201 exe_time_md = omp_get_wtime()-exe_time_md
202 if (integrator_name=='nvms') write(out_id,*) 'potential energy difference: ',potential_energy-
    prev_potential_energy
203 write(out_id,*) 'steps number:', md_step-1
204
```

```
205 write(out_id,'(A24,f10.2,A,f10.2,A)') 'MD:',exe_time_md,' S ',exe_time_md/exe_time_md*100,'%'
206 write(out_id,'(A24,f10.2,A,f10.2,A)') 'POSITION AND VELOCITY:',exe_time_pos_vel,' S ',exe_time_pos_vel/
    exe_time_md*100,'%'
207 write(out_id,'(A24,f10.2,A,f10.2,A)') 'NEIGHBOURS:',exe_time_nlists,' S ',exe_time_nlists/exe_time_md*100,
    '%'
208 write(out_id,'(A24,f10.2,A,f10.2,A)') 'NEIGHBOURS SEARCH:',exe_time_nlsearch,' S ',exe_time_nlsearch/
    exe_time_md*100,'%'
209 write(out_id,'(A24,f10.2,A,f10.2,A)') 'NEIGHBOURS DISTANCE:',exe_time_nldistance,' S ',exe_time_nldistance/
    exe_time_md*100,'%'
210 write(out_id,'(A24,f10.2,A,f10.2,A)') 'FORCES:',exe_time_forces,' S ',exe_time_forces/exe_time_md*100,'%'
211 write(out_id,'(A24,f10.2,A,f10.2,A)') 'ENERGY:',exe_time_energy,' S ',exe_time_energy/exe_time_md*100,'%'
212 write(out_id,'(A24,f10.2,A,f10.2,A)') 'REST:',exe_time_md-exe_time_pos_vel-exe_time_nlists-exe_time_forces-
    exe_time_energy,&
213 ' S ',(exe_time_md-exe_time_pos_vel-exe_time_nlists-exe_time_forces-exe_time_energy)/exe_time_md*100,'%'
214 write(out_id,'(A24,f16.2)') 'TIME STEPS PER HOUR:',md_step/exe_time_md*3600
215
216 write(all_out_id,'(A32,i9,5f20.9)',advance='no') trim(output_prefix),md_step-1,dt%simulation_time,&
217 total_energy,potential_energy,kinetic_energy,temperature
218 write(all_out_id,interactions_energies_format,advance='no') interactions%energy
219 write(log_id,'(A32,i9,5f20.9)',advance='no') trim(output_prefix),md_step-1,dt%simulation_time,&
220 total_energy,potential_energy,kinetic_energy,temperature
221 write(log_id,interactions_energies_format,advance='no') interactions%energy
222
223 write(filename,'(A,A,A)') trim(output_prefix),'final_',trim(init_xyz_filename)
224 if (md_step/=0) call write_particle_group(filename,atoms,groups(all_atoms_group_num),cell)
225 close(log_id)
226
```

Граф вызовов:



5.16 Модуль morsecosine

Типы данных

- type morsecosine_parameters

Функции/подпрограммы

- subroutine read_morsec_parameters (MorseCp, filename)
- subroutine morsec_energy (energy, nl, MorseCp)
- subroutine morsec_forces_for_graphene (atoms, nl, nl_nn, MorseCp)
- subroutine morsec_forces_for_other_atoms (atoms, nl, MorseCp)

5.16.1 Функции/подпрограммы

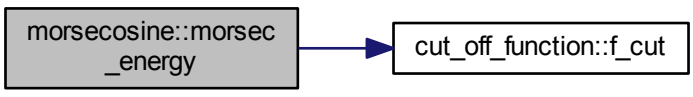
5.16.1.1 morsec_energy()

```
subroutine morsecosine::morsec_energy (
                real energy,
                type(neighbour_list) nl,
                type(morsecosine_parameters) MorseCp )
```

См. определение в файле MorseCosine.f90 строка 27

```
27    type(neighbour_list):: nl
28    type(MorseCosine_parameters):: MorseCp
29    integer:: i,p
30    real:: energy,energy_priv,V1,V2,V3
31
32    energy = 0.
33    energy_priv = 0.
34    !$OMP PARALLEL firstprivate(energy_priv,i,p,V1,V2,V3)
35    !$OMP DO
36    do i=1,nl%N
37       do p=1,nl%nnum(i)
38          if (nl%moddr(p,i)<morsecp%R2) then
39             v2 = exp(-morsecp%a*(nl%moddr(p,i)-morsecp%r))
40             v1 = v2**2
41             v3 = (abs(sum(morsecp%gr_norm(:,i)*nl%dr(:,p,i)))/nl%moddr(p,i))**morsecp%delt
42             energy_priv = energy_priv+morsecp%d*(v1-2.*v2*v3)*f_cut(nl%moddr(p,i),morsecp%R1,morsecp%R2
   )
43          endif
44       enddo
45    enddo
46    !$OMP END DO
47    !$OMP ATOMIC
48       energy = energy+energy_priv
49    !$OMP END PARALLEL
50
```

Граф вызовов:



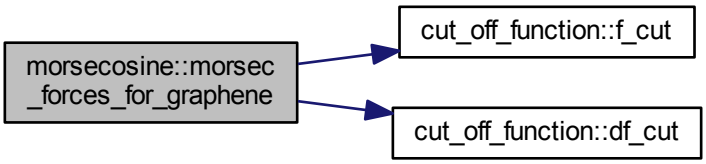
5.16.1.2 morsec_forces_for_graphene()

```
subroutine morsecosine::morsec_forces_for_graphene (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(neighbour_list) nl_nn,
                type(morsecosine_parameters) MorseCp )
```

См. определение в файле MorseCosine.f90 строка 54

```
54    type(particles):: atoms
55    type(neighbour_list):: nl,nl_nn
56    type(MorseCosine_parameters):: MorseCp
57    integer:: i,p,q,l1,l2,l3,j,k,nnum_nn
58    real:: drj12(3),drj31(3),drj23(3),V1,V2,V3,f_c,df_c
59
60    if (nl%N/=nl_nn%N .and. nl%N/=0) then; write(*,*) 'error: nl%N/=nl_nn%N',nl%N,nl_nn%N; stop; endif
61    nnum_nn = 3
62    if (morsecp%simplified) then; nnum_nn = 0; morsecp%gr_norm = 0.; morsecp%gr_norm(3,:) = 1.; endif
    !optimize
63
64    !$OMP PARALLEL firstprivate(i,p,q,l1,l2,l3,j,k,drj12,drj31,drj23,V1,V2,V3,f_c,df_c)
65    !$OMP DO
66    do i=1,nl%N
67       do p=1,nl%nnum(i)
68          if (nl%moddr(p,i)<morsecp%R2) then
69             v2 = exp(-morsecp%a*(nl%moddr(p,i)-morsecp%r))
70             v1 = v2**2
71             v3 = (abs(sum(morsecp%gr_norm(:,i)*nl%dr(:,p,i)))/nl%moddr(p,i))**morsecp%delt
72             f_c = f_cut(nl%moddr(p,i),morsecp%R1,morsecp%R2)
73             df_c = df_cut(nl%moddr(p,i),morsecp%R1,morsecp%R2)
74             atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))-morsecp%d*&
75             ((2.*(morsecp%a*v1-(morsecp%a+morsecp%delt/nl%moddr(p,i))*v2*v3)/nl%moddr(p,i)*f_c-(v1-2.*
    v2*v3)*df_c)*nl%dr(:,p,i)+&
76             (2.*morsecp%delt*v2*v3/sum(morsecp%gr_norm(:,i)*nl%dr(:,p,i))*f_c)*morsecp%gr_norm(:,i))
77          endif
78       enddo
79    enddo
80    !$OMP END DO
81    !$OMP DO
82    do i=1,nl%N
83       do q=1,nnum_nn
84          j = nl_nn%nlist(q,i)
85          do l1=1,nnum_nn
86             if (nl_nn%nlist(l1,j)==i) exit
87          enddo
88          if(l1>3) then; write(*,*) 'l1>3'; stop; endif
89          l2 = mod(l1,3)+1
90          l3 = mod(l1+1,3)+1
91          drj12 = nl_nn%dr(:,l2,j)-nl_nn%dr(:,l1,j)
92          drj31 = nl_nn%dr(:,l1,j)-nl_nn%dr(:,l3,j)
93          drj23 = nl_nn%dr(:,l3,j)-nl_nn%dr(:,l2,j)
94          do p=1,nl%nnum(j)
95             if (nl%moddr(p,j)<morsecp%R2) then
96                v2 = exp(-morsecp%a*(nl%moddr(p,j)-morsecp%r))
97                v1 = v2**2
98                v3 = (abs(sum(morsecp%gr_norm(:,j)*nl%dr(:,p,j)))/nl%moddr(p,j))**morsecp%delt
99                f_c = f_cut(nl%moddr(p,j),morsecp%R1,morsecp%R2)
100                atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))&
101                -2.*morsecp%d*morsecp%delt*v2*v3*f_c/(sum(drj12**2)*sum(drj31**2)-sum(drj12*drj31)**2)/
    sum(morsecp%gr_norm(:,j)*nl%dr(:,p,j))*&
102                morsecp%gr_norm(:,j)*sum(nl%dr(:,p,j)*(drj12*sum(drj23*drj31)-drj31*sum(drj23*drj12)))
103             endif
104          enddo
105       enddo
106    enddo
107    !$OMP END DO
108    !$OMP END PARALLEL
109
```

Граф вызовов:



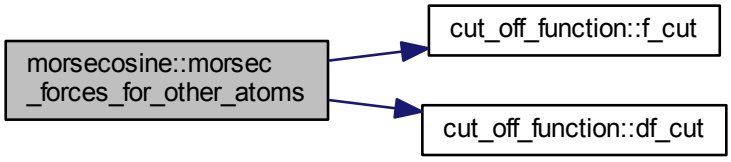
5.16.1.3 morsec_forces_for_other_atoms()

```
subroutine morsecosine::morsec_forces_for_other_atoms (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(morsecosine_parameters) MorseCp )
```

См. определение в файле MorseCosine.f90 строка 113

