

```

1
2
3
4  module tbbop_emb
5
6  use mod_precision
7  use mod_const
8  use mod_conf
9  use mod_clock
10 use topologia
11
12 #ifdef MPI
13 use mpi
14 #endif
15
16 implicit none
17
18 integer, parameter :: nr = 2 ! number of regions (only TB and BOP so far)
19
20
21 type mbtb_t
22   logical :: ovl ! overlap switch (not implemented yet)
23   integer :: rlst(nr+1) ! atom offsets for regions
24   integer :: n(nr+1) ! number of atoms according to region
25   integer :: nd ! total number of atoms
26   integer :: thsz, bhsz, ahsz ! sizes for TB, BOP and global Hamiltonians respectively
27   ! posham, aptr_tb bptr_tb aptr_dh bptr_dh
28   integer, allocatable :: pth(:), pbh(:), pah(:) ! atom offsets for Hamiltonians
29   integer, allocatable :: ad(:,:) ! coordinates of dynamic atoms
30   integer, allocatable :: apt(:), bpt(:) ! neighbour lists for TB -> TB
31   integer, allocatable :: apd(:), bpd(:) ! neighbour lists for TB -> BOP
32   integer, allocatable :: apb(:), bpb(:) ! neighbour lists for BOP -> BOP
33
34   real(dp), allocatable :: h(:,:), s(:,:) ! TB hamiltonian and overlap matrices in dense form
35   real(dp), allocatable :: dh(:,:,:) ! TB->BOP matrix elements in sparse form
36   real(dp), allocatable :: q(:), rho(:,:,:), : ! global atomic charges and density matrix in sparse
37   ! form
38   !, v(:), ! rho(mxnstat,mxnstat,jb,ia)
39
40   complex(dp), allocatable :: a(:,:), ac(:,:), gt(:,:) !In short:
41   ! a = G_BOP dh'
42   ! gt = (z - (h + dh a))^{-1}
43   ! ac = a gt
44   ! see the notes in 'formuli.pdf' pages: 4-6 for details.
45
46   complex(dp), allocatable :: ggd(:) ! global Green matrix diagonal elements
47   !, w(:,:), occ(:), gg(:,:)
48 end type
49
50 contains
51
52 subroutine emb_sc()
53 ! self consistency loop.
54 ! This is the entry point for the embedding program.
55
56 use mod_all_scalar, only : mag, quiet, dqmax, forces, mgmax, nd, qerr
57 use mod_atom_ar, only : dq, mg
58
59 include "../Include/Atom.array"
60
61
62 type(mbtb_t) :: mbc ! embedding conf
63
64 integer :: it, mxit, ia, mgit, mit
65 real(dp) :: etott
66 real(qp) :: avdde, sumz, merr
67 real(qp), allocatable :: de_prev(:), dq_prev(:), dec(:), td(:), dql(:)
68 real(dp), allocatable :: mg_o(:)
69
70 character(len=25) :: ffmt
71
72 integer,parameter :: mixing_type = 2
73

```

```

74     integer(8) :: c1,c2
75
76     integer :: i
77     real(dp) :: ef,nf
78
79     integer :: ib, ierr
80     real(dp) :: dqmax_old, aa, aamin, aamax
81
82     ef = 0.0_dp
83     !       mxit = 500
84     mxit = rtc % bop_conf % mxit
85     if (mag) then
86         mgit = rtc % bop_conf % mgit
87         merr = rtc % bop_conf % merr
88         mit = 1
89         if (merr < 0.0_dp) mgit = 1
90     end if
91
92     allocate(de_prev(nd), dq_prev(nd), td(nd), dec(nd), dql(nd))
93     dec = real(de(:nd),qp)
94     sumz = sum( (/ (real(zc(z(ia))), qp), ia = 1,nd) /) )
95
96 !
97 !+++++
98 ! This is temporary while writing the code only! should certainly be changed to load it from the cell
99 ! file.
100 !
101 !       mbc % rlst(1) = 1
102 !       mbc % rlst(2) = 21
103 !       mbc % rlst(3) = 41
104 !
105 !       mbc % rlst(1) = 1
106 !       mbc % rlst(2) = 220
107 !       mbc % rlst(3) = 271
108 !
109 !       mbc % ovl = .false.
110 ! The cell file will specify the numbers nt, nb, and nd then list all atoms ordered as follows:
111 ! first the bop atoms then the tb atoms and afterwards the pure bop atoms. Pure bop atoms are not
112 ! considered yet
113 ! but should be kept in mind
114 !
115 !+++++
116
117     call mblhdh(mbc)
118
119     if ((qerr < 0.0_dp).or.(mixing_type == 0)) mxit = 1
120     if (.not. quiet) then
121         write(9, "('Imposing LCN ...')")
122         write(ffmt, "(a,i0,a)") '(a,":",', mbc%nd, '(x,f22.14))'
123     end if
124
125     if (mag) allocate(mg_o(mbc%nd))
126
127     do
128         if (mag) then
129             if (.not. quiet) write(6, '(/,"mit:",i3)') mit
130             mg_o = mg
131
132             do ia = 1, mbc%nd
133                 dem(ia) = 0.5_dp*istn(z(ia))*mg(ia)
134             end do
135         end if
136     end do
137
138     if linmix
139         dqmax_old = 0.0_dp
140         aamax=1.0_dp
141         aamin=1.0e-4_dp
142         aa=10.0_dp
143     endif
144
145     it = 1

