```
1   module cell

2

3      use global_env

4

5      integer, parameter :: cell_share_max=50

6

7      integer :: cell_share_num
8      integer, dimension(cell_share_max) :: cell_local_walkers

9

10     integer, dimension(:,:,:), allocatable :: cell_list_root
11     integer, dimension(:), allocatable :: cell_list

12

13     integer, dimension(3) :: cell_size
14     integer, dimension(3) :: n_cells
15     integer, dimension(:,:), allocatable :: cell_walker

16

17     contains

18

19     subroutine cell_find_walkers(cell_index,cell_local_walkers)
20        ! ****************************************
21        ! Find all walkers in the specified cell
22        ! ****************************************
23        implicit none
24        integer, dimension(:), intent(in) :: cell_index
25        integer, dimension(:), intent(out) :: cell_local_walkers
26        integer i, n

27

28        i=cell_list_root(cell_index(X),cell_index(Y),cell_index(Z)
            )

29
```

```fortran
30      if(i==0) then  ! no walker in the cell
31          cell_local_walkers(1)=0
32          return
33      else ! found a walker in the cell
34          n=1
35          cell_local_walkers(n)=i
36      end if

38      do while(i>0)
39          n=n+1
40          i=cell_list(i)
41          if(i==0) then  ! no more walker
42              cell_local_walkers(n)=0
43          else ! still more walkers
44              cell_local_walkers(n)=i
45          end if
46      end do

48  end subroutine cell_find_walkers

50  function cell_coordinates(walker_pos)
51      ! ***********************************************
52      ! Find the index of the cell the walker belongs to
53      ! ***********************************************
54      implicit none

56      integer, dimension(3) :: cell_coordinates

58      integer, intent(in), dimension(3) :: walker_pos

60      cell_coordinates = (walker_pos-1)/cell_size+1
```

```fortran
61
62   end function cell_coordinates
63
64   subroutine cell_init(n_walkers,max_walkers,walker_pos)
65
66     implicit none
67
68     integer, intent(in) :: n_walkers, max_walkers
69     integer, intent(in), dimension(:,:) :: walker_pos
70     integer :: i
71
72     if(any(cell_size<1) ) then
73        write(fmain,*) '*** cell_init ***  cell_size is too
             small.'
74        stop
75     end if
76
77     n_cells = box_size/cell_size+1
78
79     if( any((n_cells-1)*cell_size /= box_size) ) then
80        write(fmain,*) '*** cell_init ***  cell_size is not
             consistent.'
81        stop
82     end if
83
84     allocate(cell_list(max_walkers))
85     allocate(cell_walker(max_walkers,3))
86     allocate(cell_list_root(n_cells(X),n_cells(Y),n_cells(Z)))
87
88   !***********************************
89   ! Find a cell that an atom belongs to.
```

```fortran
90      !**********************************
91
92          do i=1,n_walkers
93              cell_walker(i,:) = cell_coordinates(walker_pos(i,:))
94          end do
95
96
97          ! Set initial cell lists.
98          cell_list_root = 0
99          cell_list = 0
100          do i=1,n_walkers
101              cell_list(i) = cell_list_root(cell_walker(i,X),
                     cell_walker(i,Y),cell_walker(i,Z))
102              cell_list_root(cell_walker(i,X),cell_walker(i,Y),
                     cell_walker(i,Z)) = i
103          end do
104
105      end subroutine cell_init
106
107  ! ***************
108  !  cell_crossing
109  ! ***************
110      subroutine cell_crossing(walker_id,walker_pos)
111
112          implicit none
113
114          integer, intent(in) :: walker_id
115          integer, intent(in), dimension(3) :: walker_pos
116          integer :: n
117          integer, dimension(3) :: new_cell, old_cell
118
```

```fortran
119         new_cell = cell_coordinates(walker_pos)

120

121         if(all(new_cell == cell_walker(walker_id,:))) return

122

123         old_cell = cell_walker(walker_id,:)
124         cell_walker(walker_id,:) = new_cell

125

126         n = cell_list_root(old_cell(X),old_cell(Y),old_cell(Z))

127

128         if(n<1) then
129             write(*,*) 'Error in cell_list_root'
130             stop
131         end if

132

133         if(n==walker_id) then
134             cell_list_root(old_cell(X),old_cell(Y),old_cell(Z))=
                    cell_list(n)
135         else
136             do while ( cell_list(n) /= walker_id )
137               n = cell_list(n)
138             end do
139             cell_list(n) = cell_list(walker_id)
140         endif
141         cell_list(walker_id) = 0

142

143         n = cell_list_root(new_cell(X),new_cell(Y),new_cell(Z))
144         cell_list(walker_id) = n
145         cell_list_root(new_cell(X),new_cell(Y),new_cell(Z))=
                    walker_id

146

147     end subroutine
```

```fortran
148
149    subroutine cell_test(n_walkers)
150
151        implicit none
152
153        integer, intent(in) :: n_walkers
154        integer :: n, n_errors
155        integer :: i, j, k, l
156
157        write(fdbg,*) '*** Checking consistency of cell assignment
                ***'
158        write(fdbg,*)
159        n=0
160        n_errors=0
161        do i=1,n_cells(Z)
162            do j=1,n_cells(Y)
163                do k=1,n_cells(X)
164                    l=cell_list_root(k,j,i)
165
166                    do while(l>0)
167                        n=n+1
168                        if(any(cell_walker(l,:)/=(/k,j,i/)))  then
169                            write(fdbg, '(a,a,i6,a,3i4,a,3i4,a)') &
170                            'cell mismatch found:',l,'(',cell_walker(l
                                ,:),') != (',k,j,i,')'
171                            n_errors=n_errors+1
172                        endif
173                        l=cell_list(l)
174                    end do
175
176                end do
```

6

```fortran
177          end do
178        end do
179
180        if(n<n_walkers) then
181           write(fdbg,'(a,i6)')  'Some  walkers are missing: ',n
182        else if (n>n_walkers) then
183           write(fdbg,'(a,i6)')  'Too many walkers are found:',n
184        else
185           write(fdbg,'(a,i6)')  'All walkers are accounted:',n
186        end if
187
188        if(n_errors>0) then
189           write(fdbg,*) 'Inconsistency is found.'
190        else
191           write(fdbg,*) 'No inconsistency is found.'
192        end if
193     end subroutine cell_test
194
195 end module cell
```