```
113     type(particles)::   atoms
114     type(neighbour_list):: nl
115     type(MorseCosine_parameters):: MorseCp
116     integer:: i,p
117     real:: gr_n(3),V1,V2,V3,f_c,df_c
118
119     !$OMP PARALLEL firstprivate(i,p,gr_n,V1,V2,V3,f_c,df_c)
120     !$OMP DO
121     do i=1,nl%N
122        do p=1,nl%nnum(i)
123           if (nl%moddr(p,i)<morsecp%R2) then
124              gr_n = morsecp%gr_norm(:,nl%nlist(p,i))
125              v2 = exp(-morsecp%a*(nl%moddr(p,i)-morsecp%r))
126              v1 = v2**2
127              v3 = (abs(sum(gr_n*nl%dr(:,p,i)))/(sqrt(sum(gr_n**2))*nl%moddr(p,i)))**morsecp%delt
128              f_c = f_cut(nl%moddr(p,i),morsecp%R1,morsecp%R2)
129              df_c = df_cut(nl%moddr(p,i),morsecp%R1,morsecp%R2)
130              atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))-morsecp%d*&
131              ((2.*(morsecp%a*v1-(morsecp%a+morsecp%delt/nl%moddr(p,i))*v2*v3)/nl%moddr(p,i)*f_c-(v1-2.*
    v2*v3)*df_c)*nl%dr(:,p,i)+&
132              (2.*morsecp%delt*v2*v3/sum(gr_n*nl%dr(:,p,i))*f_c)*gr_n)
133           endif
134        enddo
135     enddo
136     !$OMP END DO
137     !$OMP END PARALLEL
138
```

Граф вызовов:



5.16.1.4 read_morsec_parameters()

```
subroutine morsecosine::read_morsec_parameters (
                type(morsecosine_parameters) MorseCp,
                character(*) filename )
```

См. определение в файле MorseCosine.f90 строка 15

```
15    type(MorseCosine_parameters):: MorseCp
16    character(*):: filename
17
18    open(1,file=filename)
19    read(1,*) morsecp%d,morsecp%r,morsecp%a,morsecp%delt
20    read(1,*) morsecp%R1,morsecp%R2
21    read(1,*) morsecp%simplified
22    close(1)
23
```

5.17 Модуль perfomance_settings

Модуль настройки OpenMP параллелизма.

Функции/подпрограммы

- subroutine set_openmp_perfomance (num_of_omp_treads, N)

  Настраивает количество OpenMP потоков и omp_chunk_size.

Переменные

- integer omp_chunk_size

  Массивы данных будут разбиваться на части длины omp_chunk_size при использовании omp parallel for schedule(dynamic ,omp_chunk_size)

5.17.1 Подробное описание

Модуль настройки OpenMP параллелизма.

Предупреждения

omp_chunk_size пока нигде не используется.

5.17.2 Функции/подпрограммы

5.17.2.1 set_openmp_perfomance()

```
subroutine perfomance_settings::set_openmp_perfomance (
                integer num_of_omp_treads,
                integer N )
```

Настраивает количество OpenMP потоков и omp_chunk_size.

Аргументы

num_of_omp_treads	Количество OpenMP потоков
n	Количество частиц в моделируемой системе

См. определение в файле perfomance_settings.f90 строка 14

```
14    integer:: chunks_per_thread
15    integer:: num_of_omp_treads
16    integer:: N
17
18    call omp_set_num_threads(num_of_omp_treads)
19    !call omp_thread_limit(num_of_omp_treads)
20
21    chunks_per_thread = 8
22    omp_chunk_size = int(real(N)/chunks_per_thread/num_of_omp_treads)
23    if(omp_chunk_size==0) omp_chunk_size = 1
24
```

5.17.3 Переменные

5.17.3.1 omp_chunk_size

```
integer perfomance_settings::omp_chunk_size
```

Массивы данных будут разбиваться на части длины omp_chunk_size при использовании omp parallel for schedule(dynamic ,omp_chunk_size)

См. определение в файле perfomance_settings.f90 строка 8

```
8 integer:: omp_chunk_size
```

5.18 Модуль rebosolidcarbon

Типы данных

- type rebosc_parameters

Функции/подпрограммы

- subroutine read_rebosc_parameters (REBOscp, filename)
- subroutine rebosc_energy (energy, nl, REBOscp)

5.18.1 Функции/подпрограммы

5.18.1.1 read_rebosc_parameters()

```
subroutine rebosolidcarbon::read_rebosc_parameters (
                type(rebosc_parameters) REBOscp,
                character(*) filename )
```

См. определение в файле REBOsolidcarbon.f90 строка 13

```
13    type(REBOsc_parameters):: REBOscp
14    character(*):: filename
15
16    open(1,file=filename)
17    read(1,*) reboscp%A,reboscp%Q,reboscp%alpha
18    read(1,*) reboscp%B
19    read(1,*) reboscp%beta
20    read(1,*) reboscp%T
21    read(1,*) reboscp%g
22    read(1,*) reboscp%R1,reboscp%R2
23    close(1)
24
```

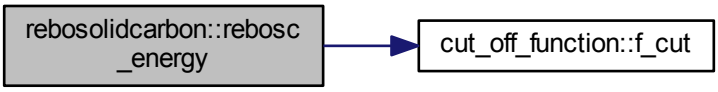
5.18.1.2 rebosc_energy()

```
subroutine rebosolidcarbon::rebosc_energy (
                real energy,
                type(neighbour_list) nl,
                type(rebosc_parameters) REBOscp )
```

См. определение в файле REBOsolidcarbon.f90 строка 28

```
28    type(neighbour_list):: nl
29    type(REBOsc_parameters):: REBOscp
30    integer:: i,p,q,l,j
31    real:: energy,energy_priv,bsp(nl%neighb_num_max,nl%N),bdh(nl%neighb_num_max,nl%N),aa,ab,ac,bc,cosine
32
33    energy = 0.
34    energy_priv = 0.
35    !$OMP PARALLEL firstprivate(energy_priv,i,p,q,l,j,aa,ab,ac,bc,cosine)
36    !$OMP DO
37    do i=1,nl%N
38       bsp(:,i) = 0. !not very optimal but hides a bug(?) in numerical forse calc in parallel
39       bdh(:,i) = 0.
40       do p=1,nl%nnum(i)
41          !bsp(p,i) = 0.
42          !bdh(p,i) = 0.
43          if (nl%moddr(p,i)<reboscp%R2) then
44             j = nl%nlist(p,i)
45             do q=1,nl%nnum(i)
46                if(p/=q .and. nl%moddr(q,i)<reboscp%R2) then
47                   cosine = sum(nl%dr(:,p,i)*nl%dr(:,q,i))/(nl%moddr(p,i)*nl%moddr(q,i))
48                   bsp(p,i) = bsp(p,i)+f_cut(nl%moddr(q,i),reboscp%R1,reboscp%R2)*(reboscp%g(1)+
   reboscp%g(2)*cosine+&
49                   reboscp%g(3)*cosine**2+reboscp%g(4)*cosine**3+reboscp%g(5)*cosine**4+reboscp%g(6)*
   cosine**5)
50                   do l=1,nl%nnum(j) !mb if j>i?
51                      if(nl%nlist(l,j)/=i .and. nl%moddr(l,j)<reboscp%R2) then
52                         aa = nl%moddr(p,i)**2
53                         ab = sum(nl%dr(:,p,i)*nl%dr(:,q,i))
54                         ac = sum(nl%dr(:,p,i)*nl%dr(:,l,j))
55                         bc = sum(nl%dr(:,q,i)*nl%dr(:,l,j))
56                         bdh(p,i) = bdh(p,i)+f_cut(nl%moddr(q,i),reboscp%R1,reboscp%R2)*f_cut(nl
   %moddr(l,j),reboscp%R1,reboscp%R2)*&
57                         (1.-(aa*bc-ab*ac)**2/(aa*nl%moddr(q,i)**2-ab**2)/(aa*nl%moddr(l,j)**2-ac**2
   ))
58                      endif
59                   enddo
60                endif
61             enddo
62             bsp(p,i) = (1.+bsp(p,i))**(-0.5)
63             bdh(p,i) = reboscp%T*bdh(p,i)
64          endif
65       enddo
66    enddo
67    !$OMP END DO
68    !$OMP DO
69    do i=1,nl%N
70       do p=1,nl%nnum(i)
71          if (nl%moddr(p,i)<reboscp%R2) then
72             j = nl%nlist(p,i)
73             if (j>i) then
74                do q=1,nl%nnum(j)
75                   if(nl%nlist(q,j)==i) exit
76                enddo
77                energy_priv = energy_priv+f_cut(nl%moddr(p,i),reboscp%R1,reboscp%R2)*&
78                ((1+reboscp%Q/nl%moddr(p,i))*reboscp%A*exp(-reboscp%alpha*nl%moddr(p,i))-&
79                ((bsp(p,i)+bsp(q,j))/2+bdh(p,i))*&
80                (reboscp%B(1)*exp(-reboscp%beta(1)*nl%moddr(p,i))+&
81                reboscp%B(2)*exp(-reboscp%beta(2)*nl%moddr(p,i))+&
82                reboscp%B(3)*exp(-reboscp%beta(3)*nl%moddr(p,i))))
83             endif
84          endif
85       enddo
86    enddo
87    !$OMP END DO
88    !$OMP ATOMIC
89       energy = energy+energy_priv
90    !$OMP END PARALLEL
91
```

Граф вызовов:



5.19 Модуль rosatoguillopelegrand

Типы данных

- type rosatoguillopelegrand_parameters

Функции/подпрограммы

- subroutine read_rjl_parameters (RJLp, filename)
- subroutine rjl_energy (energy, nl, RJLp)
- subroutine rjl_forces (atoms, nl, RJLp)

5.19.1 Функции/подпрограммы

5.19.1.1 read_rjl_parameters()

```
subroutine rosatoguillopelegrand::read_rjl_parameters (
                type(rosatoguillopelegrand_parameters) RJLp,
                character(*) filename )
```

См. определение в файле RosatoGuillopeLegrand.f90 строка 13