```

```

142      do
143      !          print "('sumzde,de:',2(x,f22.14))", sum( (/ (de(ia)*zc(z(ia))), ia = 1,nd) / ) ),
144      !          de(9)
145      de(:mbc%nd) = real(dec - sum( (/ (dec(ia)*zc(z(ia))), ia = 1,mbc%nd) / ) ) / sumz, dp)
146      !          print "('bde:',x,f22.14)", de(9)
147      if (.not. quiet) then
148          write(6, '(//,"it:",i3)') it
149          write(6, ffmt) 'de ', de(:mbc%nd)
150          write(6, ffmt) 'dem', dem(:mbc%nd)
151      end if
152      call mbeq(mbc)
153
154      !          dec(:nd) = de(:nd)
155      dql = real(dq,qp)
156
157      print *, redb, 'dqmax:', dqmax, endc
158      if (abs(dqmax) <= qerr .or. it == mxit) exit
159
160      !          if (it > 1) then
161      !              do ia = 1, nd
162      !                  if (.not. quiet) print *, 'dif:', (dec(ia) - de_prev(ia)) / (dq(ia) - dq_prev(ia) ),
163      !                  abs(dq(ia) - dq_prev(ia))
164      !                  if ( abs(dec(ia) - de_prev(ia)) > 6.0_qp*abs(dql(ia) - dq_prev(ia))) then
165      !                      td(ia) = -dql(ia)
166      !                      if (.not. quiet) print *,red,'potential daner', (dec(ia) - de_prev(ia)), (dq(ia) -
167      !                      dq_prev(ia)),endc
168      !                      else
169      !                          td(ia) = dql(ia) * (dec(ia) - de_prev(ia)) / ( dql(ia) - dq_prev(ia) )
170      !                      endif
171      !                  end do
172      !                  de_prev = dec
173      !                  dq_prev = dql
174      !                  dec = dec - td
175      !                  !          write(6,"('td,nde:',2(x,f22.14))") td(9),de(9)
176      !                  else
177      !                      de_prev = dec
178      !                      dq_prev = dql
179      !                      dec = dec + dql
180      !                  end if
181
182      if ( (dqmax_old*dqmax > 0.0_dp &
183      & .and. abs(dqmax_old) > abs(dqmax)) &
184      & .or. (dqmax_old*dqmax < 0.0_dp)) then
185          aa=aa*abs(dqmax_old)/abs(dqmax_old-dqmax)
186      end if
187
188      aa=max(aamin,aa)
189      aa=min(aamax,aa)
190
191      if (abs(dqmax) <= qerr) aa=aamin
192
193      print *, redb, 'aa:', aa, endc
194      dec(:mbc%nd) = de(:mbc%nd) + aa*dq(:mbc%nd)
195
196
197      dqmax_old = dqmax
198
199      it = it + 1
200  end do
201
202  if (.not. mag) exit
203  mgmax = maxval(abs(mg - mg_o))
204
205  if (.not. quiet) then
206      write(6,ffmt) 'mgo', mg_o
207      write(6,ffmt) 'mg ', mg
208      write(6,*) 'mgmax:', mgmax
209  end if
210  if (mgmax <= merr .or. mit == mgit) exit
211  mit = mit + 1
212 end do

```

```

213
214     if (mag) deallocate(mg_o)
215     deallocate(de_prev, dq_prev, td, dec, dql)
216
217
218     if (forces) call mbf(mbc)
219
220 end subroutine emb_sc
221
222
223
224 subroutine mbf(mbc)
225     use mod_all_scalar, only : mag, quiet, lef, locc, dqmax, nbase
226     type(mbtb_t), intent(inout) :: mbc
227
228     call mbrho(mbc, lef)
229
230
231
232 end subroutine mbf
233
234
235
236
237 subroutine gg_diag(mbc, ez, isp)
238
239     type(mbtb_t), intent(inout) :: mbc
240     complex(dp), intent(in) :: ez
241     integer, intent(in) :: isp
242
243     integer :: i, j
244
245
246     call a_and_ac(mbc, ez, isp)
247
248
249 !     write( 389, *) 'ez', ez
250
251 !     call print_c(mbc % ac, 389, 'ac'//achar(10))
252
253 !     call print_c(mbc % h, 389, 'h')
254
255
256 !The BOP G is already added, here we just add the mixed term ACA*
257
258     do i = 1, mbc % bhsz
259         do j = 1, mbc % thsz
260             mbc % ggd(i) = mbc % ggd(i) + mbc % ac(i,j)* conjg(mbc % a(i,j))
261         end do
262     end do
263
264     do i = 1, mbc % thsz
265         mbc % ggd(mbc % bhsz + i) = mbc % gt(i,i)
266     end do
267
268
269
270 !     calc the diagonal here; do over atoms and orbs  gb(ia,ia) += sum_ib(ac(ia,ib) * a(ia,ib))
271
272 !     stop
273 end subroutine gg_diag
274
275
276
277 subroutine gg_rho(mbc, ez, zp, isp)
278     use ab_io
279     use mod_ham
280
281     type(mbtb_t), intent(inout) :: mbc
282     complex(dp), intent(in) :: ez, zp
283     integer, intent(in) :: isp
284
285     integer :: ia, gia, ib, gib, it, git, ja, jb, jt, i, j, jt0
286

```

```

287 complex(dp) :: gab(mxnstat, mxnstat)
288
289
290 call a_and_ac(mbc, ez, isp)
291
292 do ia = 1, mbc % n(1)
293
294     call assoc_ab(ia,isp)
295     call assoc_ham(ia) ! for cluster, pcluster and decipher
296
297     gia = mbc % rlst(1) -1 + ia
298
299 !     BOP onsite
300     gab = 0.0_dp
301     call get_gab(ez, ia, ia, 1, gab)
302
303     do i = 1, mbc % pbh(ia+1) - mbc % pbh(ia)
304         do j = 1, mbc % pbh(ia+1) - mbc % pbh(ia)
305             gab(i,j) = gab(i,j) + sum(mbc % ac(mbc % pbh(ia) + i, :) * conjg(mbc % a(mbc % pbh(ia)
306                 + j, :)))
307         end do
308     end do
309
310     mbc % rho(:, :, 1, gia) = mbc % rho(:, :, 1, gia) + real(zp*gab)
311
312 !     BOP intersites
313
314     ja = mbc % apb(ia)
315     ib = mbc % bpb(ja)
316
317     do while (ib /= eol)
318         jb = decipher(ib)
319
320         if (jb /= 0) then !jb == 0 => ib == ia
321             gib = mbc%rlst(1)-1 + ib !
322
323             gab = 0.0_dp
324             call get_gab(ez, ia, ib, jb, gab)
325
326             do i = 1, mbc % pbh(ia+1) - mbc % pbh(ia)
327                 do j = 1, mbc % pbh(ib+1) - mbc % pbh(ib)
328                     gab(i,j) = gab(i,j) + sum(mbc % ac(mbc % pbh(ia) + i, :) * conjg(mbc % a(mbc %
329                         pbh(ib) + j, :)))
330                 end do
331             end do
332
333             mbc % rho(:, :, jb, gia) = mbc % rho(:, :, jb, gia) + real(zp*gab)
334
335             end if
336             ja = ja + 1
337             ib = mbc % bpb(ja)
338         end do
339
340     end do
341
342     do it = 1, mbc % n(2)
343
344         git = mbc % rlst(2) -1 + it
345
346 !         TB all
347
348         jt = mbc % apt(it)
349         jt0 = jt - 1
350         ib = mbc % bpt(jt)
351
352         do while (ib /= eol)
353
354             gab = 0.0_dp
355             gab(1 : mbc % pth (it + 1) - mbc % pth(it), 1 : mbc % pth (ib + 1) - mbc % pth(ib)) = &
356                 & mbc % gt(mbc % pth(it) + 1 : mbc % pth(it + 1), mbc % pbh(ib) + 1 : mbc % pbh(ib +
357                     1))