```
13    type(RosatoGuillopeLegrand_parameters):: RJLp
14    character(*):: filename
15
16    open(1,file=filename)
17    read(1,*) rjlp%A0,rjlp%xi,rjlp%p,rjlp%q,rjlp%r0
18    read(1,*) rjlp%R1,rjlp%R2
19    close(1)
20
```

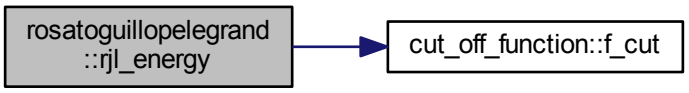
5.19.1.2 rjl_energy()

```
subroutine rosatoguillopelegrand::rjl_energy (
                real energy,
                type(neighbour_list) nl,
                type(rosatoguillopelegrand_parameters) RJLp )
```

См. определение в файле RosatoGuillopeLegrand.f90 строка 24

```
24    type(RosatoGuillopeLegrand_parameters):: RJLp
25    type(neighbour_list):: nl
26    integer:: i,p
27    real:: energy,energy_priv,Eb2,Er
28
29    energy = 0.
30    energy_priv = 0.
31    !$OMP PARALLEL firstprivate(energy_priv,i,p,Eb2,Er)
32    !$OMP DO
33    do i=1,nl%N
34       eb2 = 0.
35       er = 0.
36       do p=1,nl%nnum(i)
37          if( nl%moddr(p,i)<=rjlp%R2 ) then
38             eb2 = eb2+exp(-2.*rjlp%q*(nl%moddr(p,i)/rjlp%r0-1.))*f_cut(nl%moddr(p,i),rjlp%R1,rjlp%R2)
39             er = er+exp(-rjlp%p*(nl%moddr(p,i)/rjlp%r0-1.))*f_cut(nl%moddr(p,i),rjlp%R1,rjlp%R2)
40          endif
41       enddo
42       energy_priv = energy_priv+rjlp%A0*er-rjlp%xi*sqrt(eb2)
43    enddo
44    !$OMP END DO
45    !$OMP ATOMIC
46       energy = energy+energy_priv
47    !$OMP END PARALLEL
48
```

Граф вызовов:



5.19.1.3 rjl_forces()

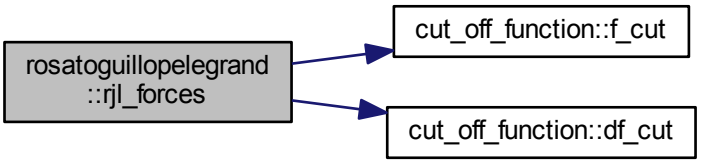
```
subroutine rosatoguillopelegrand::rjl_forces (
                type(particles) atoms,
                type(neighbour_list) nl,
                type(rosatoguillopelegrand_parameters) RJLp )
```

См. определение в файле RosatoGuillopeLegrand.f90 строка 52

```
52    type(RosatoGuillopeLegrand_parameters):: RJLp
53    type(particles)::  atoms
54    type(neighbour_list):: nl
55    integer:: i,p
56    real:: fc,dfc,Eb(nl%N),expp(nl%neighb_num_max,nl%N),expq(nl%neighb_num_max,nl%N)
57
58    !$OMP PARALLEL firstprivate(i,p,fc,dfc)
59    !$OMP DO
60    do i=1,nl%N
61       do p=1,nl%nnum(i)
62          if( nl%moddr(p,i)<=rjlp%R2 ) then
63             expq(p,i) = exp(-2.*rjlp%q*(nl%moddr(p,i)/rjlp%r0-1.))
64             expp(p,i) = exp(-rjlp%p*(nl%moddr(p,i)/rjlp%r0-1.))
65          endif
66       enddo
67    enddo
68    !$OMP END DO
69    !$OMP DO
```

```
70     do i=1,nl%N
71        eb(i) = 0.
72        do p=1,nl%nnum(i)
73           if( nl%moddr(p,i)<=rjlp%R2 ) then
74              eb(i) = eb(i)+expq(p,i)*f_cut(nl%moddr(p,i),rjlp%R1,rjlp%R2)
75           endif
76        enddo
77        eb(i) = sqrt(eb(i))
78     enddo
79     !$OMP END DO
80     !$OMP DO
81     do i=1,nl%N
82        do p=1,nl%nnum(i)
83           if( nl%moddr(p,i)<=rjlp%R2 ) then
84              fc = f_cut(nl%moddr(p,i),rjlp%R1,rjlp%R2)
85              dfc = df_cut(nl%moddr(p,i),rjlp%R1,rjlp%R2)
86              atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))&
87              -(2.*rjlp%A0*(rjlp%p/rjlp%r0*fc-dfc*nl%moddr(p,i))*expp(p,i)&
88              -rjlp%xi*(rjlp%q/rjlp%r0*fc-dfc/2.*nl%moddr(p,i))*(1./eb(i)+1./eb(nl%nlist(p,i)))*expq(p,i)
    )/nl%moddr(p,i)*nl%dr(:,p,i)
89           endif
90        enddo
91     enddo
92     !$OMP END DO
93     !$OMP END PARALLEL
```

Граф вызовов:



5.20 Модуль tersoffbrenner

Типы данных

- type tersoffbrenner_parameters

Функции/подпрограммы

- subroutine read_tb_parameters (TBp, filename)
- subroutine tb_energy (energy, nl, TBp)
- subroutine tb_forces (atoms, nl, TBp)

5.20.1 Функции/подпрограммы

5.20.1.1 read_tb_parameters()

```
subroutine tersoffbrenner::read_tb_parameters (
                type(tersoffbrenner_parameters) TBp,
                character(*) filename )
```

См. определение в файле TersoffBrenner.f90 строка 14

```
14    type(TersoffBrenner_parameters):: TBp
15    character(*):: filename
16
17    open(1,file=filename)
18    read(1,*) tbp%d,tbp%s,tbp%b,tbp%r0,tbp%delt,tbp%a0,tbp%c0,tbp%d0
19    read(1,*) tbp%R1,tbp%R2
20    close(1)
21    tbp%c02 = tbp%c0**2
22    tbp%d02 = tbp%d0**2
23
```

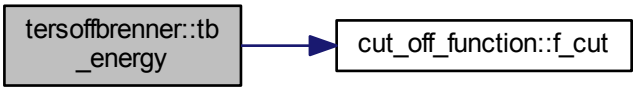
5.20.1.2 tb_energy()

```
subroutine tersoffbrenner::tb_energy (
                real energy,
                type(neighbour_list) nl,
                type(tersoffbrenner_parameters) TBp )
```

См. определение в файле TersoffBrenner.f90 строка 27

```
27    type(neighbour_list):: nl
28    type(TersoffBrenner_parameters):: TBp
29    integer:: i,p,j,q
30    real:: energy,energy_priv,B(nl%neighb_num_max,nl%N),a
31
32    energy = 0.
33    energy_priv = 0.
34    !$OMP PARALLEL firstprivate(energy_priv,i,p,j,q,a)
35    !$OMP DO
36    do i=1,nl%N
37       b(:,i) = 0. !look rebosc
38       do p=1,nl%nnum(i)
39          !B(p,i) = 0.
40          if (nl%moddr(p,i)<tbp%R2) then
41             do q=1,nl%nnum(i)
42                if(p/=q .AND. nl%moddr(q,i)<tbp%R2) then
43                   b(p,i) = b(p,i)+f_cut(nl%moddr(q,i),tbp%R1,tbp%R2)*&
44                   (1.+tbp%c02/tbp%d02-tbp%c02/(tbp%d02+(1.+sum(nl%dr(:,p,i)*nl%dr(:,q,i))/(nl%moddr(p
    ,i)*nl%moddr(q,i)))**2))
45                endif
46             enddo
47             b(p,i) = (1.+tbp%a0*b(p,i))**(-tbp%delt)
48          endif
49       enddo
50    enddo
51    !$OMP END DO
52    !$OMP DO
53    do i=1,nl%N
54       do p=1,nl%nnum(i)
55          if (nl%moddr(p,i)<tbp%R2) then
56             j = nl%nlist(p,i)
57             if (j>i) then
58                do q=1,nl%nnum(j)
59                   if(nl%nlist(q,j)==i) exit
60                enddo
61                a = -sqrt(2.*tbp%s)*tbp%b*(nl%moddr(p,i)-tbp%r0)
62                energy_priv = energy_priv+f_cut(nl%moddr(p,i),tbp%R1,tbp%R2)*tbp%d/(tbp%s-1.)*(exp(a)-(
    b(p,i)+b(q,j))/2*tbp%s*exp(a/tbp%s))
63             endif
64          endif
65       enddo
66    enddo
67    !$OMP END DO
68    !$OMP ATOMIC
69       energy = energy+energy_priv
70    !$OMP END PARALLEL
71
```

Граф вызовов:



5.20.1.3 tb_forces()

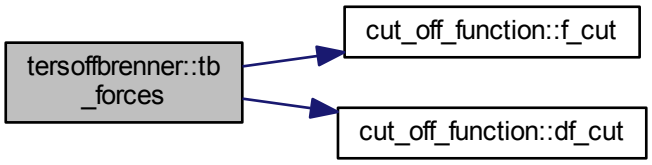
subroutine tersoffbrenner::tb_forces (
type(particles) atoms,
type(neighbour_list) nl,
type(tersoffbrenner_parameters) TBp)

См. определение в файле TersoffBrenner.f90 строка 75