```

```

358         mbc % rho(:, :, jt-jt0, it) = mbc % rho(:, :, jt-jt0, it) + real(zp*gab)
359
360         jt = jt + 1
361         ib = mbc % bpt(jt)
362     end do
363
364
365 !         TB->BOP intersites
366         jt = mbc % apd(it)
367         jt0 = jt - 1
368         ib = mbc % bpd(ja)
369
370         do while (ib /= eol)
371             gib = mbc%rlst(2)-1 + ib
372
373             gab = 0.0_dp
374 ! this is incomplete:
375 !             gab(1 : mbc % pth (it + 1) - mbc % pth(it), 1 : mbc % pbh (ib + 1) - mbc % pbh(ib)) = &
376 !             & - conjg(transpose(mbc % pbh(ib) + 1 : mbc % pbh(ib + 1)), mbc % ac(mbc % pth(it) + 1
377 : mbc % pth(it + 1) ))
378
379             mbc % rho(:, :, jt-jt0, it) = mbc % rho(:, :, jt-jt0, it) + real(zp*gab)
380
381             jt = jt + 1
382             ib = mbc % bpt(jt)
383         end do
384 !TODO: complete the transpose BOP->TB elements .... or may be put it in the previous loop over ia
385
386     end do
387
388
389
390 end subroutine gg_rho
391
392
393
394
395 ! subroutine gtb(mbc, ez, isp)
396 !     use mod_all_scalar, only : mag
397 !
398 !
399 !     type(mbtb_t), intent(inout) :: mbc
400 !     complex(dp), intent(in) :: ez
401 !     integer, intent(in) :: isp
402 !
403 !     integer :: nstt1, nstt2, nsttb, l1, l2, lb, jt, it1, it2, ib, pt1, pt2, pb, it, git, lt
404 !     integer :: i, off
405 !
406 !     real(dp) :: dea
407 !     complex, parameter :: z_one = 1.0_dp, z_zero = 0.0_dp
408 !
409 !
410 !     integer :: info, lwork
411 !     real(dp), allocatable :: rwork(:)
412 !     complex(dp), allocatable :: work(:)
413 !
414 !     include '../Include/Atom.array'
415 !
416 !     ! (z*I*S - (H + dH'*Gb*dH))^-1 = W (z*I - V)^-1 W'
417 !
418 !     mbc % w = mbc % h
419 !     mbc % w = 0.0_dp
420 !
421 !     off = mbc % rlst(2)-1
422 !
423 !     do it = 1, mbc % n(2)
424 !         git = off + it
425 !
426 !         dea = de(git)
427 !         if (mag) dea = dea + oppm(isp)*dem(git)
428 !
429 !         do lt = mbc % pth(it)+1, mbc % pth(it+1)
430 !             mbc % w(lt, lt) = mbc % w(lt, lt) + dea

```

```

431 !         end do
432 !     end do
433 !
434 !
435 !     do it1 = 1, mbc % n(2)
436 !         pt1 = mbc % pth(it1)
437 !         nstt1 = mbc % pth(it1+1) - pt1
438 !
439 !         do it2 = 1, mbc % n(2)
440 !             pt2 = mbc % pth(it2)
441 !             nstt2 = mbc % pth(it2+1) - pt2
442 !
443 !             jt = mbc % apd(it1)
444 !             ib = mbc % bpd(jt)
445 !             do while ( ib /= eol)
446 !                 pb = mbc % pbh(ib)
447 !                 nsttb = mbc % pbh(ib+1) - pb
448 !
449 !                 do l1 = 1, nstt1
450 !                     do l2 = 1, nstt2
451 !                         do lb = 1, nsttb
452 !                             mbc % w(pt1+l1,pt2+l2) = mbc % w(pt1+l1,pt2+l2) &
453 !                                 & + mbc % dh(l1, lb, jt) * mbc % a (pb+lb,pt2+l2)
454 !                         end do
455 !                     end do
456 !                 end do
457 !
458 !                 jt = jt + 1
459 !                 ib = mbc % bpd(jt)
460 !             end do
461 !
462 !         end do
463 !     end do
464 !
465 !     call print_c(mbc%w,500,'sgm'//achar(10))
466 !
467 ! !     stop
468 !     allocate(work(1), rwork(3*mbc % thsz-2))
469 !     lwork = -1
470 !
471 !     if (.not. mbc%ovl) then
472 !         call zheev('v', 'l', mbc % thsz, mbc % w, mbc % thsz, mbc % v, work, lwork, rwork, info)
473 !     else
474 !         call zhegv(1, 'v', 'l', mbc % thsz, mbc % w, mbc % thsz, mbc % s, mbc % thsz, &
475 !             & mbc % v, work, lwork, rwork, info)
476 !     end if
477 !
478 !     lwork = work(1)
479 !     deallocate(work)
480 !     allocate(work(lwork))
481 !
482 !     if (.not. mbc%ovl) then
483 !         call zheev('v', 'l', mbc % thsz, mbc % w, mbc % thsz, mbc % v, work, lwork, rwork, info)
484 !     else
485 !         call zhegv(1, 'v', 'l', mbc % thsz, mbc % w, mbc % thsz, mbc % s, mbc % thsz, &
486 !             & mbc % v, work, lwork, rwork, info)
487 !     end if
488 !
489 !     deallocate(work,rwork)
490 !
491 ! !     print *, info
492 !
493 !     call print_c(mbc%v,400,'v'//achar(10))
494 !
495 !     do i = 1, mbc % thsz
496 !         mbc % w (:,i) = mbc % w(:,i)/sqrt(ez - mbc % v(i))
497 !     end do
498 !
499 !     call print_c(mbc%w,400,'w'//achar(10))
500 !
501 !
502 !
503 !     call zgemm ('n', 'c', mbc % thsz, mbc % thsz, mbc % thsz, z_one, &
504 !         & mbc % w, mbc % thsz, mbc % w, mbc % thsz, &

```

```

505 !          & z_zero, mbc % gt, mbc % thsz)
506 !
507 !      stop 'remove me when you find out whats wrong with zgemm'
508 !
509 !      call print_c(mbc%gt,400,'gt'//achar(10))
510 !
511 !
512 ! !      AC, C: gt. may be try zhemm?
513 !      call zgemm ('n', 'n', mbc % bhsz, mbc % thsz, mbc % thsz, z_one, &
514 !          & mbc % a, mbc % bhsz, mbc % gt, mbc % thsz, &
515 !          & z_zero, mbc % ac, mbc % bhsz)
516 !
517 !      end subroutine gtb
518 !
519
520
521 subroutine gtbi(mbc, ez, isp)
522     use mod_all_scalar, only : mag
523
524
525     type(mbtb_t), intent(inout) :: mbc
526     complex(dp), intent(in) :: ez
527     integer, intent(in) :: isp
528
529     integer :: nstt1, nstt2, nsttb, l1, l2, lb, jt, it1, it2, ib, pt1, pt2, pb, it, git, lt
530     integer :: i, off, j
531
532     real(dp) :: dea
533     complex, parameter :: z_one = 1.0_dp, z_zero = 0.0_dp
534
535
536     integer :: info, lwork
537     real(dp), allocatable :: rwork(:)
538     complex(dp), allocatable :: work(:)
539     integer, allocatable :: ipiv(:)
540
541     include '../Include/Atom.array' ! for dem only
542
543     ! (z*I*S - (H + dH'*Gb*dH))^-1 = W (z*I - V)^-1 W'
544
545     mbc % gt = 0.0_dp
546     mbc % gt = mbc % h
547 !      mbc % gt = 0.0_dp
548
549     off = mbc % rlst(2)-1
550
551     do it = 1, mbc % n(2)
552         git = off + it
553
554         dea = de(git)
555         if (mag) dea = dea + oppm(isp)*dem(git)
556
557         do lt = mbc % pth(it)+1, mbc % pth(it+1)
558             mbc % gt(lt,lt) = mbc % gt(lt,lt) + dea
559         end do
560     end do
561
562
563     do it1 = 1, mbc % n(2)
564         pt1 = mbc % pth(it1)
565         nstt1 = mbc % pth(it1+1) - pt1
566
567         do it2 = 1, mbc % n(2)
568             pt2 = mbc % pth(it2)
569             nstt2 = mbc % pth(it2+1) - pt2
570
571             jt = mbc % apd(it1)
572             ib = mbc % bpd(jt)
573             do while ( ib /= eol)
574                 pb = mbc % pbh(ib)
575                 nsttb = mbc % pbh(ib+1) - pb
576
577                 do l1 = 1, nstt1
578                     do l2 = 1, nstt2