```
75    type(particles)::   atoms
76    type(neighbour_list):: nl
77    type(TersoffBrenner_parameters):: TBp
78    integer:: i,p,j,q,l
79    real:: B(nl%neighb_num_max,nl%N),dB(3),a,dff,rr,cosi,f_c,df_c
80
81    !$OMP PARALLEL firstprivate(i,p,j,q,l,a,dff,rr,cosi,f_c,df_c,dB)
82    !$OMP DO
83    do i=1,nl%N
84       do p=1,nl%nnum(i)
85          b(p,i) = 0.
86          if (nl%moddr(p,i)<tbp%R2) then
87             do q=1,nl%nnum(i)
88                if(p/=q .AND. nl%moddr(q,i)<tbp%R2) then
89                   b(p,i) = b(p,i)+f_cut(nl%moddr(q,i),tbp%R1,tbp%R2)*&
90                   (1.+tbp%c02/tbp%d02-tbp%c02/(tbp%d02+(1.+sum(nl%dr(:,p,i)*nl%dr(:,q,i))/(nl%moddr(p
     ,i)*nl%moddr(q,i)))**2))
91                endif
92             enddo
93             b(p,i) = (1.+tbp%a0*b(p,i))**(-tbp%delt)
94          endif
95       enddo
96    enddo
97    !$OMP END DO
98    !$OMP DO
99    do i=1,nl%N
100      do p=1,nl%nnum(i)
101         db=0.d0
102         do q=1,nl%nnum(i)
103            if (p/=q) then
104               rr = 1./nl%moddr(p,i)/nl%moddr(q,i)
105               cosi = sum(nl%dr(:,p,i)*nl%dr(:,q,i))*rr
106               db = db+ f_cut(nl%moddr(q,i),tbp%R1,tbp%R2)*2.d0*tbp%a0*tbp%c02*(1.+cosi)/(tbp%d02+(1.+
     cosi)**2)**2*&
107                  ( (nl%dr(:,p,i)+nl%dr(:,q,i))*rr-cosi*(nl%dr(:,p,i)/nl%moddr(p,i)**2+nl%dr(:,q,i)/
     nl%moddr(q,i)**2) )&
108                  + df_cut(nl%moddr(q,i),tbp%R1,tbp%R2)*tbp%a0*(1.+tbp%c02/tbp%d02-tbp%c02/(tbp%d02+(
     1.+cosi)**2))*nl%dr(:,q,i)
109            endif
110         enddo
111         db=db*b(p,i)**(1./tbp%delt+1.)
112         j=nl%nlist(p,i)
113         do l=1,nl%nnum(j)
114            if(nl%nlist(l,j)==i) exit
115         enddo
116         do q=1,nl%nnum(j)
```

```
117                 if (q/=l) then
118                     rr = 1./nl%moddr(l,j)/nl%moddr(q,j)
119                     cosi = sum(nl%dr(:,l,j)*nl%dr(:,q,j))*rr
120                     db = db+(b(l,j)**(1./tbp%delt+1.)*f_cut(nl%moddr(q,j),tbp%R1,tbp%R2)*&
121                         2.*tbp%a0*tbp%c02*(1.+cosi)/(tbp%d02+(1.+cosi)**2)**2)*(-nl%dr(:,q,j)*rr + cosi*nl
    %dr(:,l,j)/nl%moddr(l,j)**2)
122                 endif
123             enddo
124             db = db*(-tbp%delt)/2.
125             if (nl%moddr(p,i)<tbp%R2) then
126                 f_c = f_cut(nl%moddr(p,i),tbp%R1,tbp%R2)
127                 dff = df_cut(nl%moddr(p,i),tbp%R1,tbp%R2)/f_c
128                 a = -sqrt(2.*tbp%s)*tbp%b*(nl%moddr(p,i)-tbp%r0)
129                 atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))+&
130                 f_c*tbp%d/(tbp%s-1.)*((nl%dr(:,p,i)*dff-sqrt(2.*tbp%s)*tbp%b/nl%moddr(p,i)*nl%dr(:,p,i))*
    exp(a)&
131                 -( db+(b(p,i)+b(l,j))/2*(nl%dr(:,p,i)*dff-sqrt(2./tbp%s)*tbp%b/nl%moddr(p,i)*nl%dr(:,p,i))
    )*tbp%s*exp(a/tbp%s))
132             endif
133             do q=1,nl%nnum(j)
134                 if (q/=l) then
135                     f_c = f_cut(nl%moddr(l,j),tbp%R1,tbp%R2)
136                     df_c = df_cut(nl%moddr(l,j),tbp%R1,tbp%R2)
137                     rr = 1./nl%moddr(l,j)/nl%moddr(q,j)
138                     cosi = sum(nl%dr(:,l,j)*nl%dr(:,q,j))*rr
139                     atoms%forces(:,nl%particle_index(i)) = atoms%forces(:,nl%particle_index(i))+tbp%delt/2*
    b(q,j)**(1./tbp%delt+1.)*&
140                     ( f_c*2.*tbp%a0*tbp%c02*(1.+cosi)/(tbp%d02+(1.+cosi)**2)**2 *( -nl%dr(:,q,j)*rr + cosi*
    nl%dr(:,l,j)/nl%moddr(l,j)**2 )+&
141                     nl%dr(:,p,i)*df_c*tbp%a0*(1.+tbp%c02/tbp%d02-tbp%c02/(tbp%d02+(1.+cosi)**2)) )*&
142                     f_cut(nl%moddr(q,j),tbp%R1,tbp%R2)*tbp%d/(tbp%s-1.)*tbp%s*exp(-sqrt(2.*tbp%s)*tbp%b*(nl
    %moddr(q,j)-tbp%r0)/tbp%s)
143                 endif
144             enddo
145         enddo
146     enddo
147     !$OMP END DO
148     !$OMP END PARALLEL
149
```

Граф вызовов:



Глава 6

Оглавление типов данных

6.1 md_general::integrator_params Шаблон типа

Открытые атрибуты

- integer l
- integer period_snapshot
- integer period_log
- real dt
- character(len=32) int_name

6.1.1 Подробное описание

См. определение в файле md_general.f90 строка 37

6.1.2 Данные класса

6.1.2.1 dt

real md_general::integrator_params::dt

См. определение в файле md_general.f90 строка 39

```
39     real:: dt
```

6.1.2.2 int_name

character(len=32) md_general::integrator_params::int_name

См. определение в файле md_general.f90 строка 40

```
40     character(len=32):: int_name
```

6.1.2.3 l

integer md_general::integrator_params::l

См. определение в файле md_general.f90 строка 38

```
38     integer:: l,period_snapshot,period_log
```

6.1.2.4 period_log

integer md_general::integrator_params::period_log

См. определение в файле md_general.f90 строка 38

6.1.2.5 period_snapshot

integer md_general::integrator_params::period_snapshot

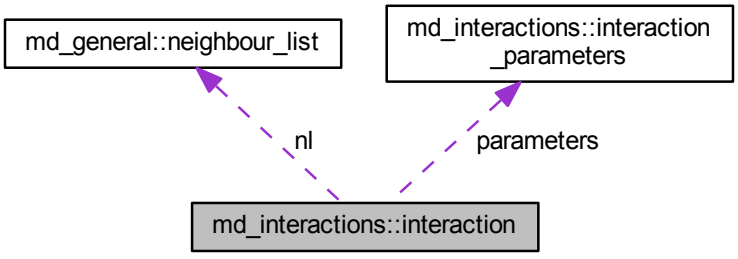
См. определение в файле md_general.f90 строка 38

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.2 md_interactions::interaction Шаблон типа

Граф связей класса md_interactions::interaction:



Открытые атрибуты

- integer nl_n
- integer neib_order
- integer, dimension(:), allocatable group_nums
- type(neighbour_list), dimension(:), allocatable nl
- real energy
- character(len=32) interaction_name
- character(len=32) parameters_file
- type(interaction_parameters) parameters
- logical numerical_force

6.2.1 Подробное описание

См. определение в файле md_interactions.f90 строка 26

6.2.2 Данные класса

6.2.2.1 energy

real md_interactions::interaction::energy

См. определение в файле md_interactions.f90 строка 30

```
30    real:: energy
```

6.2.2.2 group_nums

integer, dimension(:), allocatable md_interactions::interaction::group_nums

См. определение в файле md_interactions.f90 строка 28

```
28    integer,allocatable:: group_nums(:)
```

6.2.2.3 interaction_name

character(len=32) md_interactions::interaction::interaction_name

См. определение в файле md_interactions.f90 строка 31

```
31    character(len=32):: interaction_name,parameters_file
```

6.2.2.4 neib_order

integer md_interactions::interaction::neib_order

См. определение в файле md_interactions.f90 строка 27

6.2.2.5 nl

type(neighbour_list), dimension(:), allocatable md_interactions::interaction::nl

См. определение в файле md_interactions.f90 строка 29

```
29    type(neighbour_list),allocatable:: nl(:)
```

6.2.2.6 nl_n

integer md_interactions::interaction::nl_n

См. определение в файле md_interactions.f90 строка 27

```
27    integer:: nl_n,neib_order
```

6.2.2.7 numerical_force

logical md_interactions::interaction::numerical_force

См. определение в файле md_interactions.f90 строка 33

```
33    logical:: numerical_force
```

6.2.2.8 parameters

type(interaction_parameters) md_interactions::interaction::parameters

См. определение в файле md_interactions.f90 строка 32

```
32    type(interaction_parameters) :: parameters
```

6.2.2.9 parameters_file

character(len=32) md_interactions::interaction::parameters_file

См. определение в файле md_interactions.f90 строка 31

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_interactions.f90

6.3 md_interactions::interaction_parameters Шаблон типа

Открытые атрибуты

- type(lennardjones_parameters), dimension(:), allocatable lj
- type(lennardjones1g_parameters), dimension(:), allocatable lj1g
- type(lennardjonescosine_parameters), dimension(:), allocatable ljc
- type(morsecosine_parameters), dimension(:), allocatable morsec
- type(rosatoguillopelegrand_parameters), dimension(:), allocatable rjl
- type(tersoffbrenner_parameters), dimension(:), allocatable tb
- type(rebosc_parameters), dimension(:), allocatable rebosc

6.3.1 Подробное описание

См. определение в файле md_interactions.f90 строка 15

6.3.2 Данные класса

6.3.2.1 lj

type(lennardjones_parameters), dimension(:), allocatable md_interactions::interaction_parameters::lj

См. определение в файле md_interactions.f90 строка 16

```
16    type(LennardJones_parameters),allocatable          :: LJ(:)
```

6.3.2.2 lj1g

type(lennardjones1g_parameters), dimension(:), allocatable md_interactions::interaction_parameters::lj1g

См. определение в файле md_interactions.f90 строка 17