```



```

579         do lb = 1, nsttb
580             mbc % gt(pt1+11,pt2+12) = mbc % gt(pt1+11,pt2+12) &
581                 & + mbc % dh(11, lb, jt) * mbc % a (pb+lb,pt2+12)
582         end do
583     end do
584 end do
585
586     jt = jt + 1
587     ib = mbc % bpd(jt)
588 end do
589
590 end do
591 end do
592
593 mbc % gt = - mbc % gt
594
595 if (.not. mbc % ovl) then
596     do i = 1, mbc % thsz
597         mbc % gt(i,i) = mbc % gt(i,i) + ez
598     end do
599 else
600     mbc % gt = mbc % gt + mbc % s * ez
601 end if
602
603
604 allocate(ipiv(mbc % thsz))
605
606 call zgetrf (mbc % thsz, mbc % thsz, mbc % gt, mbc % thsz, ipiv, info)
607 if (info /= 0) stop 'zgetrf info /= 0'
608
609 allocate(work(1))
610 lwork = -1
611
612 call zgetri(mbc % thsz, mbc % gt, mbc % thsz, ipiv, work, lwork, info)
613 if (info /= 0) stop 'zgetri i info /= 0'
614
615 lwork = work(1)
616 deallocate(work)
617 allocate(work(lwork))
618
619 call zgetri(mbc % thsz, mbc % gt, mbc % thsz, ipiv, work, lwork, info)
620 if (info /= 0) stop 'zgetri r info /= 0'
621
622 deallocate(work,ipiv)
623
624
625
626 !       There is something going on with this zgemm
627 !       call zgemm ('n', 'n', mbc % bhsz, mbc % thsz, mbc % thsz, z_one, &
628 !           & mbc % a, mbc % bhsz, mbc % gt, mbc % thsz, &
629 !           & z_zero, mbc % ac, mbc % bhsz)
630
631 ! NOTE: quite inefficient
632 do j = 1, mbc % thsz
633     do i = 1, mbc % bhsz
634         mbc % ac(i,j) = sum(mbc % a(i,:) * mbc % gt(:,j))
635     end do
636 end do
637
638
639 end subroutine gtbi
640
641
642 subroutine gb_x_dh(mbc,ez,isp)
643     use ab_io
644     use mod_ham
645
646     type(mbtb_t), intent(inout) :: mbc
647     complex(dp), intent(in) :: ez
648     integer, intent(in) :: isp
649
650     integer :: ib, ia, it, jb, jt, la, lb, lt, gia, gib, git
651
652     complex(dp) :: gab(mxnstat, mxnstat)

```

```

653
654     mbc%a = 0.0_dp
655
656     do ia = 1, mbc % n(1)
657
658         call assoc_ab(ia,isp)
659         call assoc_ham(ia) ! for cluster, pcluster and decipher
660
661         gia = mbc%rlst(1)-1 + ia ! gia == ia, not sure why am i doing the extra addition.
662         consistency?
663
664         call get_gab(ez, ia, ia, 1, gab)
665
666         do la = 1, mbc%pah(gia+1)-mbc%pah(gia) ! it is a little dirty to sed ggd and gg here
667             mbc % ggd(mbc%pah(gia) + la) = gab(la,la) ! but in this way gab is reused and another
668             call get_gab is avoided
669         end do
670
671         print *, 'ia:', ia
672         do it = 1, mbc % n(2)
673             jt = mbc % apd(it)
674             ib = mbc % bpd(jt)
675             git = mbc % rlst(2) -1 + it
676             print *, 'it:', it
677             do while ( ib /= eol)
678                 jb = decipher(ib) ! ib is local for region 1 as is decipher
679                 print *, 'jb:', jb
680                 if (jb /= 0) then
681                     print *, 'ib:', ib
682                     gib = mbc%rlst(1)-1 + ib !
683
684                     call get_gab(ez, ia, ib, jb, gab)
685
686                     do lt = mbc % pth(it) + 1, mbc % pth(it + 1)
687                         do la = mbc % pbh(ia)+1, mbc % pbh(ia + 1)
688                             mbc%a(la, lt) = 0.0_dp
689                             do lb = 1, mbc % pbh(ib + 1) - mbc % pbh(ib)
690                                 mbc % a(la, lt) = mbc % a(la, lt) &
691                                     & + gab(la - mbc % pbh(ia), lb) * mbc%dh(lt - mbc % pth(it), lb, jt)
692                             end do
693                         end do
694                     end do
695                 end if
696                 jt = jt + 1
697                 ib = mbc % bpd(jt)
698             end do
699         end do
700     end do
701
702 end subroutine gb_x_dh
703
704
705
706 subroutine get_gab(ez, ia, ib, jb, gab)
707     use ab_io
708     use mod_g0n
709     use mod_all_scalar, only : momflg, term
710
711     complex(dp), intent(in) :: ez
712     integer, intent(in) :: ia, ib, jb
713     complex(dp), intent(out) :: gab(mxnstat,mxnstat)
714
715     complex(dp) :: g0n(0:mrec+2), hia(0:mrec+1), hib(1:mrec+1)
716
717     integer :: la, lb, nla, nstta, nsttb, lla, nma, ma, nmax, nrec, za, zb
718
719     include "../Include/Atom.array"
720
721     za = z(ia) ! offset = 0 region 1 ! These are inside because the number of cross-neighbours is
722     expected to be low.
723     call states(za, nla, nstta, llista) !

```

```

724   if (momflg == 1) then
725       lla = 0
726       do la = 1, nla
727           nma = 2*llista(la) + 1
728           mlista(lla+1:lla+nma) = la
729           lla = lla + nma
730       enddo
731   else
732       do la = 1, nstta
733           mlista(la) = la
734       enddo
735   endif
736
737   gab = 0.0_dp
738
739   nsttb = nstt(z(ib))
740
741   do la = 1, nstta
742
743       ma = mlista(la)
744       nrec = lchain(ma)
745       nmax = nrec + 1
746
747       call getg0n(ez, arec(0:nrec,ma), brec(0:nrec,ma), g0n, nmax, nrec, lainf(ma), lbinf(ma))
748
749       hia(0:nmax) = g0n(0:nmax) * g0n(0:nmax)
750       hib(1:nmax) = g0n(0:nmax-1) * g0n(1:nmax)
751
752       do lb = 1, nsttb
753           if (la /= lb .or. jb /= 1) then
754               gab(la,lb) = sum(hia(0:nrec) * darec(0:nrec,la,lb,jb)) &
755                           & + 2*sum(hib(1:nrec) * dbrec(1:nrec,la,lb,jb))
756               if (term == 1) gab(la,lb) = gab(la,lb) + 2 * hib(nmax) * dbrec(nmax,la,lb,jb)
757           else
758               gab(la,la) = g00(ez, arec(0:nrec,ma), brec(0:nrec,ma), nrec, lainf(ma), lbinf(ma))
759           end if
760       end do
761   end do
762
763   !
764   ! write(390,*) 'ez, ia, ib, jb',ez, ia, ib, jb
765   ! call print_c(gab, 390, 'gab'//achar(10))
766
767   end subroutine get_gab
768
769
770   subroutine gbop(mbc, ez, isp)
771       use ab_io
772       use mod_ham
773       use mod_all_scalar, only : nbase
774
775       type(mbtb_t), intent(inout) :: mbc
776       complex(dp), intent(in) :: ez
777       integer, intent(in) :: isp
778
779       complex(dp) :: gab(mxnstat,mxnstat)
780       complex(dp), allocatable :: g(:, :)
781
782       integer :: ia, ib, la, lb, jb, pa, pb
783
784       allocate(g(mbc % bhsz, mbc % bhsz))
785
786       g = 0.0_dp
787
788       do ia = 1, mbc % n(1)
789           call assoc_ab(ia,isp)
790           call assoc_ham(ia)
791
792           pa = mbc % pbh(ia)
793
794           do jb = 1, pcluster(nbase+1)-1
795               ib = cluster(jb)
796               pb = mbc % pbh(ib)
797

```

```

798      call get_gab(ez, ia, ib, jb, gab)
799
800      do la = 1, mbc % pbh(ia+1) - pa
801          do lb = 1, mbc % pbh(ib+1) - pb
802              g(pa+la,pb+lb) = gab(la,lb)
803          end do
804      end do
805
806      !      write(700,*) 'ia,ib,jb,ez:', ia,ib,jb, ez
807      !      call print_c(gab,700,'')
808      !
809      end do
810  end do
811
812      !      write(600,*) 'ez', ez
813      !      call print_c(g,600,'g'//achar(10),f='f8.3')
814
815      deallocate(g)
816
817  end subroutine gbop
818
819  subroutine a_and_ac(mbc, ez, isp)
820      type(mbtb_t), intent(inout) :: mbc
821      complex(dp), intent(in) :: ez
822      integer, intent(in) :: isp
823
824      !      call gbop(mbc,ez,isp)
825
826
827      call gb_x_dh(mbc,ez,isp)
828      call gtbi(mbc, ez, isp)
829  end subroutine a_and_ac
830
831
832  subroutine mbeq(mbc)
833      use mod_all_scalar, only : mag, quiet, lef, locc, dqmax, nbase
834      use mod_atom_ar
835      use ab_io
836      use mod_ham
837
838      type(mbtb_t), intent(inout) :: mbc
839
840      real(dp) :: ef
841      character(len=25) :: ffmt
842      integer :: ia, jb
843
844      include '../Include/Atom.array' ! for dq, zc, z
845
846      call recurse(.false.)
847
848      do ia = 1, mbc % n(1)
849          call assoc_ham(ia)
850          do jb = 1, pcluster(nbase+1)-1
851              decipher(cluster(jb)) = jb
852          end do
853      end do
854
855      ef = embeff(mbc,locc,lef)
856      lef = ef
857
858      dqmax = 0.0_dp
859      do ia = 1, mbc % nd
860          dq(ia) = dq(ia) - zc(z(ia))
861          if (abs(dq(ia)) > abs(dqmax)) dqmax = dq(ia)
862      end do
863
864
865      if (.not. quiet) then
866          write(ffmt,"(a,i0,a)") '(a,":",', mbc % nd, '(x,f22.14))'
867          write(6,'(a)',advance='no') grnb
868          write(6, ffmt,advance='no') 'dq', dq(: mbc % nd)
869          write(6,'(a)') endc
870          if (mag) write(6, ffmt) 'dm', dem(:mbc % nd)
871      !      if (mag) write(6, ffmt) 'mg', mg(:mbc%nd)

```

```

872     end if
873
874 !       call mbq
875
876 end subroutine mbeq
877
878 function embeff(mbc,locc,ef0) result(ef)
879     use mod_all_scalar, only : totnia
880
881     type(mbtb_t), intent(inout) :: mbc
882     real(dp), intent(in) :: locc, ef0
883     real(dp) :: ef
884     real(dp) :: eflo,efhi,nello,nelhi
885     real(dp) :: nel
886
887
888     real(dp), parameter :: def = 1.0_dp, maxdn = 1.0e-3_dp
889
890
891
892 !       do it = 1, 100
893 !           if (it > 1) nel = mbq(mbc, ef)
894 !
895 !           if (.not. quiet) then
896 !               write(6, '(/, "it:", i3)') it
897 !               write(6, '("ef,nf:", 4(x, f22.14))') ef_prev, ef, dnf_prev, dnf
898 !           end if
899 !
900 !           if (abs(dnf) <= 1.0e-10_dp) exit
901 !           def = dnf * (ef - ef_prev) / ( dnf - dnf_prev )
902 !
903 !           ef_prev = ef
904 !           dnf_prev = dnf
905 !           ef = ef - def
906 !
907 !       end do
908
909
910     ef = ef0
911     nel = mbq(mbc, ef)
912     print *, 'ef,nel:', ef,nel
913     if (nel < locc) then
914         eflo = ef
915         nello = nel
916         efhi = eflo + def
917     1       nelhi = mbq(mbc, efhi)
918         print *, 'efhi,nelhi:', efhi,nelhi
919         if (nelhi < locc) then
920             eflo = efhi
921             nello = nelhi
922             efhi = efhi+def
923             goto 1
924         endif
925     else
926         efhi = ef
927         nelhi = nel
928         eflo = efhi - def
929     4       nello = mbq(mbc, eflo)
930         print *, 'eflo,nello:', eflo,nello
931         if (nello > locc) then
932             efhi = eflo
933             nelhi = nello
934             eflo = eflo-def
935             goto 4
936         endif
937     endif
938
939
940 !       Use binary sections to find the Fermi energy.
941
942     2       ef = (eflo + efhi)*0.5_dp
943         nel = mbq(mbc, ef)
944         print *, 'ef,nel:', ef,nel
945         if (nel > locc) then