```
17    type(LennardJones1g_parameters),allocatable        :: LJ1g(:)
```

6.3.2.3 ljc

type(lennardjonescosine_parameters), dimension(:), allocatable md_interactions::interaction_parameters::ljc

См. определение в файле md_interactions.f90 строка 19

```
19    type(LennardJonesCosine_parameters),allocatable     :: LJC(:)
```

6.3.2.4 morsec

type(morsecosine_parameters), dimension(:), allocatable md_interactions::interaction_parameters::morsec

См. определение в файле md_interactions.f90 строка 20

```
20    type(MorseCosine_parameters),allocatable          :: MorseC(:)
```

6.3.2.5 rebosc

type(rebosc_parameters), dimension(:), allocatable md_interactions::interaction_parameters::rebosc

См. определение в файле md_interactions.f90 строка 23

```
23    type(REBOsc_parameters),allocatable               :: REBOsc(:)
```

6.3.2.6 rjl

type(rosatoguillopelegrand_parameters), dimension(:), allocatable md_interactions::interaction_parameters::rjl

См. определение в файле md_interactions.f90 строка 21

```
21    type(RosatoGuillopeLegrand_parameters),allocatable  :: RJL(:)
```

6.3.2.7 tb

type(tersoffbrenner_parameters), dimension(:), allocatable md_interactions::interaction_parameters::tb

См. определение в файле md_interactions.f90 строка 22

```
22    type(TersoffBrenner_parameters),allocatable       :: TB(:)
```

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_interactions.f90

6.4 lennardjones_1g::lennardjones1g_parameters Шаблон типа

Открытые атрибуты

- real eps
- real sig
- real r1
- real r2
- real c6
- real c12
- real c6t6
- real c12t12

6.4.1 Подробное описание

См. определение в файле LennardJones_1g.f90 строка 7

6.4.2 Данные класса

6.4.2.1 c12

real lennardjones_1g::lennardjones1g_parameters::c12

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.2 c12t12

real lennardjones_1g::lennardjones1g_parameters::c12t12

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.3 c6

real lennardjones_1g::lennardjones1g_parameters::c6

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.4 c6t6

real lennardjones_1g::lennardjones1g_parameters::c6t6

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.5 eps

real lennardjones_1g::lennardjones1g_parameters::eps

См. определение в файле LennardJones_1g.f90 строка 8

```
8     real eps,sig,R1,R2,c6,c12,c6t6,c12t12
```

6.4.2.6 r1

real lennardjones_1g::lennardjones1g_parameters::r1

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.7 r2

real lennardjones_1g::lennardjones1g_parameters::r2

См. определение в файле LennardJones_1g.f90 строка 8

6.4.2.8 sig

real lennardjones_1g::lennardjones1g_parameters::sig

См. определение в файле LennardJones_1g.f90 строка 8

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/LennardJones_1g.f90

6.5 lennardjones::lennardjones_parameters Шаблон типа

Открытые атрибуты

- real eps
- real sig
- real r1
- real r2

6.5.1 Подробное описание

См. определение в файле LennardJones.f90 строка 6

6.5.2 Данные класса

6.5.2.1 eps

real lennardjones::lennardjones_parameters::eps

См. определение в файле LennardJones.f90 строка 7

```
7     real eps,sig,R1,R2
```

6.5.2.2 r1

real lennardjones::lennardjones_parameters::r1

См. определение в файле LennardJones.f90 строка 7

6.5.2.3 r2

real lennardjones::lennardjones_parameters::r2

См. определение в файле LennardJones.f90 строка 7

6.5.2.4 sig

real lennardjones::lennardjones_parameters::sig

См. определение в файле LennardJones.f90 строка 7

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/LennardJones.f90

6.6 lennardjonescosine::lennardjonescosine_parameters Шаблон типа

Открытые атрибуты

- real eps
- real sig
- real delt
- real r1
- real r2
- logical simplified
- real, dimension(:,:), allocatable gr_norm

6.6.1 Подробное описание

См. определение в файле LennardJonesCosine.f90 строка 6

6.6.2 Данные класса

6.6.2.1 delt

real lennardjonescosine::lennardjonescosine_parameters::delt

См. определение в файле LennardJonesCosine.f90 строка 7

6.6.2.2 eps

real lennardjonescosine::lennardjonescosine_parameters::eps

См. определение в файле LennardJonesCosine.f90 строка 7

```
7    real eps,sig,delt,R1,R2
```

6.6.2.3 gr_norm

real, dimension(:,:), allocatable lennardjonescosine::lennardjonescosine_parameters::gr_norm

См. определение в файле LennardJonesCosine.f90 строка 9

```
9     real,allocatable:: gr_norm(:,:)
```

6.6.2.4 r1

real lennardjonescosine::lennardjonescosine_parameters::r1

См. определение в файле LennardJonesCosine.f90 строка 7

6.6.2.5 r2

real lennardjonescosine::lennardjonescosine_parameters::r2

См. определение в файле LennardJonesCosine.f90 строка 7

6.6.2.6 sig

real lennardjonescosine::lennardjonescosine_parameters::sig

См. определение в файле LennardJonesCosine.f90 строка 7

6.6.2.7 simplified

logical lennardjonescosine::lennardjonescosine_parameters::simplified

См. определение в файле LennardJonesCosine.f90 строка 8

```
8     logical:: simplified
```

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/LennardJonesCosine.f90

6.7 morsecosine::morsecosine_parameters Шаблон типа

Открытые атрибуты

- real d
- real r
- real a
- real delt
- real r1
- real r2
- logical simplified
- real, dimension(:,:), allocatable gr_norm

6.7.1 Подробное описание

См. определение в файле MorseCosine.f90 строка 6

6.7.2 Данные класса

6.7.2.1 a

real morsecosine::morsecosine_parameters::a

См. определение в файле MorseCosine.f90 строка 7

6.7.2.2 d

real morsecosine::morsecosine_parameters::d

См. определение в файле MorseCosine.f90 строка 7

```
7     real:: d,r,a,delt,R1,R2
```

6.7.2.3 delt

real morsecosine::morsecosine_parameters::delt

См. определение в файле MorseCosine.f90 строка 7

6.7.2.4 gr_norm

real, dimension(:,:), allocatable morsecosine::morsecosine_parameters::gr_norm

См. определение в файле MorseCosine.f90 строка 9

```
9     real,allocatable:: gr_norm(:,:)
```

6.7.2.5 r

real morsecosine::morsecosine_parameters::r

См. определение в файле MorseCosine.f90 строка 7

6.7.2.6 r1

real morsecosine::morsecosine_parameters::r1

См. определение в файле MorseCosine.f90 строка 7

6.7.2.7 r2

real morsecosine::morsecosine_parameters::r2

См. определение в файле MorseCosine.f90 строка 7

6.7.2.8 simplified

logical morsecosine::morsecosine_parameters::simplified

См. определение в файле MorseCosine.f90 строка 8

```
8     logical:: simplified
```

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/MorseCosine.f90

6.8 md_general::neighbour_list Шаблон типа

Открытые атрибуты

- integer n
- integer neighb_num_max
- integer update_period
- real r_cut
- integer, dimension(:,:), allocatable nlist
- integer, dimension(:), allocatable nnum
- integer, dimension(:), allocatable lessnnum
- integer, dimension(:), allocatable particle_index
- real, dimension(:,:,:), allocatable dr
- real, dimension(:,:), allocatable moddr

6.8.1 Подробное описание

См. определение в файле md_general.f90 строка 30

6.8.2 Данные класса

6.8.2.1 dr

real, dimension(:,:,:), allocatable md_general::neighbour_list::dr

См. определение в файле md_general.f90 строка 34

```
34     real,allocatable:: dr(:,:,:),moddr(:,:)
```

6.8.2.2 lessnnum

integer, dimension(:), allocatable md_general::neighbour_list::lessnnum

См. определение в файле md_general.f90 строка 33

6.8.2.3 moddr

real, dimension(:,:), allocatable md_general::neighbour_list::moddr

См. определение в файле md_general.f90 строка 34

6.8.2.4 n

integer md_general::neighbour_list::n

См. определение в файле md_general.f90 строка 31

```
31    integer:: N,neighb_num_max,update_period
```

6.8.2.5 neighb_num_max

integer md_general::neighbour_list::neighb_num_max

См. определение в файле md_general.f90 строка 31

6.8.2.6 nlist

integer, dimension(:,:), allocatable md_general::neighbour_list::nlist

См. определение в файле md_general.f90 строка 33

```
33    integer,allocatable:: nlist(:,:),nnum(:),lessnnum(:),particle_index(:)
```

6.8.2.7 nnum

integer, dimension(:), allocatable md_general::neighbour_list::nnum

См. определение в файле md_general.f90 строка 33

6.8.2.8 particle_index

integer, dimension(:), allocatable md_general::neighbour_list::particle_index

См. определение в файле md_general.f90 строка 33

6.8.2.9 r_cut

real md_general::neighbour_list::r_cut

См. определение в файле md_general.f90 строка 32

```
32     real:: r_cut
```

6.8.2.10 update_period

integer md_general::neighbour_list::update_period

См. определение в файле md_general.f90 строка 31

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.9 md_general::nose_hoover_chain Шаблон типа

Открытые атрибуты

- integer m
- integer l
- real, dimension(:), allocatable x
- real, dimension(:), allocatable v
- real, dimension(:), allocatable q
- real temperature
- real s
- real e

6.9.1 Подробное описание

См. определение в файле md_general.f90 строка 24

6.9.2 Данные класса

6.9.2.1 e

real md_general::nose_hoover_chain::e

См. определение в файле md_general.f90 строка 27

6.9.2.2 l

integer md_general::nose_hoover_chain::l

См. определение в файле md_general.f90 строка 25

6.9.2.3 m

integer md_general::nose_hoover_chain::m

См. определение в файле md_general.f90 строка 25

```
25    integer:: M,L
```

6.9.2.4 q

real, dimension(:), allocatable md_general::nose_hoover_chain::q

См. определение в файле md_general.f90 строка 26

6.9.2.5 s

real md_general::nose_hoover_chain::s

См. определение в файле md_general.f90 строка 27

6.9.2.6 temperature

real md_general::nose_hoover_chain::temperature

См. определение в файле md_general.f90 строка 27