```

```

946     nelhi = nel
947     efhi = ef
948     else
949         nello = nel
950         eflo = ef
951     endif
952     if (abs(nel-locc) > maxdn) goto 2
953
954     totnia = nel
955
956 end function embeff
957
958
959 function mbq(mbc, ef) result(cnel)
960     use mod_all_scalar, only : kt, mfac, nsp
961     use mod_atom_ar
962
963     type(mbtb_t), intent(inout) :: mbc
964     real(dp), intent(in) :: ef
965     integer :: isp, la, ia
966
967     real(dp) :: cnel
968
969     complex(dp) :: zp, zfac, zep
970     real(dp) :: phase, w0
971     integer :: m, p
972
973
974 !     M      = NINT(MFAC*0.25D0*(EF-LAINF(LA)+2.0D0*LBINF(LA))/KT)
975 !     m      = nint(mfac*0.25_dp*(ef-lainf(la)+2*lbinf(la))/kt)
976 !     print *,ef, lainf(la), lbinf(la), (ef-lainf(la)+2.0_dp*lbinf(la))
977 !     m      = nint(mfac*0.25_dp*1.8_dp/kt)
978     m = mfac
979
980     print *, 'm:', m
981 !     write(100+iproc,*) ef, lainf(la), lbinf(la)
982 !     write(100+iproc,*) ' ', ef, arec(0:lchain(la),la), brec(0:lchain(la)+1,la)
983
984     dq = 0.0_dp
985     do isp = 1, nsp
986         mbc % q = 0.0_dp
987         mbc % ggd = 0.0_dp
988
989         w0 = kt*(2*m)
990         phase = pi/real(2*m, dp)
991         zp = cmplx(cos(phase), sin(phase), kind=dp)
992         zfac = zp*zp
993
994         do p = 0, m-1
995 !             print *, 'p', p
996             zep = ef+w0*(zp-1)
997
998             call gg_diag(mbc, zep, isp)
999
1000             do ia = 1, mbc % nd
1001 !                 print *, 'ia:', ia
1002                 do la = mbc%pah(ia)+1, mbc%pah(ia+1)
1003 !                     print *, ' ', la,g:', la-mbc%pah(ia), mbc % ggd(la)
1004                     mbc % q(ia) = mbc % q(ia) + real(zp * mbc % ggd(la))
1005                 end do
1006             end do
1007
1008             zp = zfac*zp
1009         enddo
1010
1011         mbc % q = 4 * kt * mbc % q
1012
1013         dq(:mbc%nd) = dq(:mbc%nd) + mbc % q
1014
1015         if (isp /= 1) then ! mag == true and isp == 2
1016             dq(:mbc%nd) = 0.5_dp * dq(:mbc%nd)
1017             mg(:mbc%nd) = dq(:mbc%nd) - mbc%q(:mbc%nd)
1018         end if
1019
1020

```

```

1020     end do
1021
1022     cnel = sum(dq(:mbc%nd))
1023
1024 !     stop
1025 end function mbq
1026
1027
1028
1029 subroutine mbrho(mbc, ef)
1030     use mod_all_scalar, only : kt, mfac, nsp
1031     use mod_atom_ar
1032
1033     type(mbtb_t), intent(inout) :: mbc
1034     real(dp), intent(in) :: ef
1035     integer :: isp, la, ia
1036
1037     real(dp) :: cnel
1038
1039     complex(dp) :: zp, zfac, zep
1040     real(dp) :: phase, w0
1041     integer :: m, p
1042
1043
1044 !     M      = NINT(MFAC*0.25D0*(EF-LAINF(LA)+2.0D0*LBINF(LA))/KT)
1045 !     m      = nint(mfac*0.25_dp*(ef-lainf(la)+2*lbinf(la))/kt)
1046 !     print *,ef, lainf(la), lbinf(la), (ef-lainf(la)+2.0_dp*lbinf(la))
1047 !     m      = nint(mfac*0.25_dp*1.8_dp/kt)
1048     m = mfac
1049
1050     print *, 'm:', m
1051 !     write(100+iprocc,*) ef, lainf(la), lbinf(la)
1052 !     write(100+iprocc,*) ' ', ef, arec(0:lchain(la),la), brec(0:lchain(la)+1,la)
1053
1054     mbc % rho = 0.0_dp
1055
1056     do isp = 1, nsp
1057
1058         w0      = kt*(2*m)
1059         phase = pi/real(2*m, dp)
1060         zp      = cmplx(cos(phase), sin(phase), kind=dp)
1061         zfac    = zp*zp
1062
1063         do p = 0, m-1
1064 !             print *, 'p', p
1065             zep = ef+w0*(zp-1)
1066             call gg_rho(mbc, zep, zp, isp)
1067             zp  = zfac*zp
1068         enddo
1069
1070 !         mbc % q = 4 * kt * mbc % q
1071 !
1072 !         dq(:mbc%nd) = dq(:mbc%nd) + mbc % q
1073 !
1074 !         if (isp /= 1) then ! mag == true and isp == 2
1075 !             dq(:mbc%nd) = 0.5_dp * dq(:mbc%nd)
1076 !             mg(:mbc%nd) = dq(:mbc%nd) - mbc%q(:mbc%nd)
1077 !         end if
1078
1079     end do
1080
1081 end subroutine mbrho
1082
1083
1084
1085
1086
1087
1088
1089 subroutine areduce(n, pos, full, cnds)
1090     integer, intent(in) :: n, pos(1:)
1091     real(dp), intent(in) :: full(1:)
1092     real(dp), intent(out) :: cnds(1:)
1093

```

```

1094     integer :: i
1095
1096     do i = 1, n
1097         cnds(i) = cnds(i) + sum(full(pos(i)+1 : pos(i+1)))
1098     end do
1099
1100 end subroutine areduce
1101
1102
1103 subroutine mblldh(mbc)
1104     use ab_io , only : init_ab , free_ab , assoc_ab
1105     use mod_ham, only : init_ham, free_ham, assoc_ham
1106     use mod_chi, only : init_chi, free_chi
1107     use mod_all_scalar, only : nsp, nd, quiet
1108     use topologia, only : mpmap, iproc
1109
1110
1111     type(mbtb_t), intent(inout) :: mbc
1112
1113     integer :: ia, ja, jb, p, mp, ib, ir
1114     integer :: loc_clusiz
1115
1116     include "../Include/PosVel.array"
1117     include "../Include/NebList.array"
1118
1119     mbc % nd = mbc % rlst(nr+1)-1
1120     do ir = 1, nr
1121         mbc % n(ir) = mbc%rlst(ir+1) - mbc%rlst(ir)
1122     end do
1123
1124     if (.not. allocated(mbc % ad)) allocate(mbc % ad(3, mbc % nd))
1125
1126     if (.not. allocated(mbc % apb)) allocate(mbc % apb(mbc % n(1)+1))
1127     if (.not. allocated(mbc % bpb)) allocate(mbc % bpb(mbc % n(1)*(mxnnb+1)))
1128
1129     if (.not. allocated(mbc % apt)) allocate(mbc % apt(mbc % n(2)+1))
1130     if (.not. allocated(mbc % bpt)) allocate(mbc % bpt(mbc % n(2)*(mxnnb+1)))
1131
1132     if (.not. allocated(mbc % apd)) allocate(mbc % apd(mbc % n(2)+1))
1133     if (.not. allocated(mbc % bpd)) allocate(mbc % bpd(mbc % n(2)*(mxnnb))) ! there will be at least
1134     one neighbour in its own region
1135
1136     mbc % ad = ad(:, :mbc % nd)
1137
1138
1139     call extract_neblist(aptr, bptr, 1, 1, mbc % rlst, mbc % apb, mbc % bpb)
1140     call extract_neblist(aptr, bptr, 2, 2, mbc % rlst, mbc % apt, mbc % bpt)
1141     call extract_neblist(aptr, bptr, 2, 1, mbc % rlst, mbc % apd, mbc % bpd) ! list of atoms from r2
1142     which have neighbours in r1
1143
1144     print *, 'in'
1145     call print_neblist(1, mbc % nd, aptr, bptr)
1146     print *, 'out, 2->1'
1147     call print_neblist(1, mbc % n(2), mbc % apd, mbc % bpd )
1148     print *, 'out, 2'
1149     call print_neblist(1, mbc % n(2), mbc % apt, mbc % bpt )
1150
1151 !     stop
1152 !     BOP r1
1153     nd = mbc % n(1)
1154     mpmap(iproc) = 0
1155     mpmap(iproc+1) = mbc % n(1)
1156
1157     aptr(:mbc % n(1)+1) = mbc % apb
1158     bptr(:mbc % apb(mbc%n(1)+1)) = mbc % bpb(:mbc % apb(mbc%n(1)+1))
1159
1160     call init_ab (1, mbc % n(1), nsp)
1161 !     print *, 'init_ab done'
1162     call init_ham(1, mbc % n(1))
1163 !     print *, 'init_ham done'
1164
1165     do ia = 1, mbc % n(1) ! bop is r1

```



```

1166      call assoc_ab(ia,1)
1167      call assoc_ham(ia)
1168      call getnch(ia) !Find number of linear chains per site.
1169      call bldclus(ia)
1170 !      loc_clusiz = loc_clusiz + pcluster(nbase+1)-1
1171      call bldh()
1172  end do
1173 !
1174 ! End BOP r1
1175
1176
1177 !      TB r2
1178
1179  if (.not. allocated(mbc % pbh)) allocate(mbc % pbh( mbc % n(1)+1))
1180  call mpos(mbc%rlst(1), mbc%rlst(2)-1, mbc%pbh)
1181  mbc % bhsz = mbc % pbh(mbc % n(1)+1)
1182 !      print *, 'pbh', mbc % pbh
1183
1184  if (.not. allocated(mbc % pth)) allocate(mbc % pth( mbc % n(2)+1))
1185  call mpos(mbc%rlst(2), mbc%rlst(3)-1, mbc%pth)
1186  mbc % thsz = mbc % pth(mbc % n(2)+1)
1187  print *, 'pth', mbc % pth
1188
1189  mbc % ahsz = mbc % bhsz + mbc % thsz
1190  if (.not. allocated(mbc % pah)) allocate(mbc % pah( mbc % nd+1))
1191  mbc % pah(:mbc % rlst(2)) = mbc % pbh
1192  mbc % pah(mbc % rlst(2) : mbc % rlst(3)) = mbc%bhsz + mbc % pth
1193 !      print *, 'pah', mbc % pah
1194 !      print *, 'thsz:', mbc % thsz
1195
1196 !      stop
1197
1198  print *, 'thsz:', mbc%thsz
1199  if (.not. allocated(mbc%h)) allocate(mbc%h(mbc%thsz,mbc%thsz))
1200 !      print *, 'h'
1201  if (.not. allocated(mbc%dh)) allocate(mbc%dh(mxnstat,mxnstat,mbc%apd(mbc%n(2)+1)))
1202 !      print *, 'dh'
1203  if (mbc%ovl .and. (.not. allocated(mbc%s))) allocate(mbc%s(mbc%thsz,mbc%thsz))
1204
1205 !      print *, 's'
1206
1207  call bldmtbmat(mbc % rlst(2), mbc % rlst(3)-1, mbc % pth, mbc % apt, mbc % bpt, mbc % h)
1208  if (mbc%ovl) call bldmtbmat(mbc % rlst(2), mbc % rlst(3)-1, mbc % pth, mbc % apt, mbc % bpt, mbc
% s)
1209  call bldsparsedh(mbc % apd, mbc % bpd, 2, 1, mbc % rlst, mbc%dh )
1210
1211 !      a(:,:), ac(:,:), aca(:,:), v(:), w(:,:), occ(:)
1212
1213
1214  if (.not. allocated(mbc%a)) allocate(mbc % a(mbc % bhsz, mbc % thsz))
1215  if (.not. allocated(mbc%ac)) allocate(mbc% ac(mbc % bhsz, mbc % thsz))
1216 !      if (.not. allocated(mbc%v)) allocate(mbc % v(mbc % thsz))
1217 !      if (.not. allocated(mbc%occ)) allocate(mbc % occ(mbc % thsz))
1218 !      if (.not. allocated(mbc%w)) allocate(mbc % w(mbc % thsz, mbc % thsz))
1219  if (.not. allocated(mbc%gt)) allocate(mbc % gt(mbc % thsz, mbc % thsz))
1220  if (.not. allocated(mbc%ggd)) allocate(mbc%ggd(mbc % ahsz))
1221  if (.not. allocated(mbc%q)) allocate(mbc%q(mbc % nd))
1222
1223
1224  end subroutine mbldh
1225
1226
1227  subroutine mpos(bgn, fin, p)
1228      integer, intent(in) :: bgn, fin
1229      integer, intent(out) :: p(:)
1230      integer :: ia
1231 !      calculate offsets in matrices using the common nstt.
1232 !      To embed atoms of the same type but with different number of orbitals it may be better if the
conf_t structures are used
1233
1234      include "../Include/Atom.array"
1235
1236      p(1) = 0
1237      do ia = 2, fin - bgn + 2 ! go one extra