```
27    real:: temperature,s,e
```

6.9.2.7 v

real, dimension(:), allocatable md_general::nose_hoover_chain::v

См. определение в файле md_general.f90 строка 26

6.9.2.8 x

real, dimension(:), allocatable md_general::nose_hoover_chain::x

См. определение в файле md_general.f90 строка 26

```
26     real,allocatable:: x(:),v(:),q(:)
```

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.10 md_general::particle_group Шаблон типа

Открытые атрибуты

- integer n
- integer, dimension(:), allocatable indexes

6.10.1 Подробное описание

См. определение в файле md_general.f90 строка 19

6.10.2 Данные класса

6.10.2.1 indexes

integer, dimension(:), allocatable md_general::particle_group::indexes

См. определение в файле md_general.f90 строка 21

```
21     integer,allocatable:: indexes(:)
```

6.10.2.2 n

integer md_general::particle_group::n

См. определение в файле md_general.f90 строка 20

```
20    integer:: N
```

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.11 md_general::particles Шаблон типа

Открытые атрибуты

- integer n
- real, dimension(:,:), allocatable positions
- real, dimension(:,:), allocatable velocities
- real, dimension(:), allocatable masses
- real, dimension(:,:), allocatable forces
- character(len=32), dimension(:), allocatable atom_types

6.11.1 Подробное описание

См. определение в файле md_general.f90 строка 13

6.11.2 Данные класса

6.11.2.1 atom_types

character(len=32), dimension(:), allocatable md_general::particles::atom_types

См. определение в файле md_general.f90 строка 16

```
16    character(len=32),allocatable:: atom_types(:)
```

6.11.2.2 forces

real, dimension(:,:), allocatable md_general::particles::forces

См. определение в файле md_general.f90 строка 15

6.11.2.3 masses

real, dimension(:), allocatable md_general::particles::masses

См. определение в файле md_general.f90 строка 15

6.11.2.4 n

integer md_general::particles::n

См. определение в файле md_general.f90 строка 14

```
14    integer:: N
```

6.11.2.5 positions

real, dimension(:,:), allocatable md_general::particles::positions

См. определение в файле md_general.f90 строка 15

```
15    real,allocatable::  positions(:,:),velocities(:,:),masses(:),forces(:,:)
```

6.11.2.6 velocities

real, dimension(:,:), allocatable md_general::particles::velocities

См. определение в файле md_general.f90 строка 15

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.12 rebosolidcarbon::rebosc_parameters Шаблон типа

Открытые атрибуты

- real a
- real q
- real alpha
- real, dimension(3) b
- real, dimension(3) beta
- real t
- real, dimension(6) g
- real r1
- real r2

6.12.1 Подробное описание

См. определение в файле REBOsolidcarbon.f90 строка 6

6.12.2 Данные класса

6.12.2.1 a

real rebosolidcarbon::rebosc_parameters::a

См. определение в файле REBOsolidcarbon.f90 строка 7

```
7     real A,Q,alpha,B(3),beta(3),T,g(6),R1,R2
```

6.12.2.2 alpha

real rebosolidcarbon::rebosc_parameters::alpha

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.3 b

real, dimension(3) rebosolidcarbon::rebosc_parameters::b

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.4 beta

real, dimension(3) rebosolidcarbon::rebosc_parameters::beta

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.5 g

real, dimension(6) rebosolidcarbon::rebosc_parameters::g

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.6 q

real rebosolidcarbon::rebosc_parameters::q

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.7 r1

real rebosolidcarbon::rebosc_parameters::r1

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.8 r2

real rebosolidcarbon::rebosc_parameters::r2

См. определение в файле REBOsolidcarbon.f90 строка 7

6.12.2.9 t

real rebosolidcarbon::rebosc_parameters::t

См. определение в файле REBOsolidcarbon.f90 строка 7

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/REBOsolidcarbon.f90

6.13 rosatoguillopelegrand::rosatoguillopelegrand_parameters Шаблон типа

Открытые атрибуты

- real a0
- real xi
- real p
- real q
- real r0
- real r1
- real r2

6.13.1 Подробное описание

См. определение в файле RosatoGuillopeLegrand.f90 строка 6

6.13.2 Данные класса

6.13.2.1 a0

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::a0

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

```
7     real A0,xi,p,q,r0,R1,R2
```

6.13.2.2 p

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::p

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

6.13.2.3 q

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::q

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

6.13.2.4 r0

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::r0

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

6.13.2.5 r1

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::r1

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

6.13.2.6 r2

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::r2

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

6.13.2.7 xi

real rosatoguillopelegrand::rosatoguillopelegrand_parameters::xi

См. определение в файле RosatoGuillopeLegrand.f90 строка 7

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/RosatoGuillopeLegrand.f90

6.14 md_general::simulation_cell Шаблон типа

Открытые атрибуты

- real, dimension(3) box_size
- real, dimension(3) half_box_size

6.14.1 Подробное описание

См. определение в файле md_general.f90 строка 9

6.14.2 Данные класса

6.14.2.1 box_size

real, dimension(3) md_general::simulation_cell::box_size

См. определение в файле md_general.f90 строка 10

```
10    real:: box_size(3),half_box_size(3)
```

6.14.2.2 half_box_size

real, dimension(3) md_general::simulation_cell::half_box_size

См. определение в файле md_general.f90 строка 10

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

6.15 tersoffbrenner::tersoffbrenner_parameters Шаблон типа

Открытые атрибуты

- real d
- real s
- real b
- real r0
- real delt
- real a0
- real c0
- real d0
- real r1
- real r2
- real c02
- real d02

6.15.1 Подробное описание

См. определение в файле TersoffBrenner.f90 строка 6

6.15.2 Данные класса

6.15.2.1 a0

real tersoffbrenner::tersoffbrenner_parameters::a0

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.2 b

real tersoffbrenner::tersoffbrenner_parameters::b

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.3 c0

real tersoffbrenner::tersoffbrenner_parameters::c0

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.4 c02

real tersoffbrenner::tersoffbrenner_parameters::c02

См. определение в файле TersoffBrenner.f90 строка 8

```
8     real c02,d02
```

6.15.2.5 d

real tersoffbrenner::tersoffbrenner_parameters::d

См. определение в файле TersoffBrenner.f90 строка 7

```
7     real d,s,b,r0,delt,a0,c0,d0,R1,R2
```

6.15.2.6 d0

real tersoffbrenner::tersoffbrenner_parameters::d0

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.7 d02

real tersoffbrenner::tersoffbrenner_parameters::d02

См. определение в файле TersoffBrenner.f90 строка 8

6.15.2.8 delt

real tersoffbrenner::tersoffbrenner_parameters::delt

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.9 r0

real tersoffbrenner::tersoffbrenner_parameters::r0

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.10 r1

real tersoffbrenner::tersoffbrenner_parameters::r1

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.11 r2

real tersoffbrenner::tersoffbrenner_parameters::r2

См. определение в файле TersoffBrenner.f90 строка 7

6.15.2.12 s

real tersoffbrenner::tersoffbrenner_parameters::s

См. определение в файле TersoffBrenner.f90 строка 7

Документация по типу сгенерирована на основе следующего файла:

- INTERACTION_POTENTIALS/TersoffBrenner.f90

6.16 md_general::time_steps Шаблон типа

Открытые атрибуты

- real, dimension(4) ts
- real simulation_time

6.16.1 Подробное описание

См. определение в файле md_general.f90 строка 5

6.16.2 Данные класса

6.16.2.1 simulation_time

real md_general::time_steps::simulation_time

См. определение в файле md_general.f90 строка 6

6.16.2.2 ts

real, dimension(4) md_general::time_steps::ts

См. определение в файле md_general.f90 строка 6