```

```

1238      p(ia) = p(ia-1) + nstt(z( bgn - 2 + ia)) ! gia = off + ia; gia: global ia
1239      enddo
1240
1241  end subroutine mpos
1242
1243
1244  subroutine bldsparsedh(ap, bp, r1, r2, lst, h)
1245 !    build sparse dH using the neighbour table
1246
1247      integer, intent(in) :: ap(:), bp(:), r1, r2, lst(:)
1248      real(dp), intent(out) :: h(:, :, :)
1249
1250      integer :: i, j, off1, off2, iap, ibp
1251      integer :: za, zb
1252      integer :: ja, ja0, ia, ib, la, lb, gia, gib
1253      integer :: nstta, nsttb
1254
1255      real(dp) :: dr(3), dea
1256      real(dp) :: subh(mxnstat, mxnstat)
1257
1258      real(dp) :: scfcut(14)
1259
1260      include "../Include/Atom.array" ! for z mainly
1261      include "../Include/PosVel.array"
1262
1263
1264      dea = 0.0_dp
1265      scfcut = 1.0_dp
1266
1267
1268      off1 = lst(r1)-1
1269      off2 = lst(r2)-1
1270
1271      do gia = lst(r1), lst(r1+1)-1
1272          za = z(gia)
1273          nstta = nstt(za)
1274
1275          ja = ap(gia-off1) ! ia = gia-off1
1276          ja0 = ja - 1
1277          do while (bp(ja) /= eol)
1278              ib = bp(ja)
1279              gib = off2 + ib
1280              zb = z(gib)
1281              nsttb = nstt(zb)
1282
1283              dr = ad(:, gia)-ad(:, gib)
1284 !              print *, 'sparse: gia, gib, ia, ib, ja:', gia, gib, ia, ib, ja
1285              call matel(za, zb, dr, h(:, :, ja), dea, scfcut)
1286
1287              ja = ja+1
1288          end do
1289      end do
1290  end subroutine bldsparsedh
1291
1292
1293  subroutine bldmtbmat(bgn, fin, ph, ap, bp, h)
1294
1295 !    Build the full size embedded molecular TB Hamiltonian matrix
1296
1297      integer, intent(in) :: bgn, fin, ph(:), ap(:), bp(:)
1298      real(dp), intent(out) :: h(:, :, :)
1299
1300      integer :: i, j, off, iap, ibp
1301      integer :: za, zb
1302      integer :: ja, ja0, ia, ib, la, lb, gia, gib
1303      integer :: nstta, nsttb
1304
1305      real(dp) :: dr(3), dea
1306      real(dp) :: subh(mxnstat, mxnstat)
1307
1308
1309      real(dp) :: scfcut(14)
1310
1311      include "../Include/Atom.array" ! for z mainly

```

```
1312     include "../Include/PosVel.array"
1313
1314
1315     dea = 0.0_dp
1316     scfcut = 1.0_dp
1317
1318     h(:, :) = 0.0_dp
1319
1320     off = bgn - 1
1321
1322     do ia = 1, fin - off
1323         gia = off + ia
1324         za = z(gia)
1325         nstta = nstt(za)
1326
1327         ja = ap(ia)
1328         ja0 = ja - 1
1329         do while (bp(ja) /= eol)
1330             ib = bp(ja)
1331             gib = off + ib
1332             zb = z(gib)
1333             nsttb = nstt(zb)
1334
1335             dr = ad(:, gia) - ad(:, gib)
1336
1337             call matel(za, zb, dr, subh, dea, scfcut)
1338
1339             iap = ph(ia)
1340             ibp = ph(ib)
1341
1342             h(iap+1:iap+nstta, ibp+1:ibp+nsttb) = subh(:nstta, :nsttb)
1343
1344             ja = ja+1
1345         end do
1346     end do
1347
1348 end subroutine bldmtbmat
1349
1350
1351 subroutine swap(n, a, b)
1352 !     utility routine to test different swap algorithms
1353
1354     integer, intent(in) :: n
1355     integer, intent(inout) :: a(n), b(n)
1356
1357     integer :: i, t
1358
1359     do i = 1, n
1360         t = a(i)
1361         a(i) = b(i)
1362         b(i) = t
1363     end do
1364
1365 !     do i = 1, n
1366 !         a(i) = ieor(a(i), b(i))
1367 !         b(i) = ieor(a(i), b(i))
1368 !         a(i) = ieor(a(i), b(i))
1369 !     end do
1370
1371
1372 end subroutine swap
1373
1374
1375
1376
1377 subroutine extract_neblist(api, bpi, r1, r2, lst, apo, bpo)
1378 !     extracts the output position tables apo, bpo from the input ones api, bpi, for the atoms listed
in lst,
1379 !     such that the first atom of a bond is bound to be in region r1 and the second one in r2
1380
1381     integer, intent(in) :: api(:), bpi(:), r1, r2, lst(:)
1382     integer, intent(out) :: apo(:), bpo(:)
1383
1384     integer :: ia, ib, ja, jb, p, mp, prevmp
```

```

1385
1386     mp = 1
1387     do ia = lst(r1), lst(r1+1) - 1
1388         ja = ia - lst(r1) + 1
1389         apo(ja) = mp
1390
1391         jb = api(ia)
1392         p = 0
1393         ib = bpi(jb+p)
1394         do while (ib /= eol)
1395             if (rmap(ib, lst) == r2) then
1396                 bpo(mp) = ib - lst(r2) + 1
1397                 mp = mp + 1
1398             end if
1399             p = p + 1
1400             ib = bpi(jb+p)
1401         end do
1402         bpo(mp) = eol
1403
1404         if (bpo(max(mp-1,1)) /= eol) then
1405             mp = mp + 1
1406         else
1407             apo(ja) = apo(ja)-1
1408         end if
1409     end do
1410
1411     apo(lst(r1+1)- lst(r1) + 1) = mp-2
1412
1413     print *, 'apo:', apo
1414     print *, 'bpo:', bpo
1415
1416
1417 end subroutine extract_neblist
1418
1419
1420
1421 subroutine print_neblist(s,e,a,b)
1422 !   print the position tables ap, bp from starting atom s to endind atom e
1423
1424     integer, intent(in) :: s,e, a(:), b(:)
1425
1426     integer :: ia, ja, p, ib
1427
1428     do ia = s, e
1429         print *, 'ia:', ia
1430         ja = a(ia)
1431         p = 0
1432         ib = b(ja+p)
1433         do while(ib/=eol)
1434             print *, '    p, ib', p, ib
1435             p = p + 1
1436             ib = b(ja+p)
1437         end do
1438     end do
1439
1440 end subroutine print_neblist
1441
1442
1443 function rmap(ia, rlst)
1444 !   range map, return the range in which atom ia resides according to the range's list rlst
1445
1446     integer, intent(in) :: ia, rlst(:)
1447     integer :: rmap
1448 !   To be generalised in the future. For now there are only 2 blocks
1449
1450     if (ia < rlst(2)) then
1451         rmap = 1
1452     else if (ia >= rlst(2) .and. ia < rlst(3)) then
1453         rmap = 2
1454     else
1455         rmap = 3
1456     end if
1457
1458 end function rmap

```

1459

1460 **end module** tbbop_emb

1461