```
6     real    ts(4),simulation_time
```

Документация по типу сгенерирована на основе следующего файла:

- MOLECULAR_DYNAMICS/md_general.f90

Глава 7

Файлы

7.1 Файл graphene_on_surface_analysis/graphene_on_surface_analysis.f90

Группы

- module graphene_on_surface_analysis

Функции/подпрограммы

- subroutine graphene_on_surface_analysis::gr_on_cu_analysis (arr1, arr2, filename, z)

7.2 Файл INTERACTION_POTENTIALS/cut_off_function.f90

Группы

- module cut_off_function

Функции/подпрограммы

- real function cut_off_function::f_cut (r, R1, R2)
- real function cut_off_function::df_cut (r, R1, R2)

7.3 Файл INTERACTION_POTENTIALS/cut_off_poly.f90

Группы

- module cut_off_poly

Функции/подпрограммы

- real function cut_off_poly::f_cut (r, R1, R2)
- real function cut_off_poly::df_cut (r, R1, R2)
- pure subroutine cut_off_poly::f_dfr_cut (f, dfr, r, R1, R2)

7.4 Файл INTERACTION_POTENTIALS/graphenenorm.f90

Группы

- module graphenenorm

Функции/подпрограммы

- subroutine graphenenorm::find_gr_nearest_neighbors (nl_nn, nl)
- subroutine graphenenorm::find_norm_in_graphene (gr_norm, dr_nn)
- subroutine graphenenorm::update_nearest_neighbours_in_graphene (md_step, nl_nn, nl, atoms, group, box)

7.5 Файл INTERACTION_POTENTIALS/LennardJones.f90

Типы данных

- type lennardjones::lennardjones_parameters

Группы

- module lennardjones

Функции/подпрограммы

- subroutine lennardjones::read_lj_parameters (LJp, filename)
- subroutine lennardjones::lj_energy (energy, nl, LJp)
- subroutine lennardjones::lj_forces (atoms, nl, LJp)

7.6 Файл INTERACTION_POTENTIALS/LennardJones_1g.f90

Типы данных

- type lennardjones_1g::lennardjones1g_parameters

Группы

- module lennardjones_1g

Функции/подпрограммы

- subroutine lennardjones_1g::read_lj1g_parameters (LJp, filename)
- subroutine lennardjones_1g::lj1g_energy (energy, nl, LJp)
- subroutine lennardjones_1g::lj1g_forces (atoms, nl, LJp)
- real function lennardjones_1g::scalar_lj_force (r, R1, R2, c12, c6, c12t12, c6t6)

7.7 Файл INTERACTION_POTENTIALS/LennardJonesCosine.f90

Типы данных

- type lennardjonescosine::lennardjonescosine_parameters

Группы

- module lennardjonescosine

Функции/подпрограммы

- subroutine lennardjonescosine::read_ljc_parameters (LJCp, filename)
- subroutine lennardjonescosine::ljc_energy (energy, nl, LJCp)
- subroutine lennardjonescosine::ljc_forces_for_graphene (atoms, nl, nl_nn, LJCp)
- subroutine lennardjonescosine::ljc_forces_for_other_atoms (atoms, nl, LJCp)

7.8 Файл INTERACTION_POTENTIALS/MorseCosine.f90

Типы данных

- type morsecosine::morsecosine_parameters

Группы

- module morsecosine

Функции/подпрограммы

- subroutine morsecosine::read_morsec_parameters (MorseCp, filename)
- subroutine morsecosine::morsec_energy (energy, nl, MorseCp)
- subroutine morsecosine::morsec_forces_for_graphene (atoms, nl, nl_nn, MorseCp)
- subroutine morsecosine::morsec_forces_for_other_atoms (atoms, nl, MorseCp)

7.9 Файл INTERACTION_POTENTIALS/REBOsolidcarbon.f90

Типы данных

- type rebosolidcarbon::rebosc_parameters

Группы

- module rebosolidcarbon

Функции/подпрограммы

- subroutine rebosolidcarbon::read_rebosc_parameters (REBOscp, filename)
- subroutine rebosolidcarbon::rebosc_energy (energy, nl, REBOscp)

7.10 Файл INTERACTION_POTENTIALS/RosatoGuillopeLegrand.f90

Типы данных

- type rosatoguillopelegrand::rosatoguillopelegrand_parameters

Группы

- module rosatoguillopelegrand

Функции/подпрограммы

- subroutine rosatoguillopelegrand::read_rjl_parameters (RJLp, filename)
- subroutine rosatoguillopelegrand::rjl_energy (energy, nl, RJLp)
- subroutine rosatoguillopelegrand::rjl_forces (atoms, nl, RJLp)

7.11 Файл INTERACTION_POTENTIALS/TersoffBrenner.f90

Типы данных

- type tersoffbrenner::tersoffbrenner_parameters

Группы

- module tersoffbrenner

Функции/подпрограммы

- subroutine tersoffbrenner::read_tb_parameters (TBp, filename)
- subroutine tersoffbrenner::tb_energy (energy, nl, TBp)
- subroutine tersoffbrenner::tb_forces (atoms, nl, TBp)

7.12 Файл ljc_and_morsec_moire_graphene_fitting/fit_gr_moire.f90

Группы

- module fit_gr_moire

Функции/подпрограммы

- subroutine fit_gr_moire::calc_error (error, from_init_xyz, params)
- subroutine fit_gr_moire::set_fitting_parameters (fitting_parameters_file_name, init_min_↩ params, init_max_params)

Переменные

- integer fit_gr_moire::sim_num
- integer fit_gr_moire::out_period
- integer fit_gr_moire::num_of_omp_treads
- integer fit_gr_moire::out_id
- integer fit_gr_moire::final_out_id
- integer fit_gr_moire::oid
- integer, dimension(2) fit_gr_moire::ar_c_num
- character(len=256) fit_gr_moire::interaction_name
- character(len=256), dimension(2) fit_gr_moire::ar_settings_filename
- character(len=256) fit_gr_moire::output_prefix
- character(len=256) fit_gr_moire::input_path
- character(len=256) fit_gr_moire::out_path
- character(len=256), dimension(2) fit_gr_moire::ar_final_file
- character(len=256) fit_gr_moire::param_file
- character(len=256), dimension(2) fit_gr_moire::ar_start_xyz_file
- character(len=256), dimension(2) fit_gr_moire::ar_xyz_file
- real fit_gr_moire::z
- real, dimension(2) fit_gr_moire::ar_zero_energy_level
- real fit_gr_moire::be0
- real, dimension(2) fit_gr_moire::ar_grd0
- real, dimension(2) fit_gr_moire::rcut
- logical fit_gr_moire::simplified
- character(len=80) fit_gr_moire::line

7.13 Файл MOLECULAR_DYNAMICS/IFPORT_illusion.f90

Группы

- module ifport

  Заглушка для удобства компиляции. В IFPORT находится функция rand() при компиляции с ifort. При компиляции с gfortran такого модуля нет. Этот пустой модуль нужен чтобы не убирать use IFPORT в md_general при компиляции с gfortran.

7.14 Файл MOLECULAR_DYNAMICS/md_general.f90

Типы данных

- type md_general::time_steps
- type md_general::simulation_cell
- type md_general::particles
- type md_general::particle_group
- type md_general::nose_hoover_chain
- type md_general::neighbour_list
- type md_general::integrator_params

Группы

- module md_general

Функции/подпрограммы

- subroutine md_general::init_time_steps (dt, delta_t)
- subroutine md_general::create_particle_group (group, type_names, atoms)
- subroutine md_general::change_particle_group_n (group, md_step, change_ts1, change_ts2, change_frec, init_group)
- subroutine md_general::scale_velocities (atoms, group, s)
- subroutine md_general::random_velocities (atoms, group)
- subroutine md_general::random_momenta (atoms, group)
- subroutine md_general::calculate_kinetic_energy (ke, atoms, group)
- subroutine md_general::calculate_mass_center (mc, atoms, group)
- subroutine md_general::calculate_mass_center_velosity (mcv, atoms, group)
- subroutine md_general::zero_momentum (atoms, group)
- subroutine md_general::calculate_masses_sum (totm, atoms, group)
- subroutine md_general::calculate_force_sum (fs, atoms, group)
- subroutine md_general::calculate_temperature (temp, ke, atoms, group)
- subroutine md_general::set_new_temperature (atoms, group, temp)
- subroutine md_general::check_positions (out_id, atoms, box)
- subroutine md_general::check_velocities (out_id, atoms)
- subroutine md_general::invert_z_velocities (atoms, z_low_border, z_high_border)
- subroutine md_general::position_analysis (av, mi, ma, atoms, group, direction, minimum, maximum)
- pure subroutine md_general::find_distance (dr, dr2, vec1, vec2, box)

7.15 Файл MOLECULAR_DYNAMICS/md_integrators.f90

Группы

- module md_integrators

Функции/подпрограммы

- subroutine md_integrators::integrate_verlet_xyz_positions (atoms, group, steps, box)
- subroutine md_integrators::integrate_verlet_z_positions (atoms, group, steps, box)
- subroutine md_integrators::integrate_verlet_xyz_velocities (atoms, group, steps)
- subroutine md_integrators::integrate_verlet_z_velocities (atoms, group, steps)
- subroutine md_integrators::molecular_static_xyz_velocities (atoms, group)
- subroutine md_integrators::molecular_static_1d_velocities (atoms, group)
- subroutine md_integrators::zero_forces (atoms, group)
- subroutine md_integrators::create_nose_hoover_chain (nhc)
- subroutine md_integrators::set_nose_hoover_chain (nhc, temp, q1, l)
- subroutine md_integrators::integrate_nose_hoover_chain (nhc, atoms, group, dt)
- subroutine md_integrators::calculate_nose_hoover_chain_energy (nhc)

7.16 Файл MOLECULAR_DYNAMICS/md_interactions.f90

Типы данных

- type md_interactions::interaction_parameters
- type md_interactions::interaction

Группы

- module md_interactions

Функции/подпрограммы

- subroutine md_interactions::create_groups (groups, file_id, out_id, atoms)
- subroutine md_interactions::create_interactions (interactions, groups, file_id, out_id, input_↩ path)
- subroutine md_interactions::update_interactions_neighbour_lists (md_step, interactions, atoms, groups, cell, exe_time_nlsearch, exe_time_nldistance)
- subroutine md_interactions::allocate_graphene_norm (interactions)
- subroutine md_interactions::update_norm_in_graphene (interactions)
- subroutine md_interactions::calculate_forces (atoms, interactions)
- subroutine md_interactions::energy (inter_name, e, nl, p)
- subroutine md_interactions::calculate_potential_energies (interactions)
- subroutine md_interactions::calculate_forces_numerically (atoms, interactions)
- subroutine md_interactions::create_truncated_nl (tnl, nl)
- subroutine md_interactions::destroy_truncated_nl (tnl)
- subroutine md_interactions::calculate_truncated_nl (tnl, nl, i, n)
- subroutine md_interactions::shift_drs (tnl, inl, k, nl_n, dx)
- subroutine md_interactions::shift_gr_norm (gr_norm, nl_nn, inl, k, dx)
- subroutine md_interactions::nlists_load (out_id, interactions)

7.17 Файл MOLECULAR_DYNAMICS/md_neighbours.f90

Группы

- module md_neighbours

  Модуль содержит подпрограммы относящиеся к спискам соседей частиц

Функции/подпрограммы

- subroutine md_neighbours::create_neighbour_list (nl)

  Инициализирует пустой список соседей.
- subroutine md_neighbours::update_neighbour_list (md_step, nl, atoms, group1, group2, box, exe_time_nlsearch, exe_time_nldistance)

  Вычисляет расстояния между соседями. Обновляет список соседей с нужной частотой. Расстояния между соседями вычисляются всегда. Список соседей обнавляется с периодом указанным в списке. Также замеряется затраченное время.
- subroutine md_neighbours::find_neighbours (nl, atoms, group1, group2, box)

  Ищет соседей для частиц из первой группы среди второй группы. Группы могут совпадать.
- subroutine md_neighbours::find_neighbour_distances (nl, atoms, group1, group2, box)

  Пересчитывает взаимное расположение соседей.
- subroutine md_neighbours::converce_neighbour_list (cnl, group2, nl)

  Делает список соседей для второй группы из списка соседей для первой группы. Эквивалентно find_neighbours(cnl,atoms,group2,group1,box). Обращенный список соседей заполняется данными из списка полученного подпрогаммой find_neighbours(nl,atoms,group1,group2,box)

7.18 Файл MOLECULAR_DYNAMICS/md_read_write.f90

Группы

- module md_read_write

  Модуль ввода вывода .xyz файлов и настроек моделирования.

Функции/подпрограммы

- subroutine md_read_write::read_integrator_params (integr, file_id)

  Читает параметры интегратора.
- subroutine md_read_write::read_box_size (box, filename)

  Читает параметры моделируемой ячейки в файле .xyz. Читает второю строку в xyz файле. В размер ячейки задается тремя векторами. В общем случае ячейка триклинная.
- subroutine md_read_write::read_particles (atoms, filename)

  Читает информацию о частицах из .xyz файла. Первая строка файла содержит количество частиц. Третья и последующие - координаты по X, Y, Z, скорости по X, Y, Z, массу и название частицы. Всего 8 столбцов для каждой частицы.
- subroutine md_read_write::write_particle_group (filename, atoms, group, box)

  Выводит информацию о группе частиц в новый файл.
- subroutine md_read_write::write_particle_group_append (filename, atoms, group, box, md_↩ step)

  Выводит информацию о группе частиц в конец уже имеющегося файла.

7.19 Файл MOLECULAR_DYNAMICS/md_simulation.f90

Группы

- module md_simulation

Функции/подпрограммы

- subroutine md_simulation::md (out_id, all_out_id, input_path, settings_filename, output_↩ prefix, out_period, num_of_omp_treads)

7.20 Файл MOLECULAR_DYNAMICS/perfomance_settings.f90

Группы

- module perfomance_settings

  Модуль настройки OpenMP параллелизма.

Функции/подпрограммы

- subroutine perfomance_settings::set_openmp_perfomance (num_of_omp_treads, N)

  Настраивает количество OpenMP потоков и omp_chunk_size.

Переменные

- integer perfomance_settings::omp_chunk_size

  Массивы данных будут разбиваться на части длины omp_chunk_size при использовании omp parallel for schedule(dynamic ,omp_chunk_size)

7.21 Файл runners/run_gr_analysis.f90

Функции/подпрограммы

- program run_gr_analysis

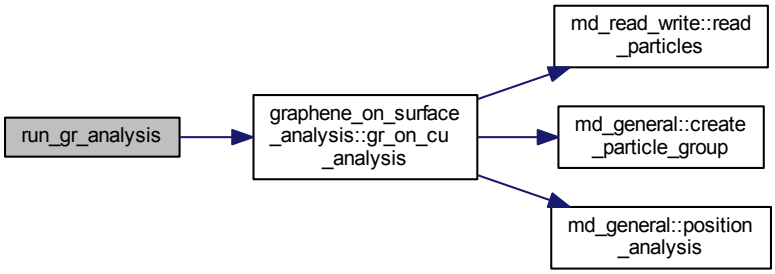
7.21.1 Функции/подпрограммы

7.21.1.1 run_gr_analysis()

program run_gr_analysis ( )

См. определение в файле run_gr_analysis.f90 строка 1

Граф вызовов:



7.22 Файл runners/run_gr_moire_fitting.f90

Функции/подпрограммы

- program run_gr_moire_fitting
- real function delta_error (error_array)

7.22.1 Функции/подпрограммы

7.22.1.1 delta_error()

```
real function delta_error (
                real, dimension(4) error_array )
```

См. определение в файле run_gr_moire_fitting.f90 строка 108

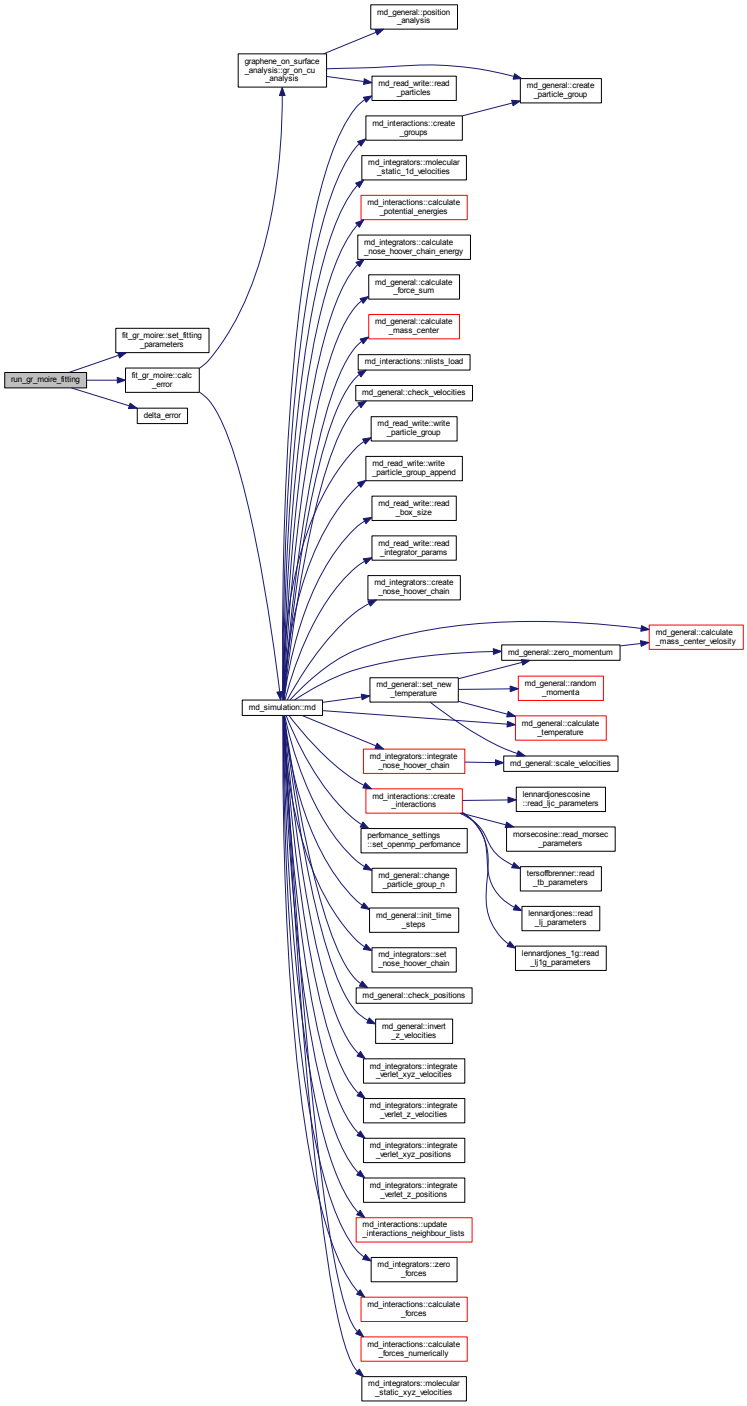
```
108     real:: delta_error,error_array(4)
109     delta_error = maxval(error_array)-minval(error_array)
110     return
```

7.22.1.2 run_gr_moire_fitting()

```
program run_gr_moire_fitting ( )
```

См. определение в файле run_gr_moire_fitting.f90 строка 1

Граф вызовов:



7.23 Файл runners/run_md_simulation.f90

Функции/подпрограммы

- program md_run

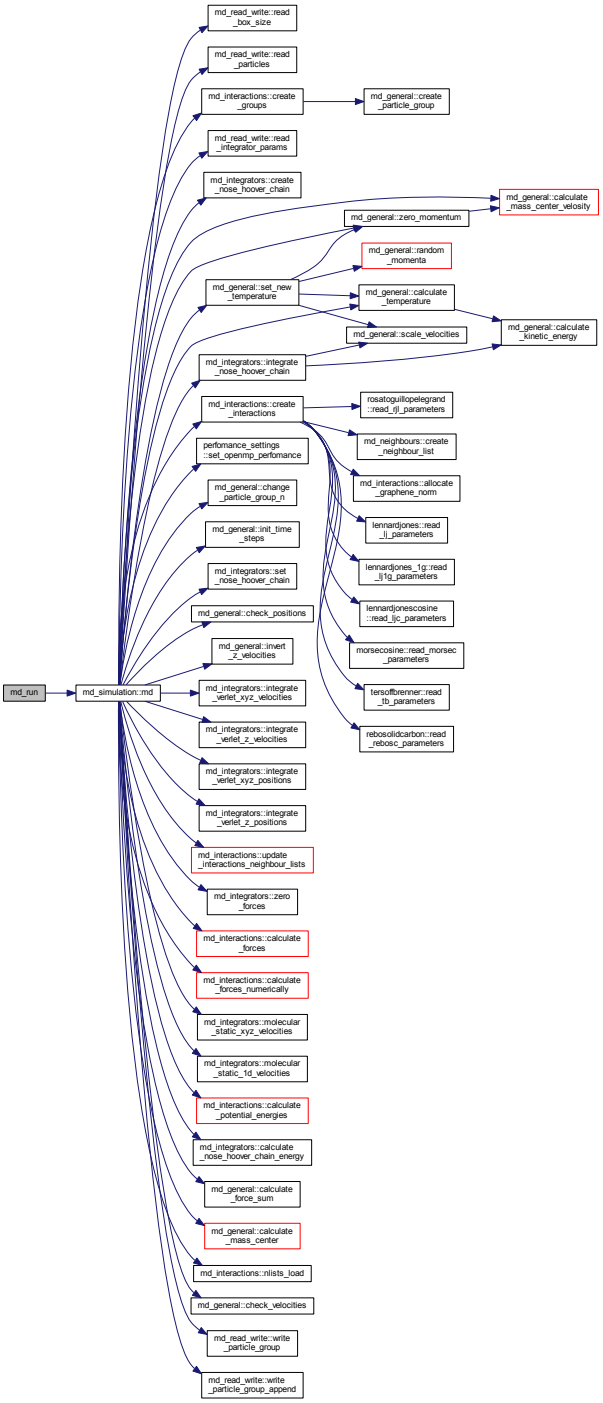
7.23.1 Функции/подпрограммы

7.23.1.1 md_run()

program md_run ( )

См. определение в файле run_md_simulation.f90 строка 1

Граф вызовов:



Предметный указатель