sievgene モジュール仕様書

2018 年 2 月 5 日

－ 目次 －

各モジュールの仕様は以下の項目で構成する。なお、型の記述はFortran90の文法に従う。

項番	項目	概要
#1	呼び出し形式	メソッドの種別(手続き種別)と引数を規定する。
#2	引数	手続きの仮引数を規定する。
#3	戻り値	当該メソッドが関数である場合の戻り値を規定する。
#4	機能	当該メソッドの機能概要を規定する。

1.Conf_Method

ファイル名：Conf_Method.f90

conformer 生成のメソッドを定義する。

1.1.Conformer_Main

呼び出し形式：

```
subroutine Conformer_Main(conf,numatom,ato,co,nat,hank,charge,              &
                          nbond,bondorder,numbond,nframe,rflag,             &
                          no_conf,confomer,mat,rotNum,                      &
                          sybylType, substID, substName, isReadCharge,      &
                          atomStatus, bondStatus)
```

引数：

```
  type(Conf_t),intent(in)::conf                                  ! conformer 情報
  integer*4,intent(in)::numatom                                  ! リガンド原子数
  character*1,intent(out),dimension(MAX_ATOM)::ato               ! 原子名の１文字目
  real*4,intent(inout),dimension(MAX_ATOM,3)::co                 ! 原子座標
  integer*4,intent(out),dimension(MAX_ATOM)::nat                 ! 原子タイプ
  real*4,intent(out),dimension(MAX_ATOM)::hank                   ! 原子半径
  real*4,intent(out),dimension(MAX_ATOM)::charge                 ! 原子電荷
  integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond           ! bond 対象原子 ID ペア
  real*4,intent(inout),dimension(MAX_ATOM)::bondorder            ! bond の結合度
  integer*4,intent(inout)::numbond                               ! bond 数
  integer*4,intent(out),dimension(MAX_ATOM,3)::nframe            ! 原子フレームの原子 ID
  real*4,intent(out),dimension(MAX_LIGAND,MAX_LIGAND)::rflag   ! 環情報マトリックス
  integer*4,intent(out)::no_conf                                 ! 生成 conformer 数
  real*4,intent(out),dimension(MAX_LIGAND,3,MAX_CONF)::confomer
                                                                 ! conformer 座標
  integer*4,intent(out),dimension(MAX_ATOM)::mat                 ! 各原子の原子番号
  integer,intent(out)::rotNum                                    ! 回転可能結合数
  character*5,intent(inout),dimension(MAX_ATOM)::sybylType ! SYBYL 原子タイプ
  integer*4,intent(inout),dimension(MAX_ATOM)::substID           ! 残基番号
  character*10,intent(inout),dimension(MAX_ATOM)::substName! 残基名
  logical*4,intent(inout),dimension(MAX_ATOM)::isReadCharge! 電荷読み込みフラグ
  integer*4,intent(inout),dimension(MAX_ATOM)::atomStatus  ! 原子情報
  integer*4,intent(inout),dimension(MAX_ATOM)::bondStatus  ! 結合情報
```

戻り値：

　なし

機能：

　リガンドの conformer 作成の主制御を行う。

1.2.Make_Frame

呼び出し形式：

```
  subroutine Make_Frame(numatom,numbond,nbond,nframe)
```

引数：

```
  integer*4,intent(in)::numatom                              ! 原子数
  integer*4,intent(in)::numbond                              ! bond 数
  integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond       ! bond 対象原子 ID ペア
  integer*4,intent(out),dimension(MAX_ATOM,3)::nframe        ! 原子フレームの原子 ID
```

戻り値：

　なし

機能：

全原子の Z-matrix のフレーム情報を作成する。

原子フレームは 1-2,1-3,1-4 原子の ID および 1-2 の距離、1-2-3 の角度(angle 角)、1-2-3-4 の角度(torsion 角)の情報で構成され、ここでは原子 ID のみ定義する。

1.3.Sort_AtomAndBond

呼び出し形式：

```
  subroutine Sort_AtomAndBond(numatom,numbond,n_seria,number,                &
                              co,nat,hank,charge,ato,nbond,bondorder,mat ,   &
                              sybylType, substID, substName, isReadCharge,   &
                              atomStatus, bondStatus, isUnited)
```

引数：

```
  integer*4,intent(in)::numatom                              ! 原子数
  integer*4,intent(in)::numbond                              ! bond 数
  integer*4,intent(in),dimension(MAX_ATOM)::n_seria          ! 逐次番号の原子 ID の index
  integer*4,intent(in),dimension(MAX_ATOM)::number           ! 当該原子のソート後の ID
  real*4,intent(inout),dimension(MAX_ATOM,3)::co             ! 原子座標
  integer*4,intent(inout),dimension(MAX_ATOM)::nat           ! 原子タイプ
  real*4,intent(inout),dimension(MAX_ATOM)::hank             ! 原子半径
  real*4,intent(inout),dimension(MAX_ATOM)::charge           ! 原子電荷
  character*1,intent(inout),dimension(MAX_ATOM)::ato         ! 原子名の１文字目
  integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond       ! bond ペアの原子 ID
  real*4,intent(inout),dimension(MAX_ATOM)::bondorder        ! bond の結合度
  integer*4,intent(inout),dimension(MAX_ATOM)::mat           ! 元素番号
  character*5,intent(inout),dimension(MAX_ATOM)::sybylType   ! SYBYL 原子タイプ
  integer*4,intent(inout),dimension(MAX_ATOM)::substID       ! 残基番号
  character*10,intent(inout),dimension(MAX_ATOM)::substName  ! 残基名
  logical*4,intent(inout),dimension(MAX_ATOM)::isReadCharge  ! 電荷読み込みフラグ
  integer*4,intent(inout),dimension(MAX_ATOM)::atomStatus    ! 原子情報
  integer*4,intent(inout),dimension(MAX_ATOM)::bondStatus    ! 結合情報
  logical,intent(in)::isUnited                               ! 原子モデル
```

戻り値：

　なし

機能：

原子情報、bond 情報の並べ替えを行う

1.4.Sort_ConformerInfo

呼び出し形式：

```
  subroutine Sort_ConformerInfo(numatom,number,co,                          &
                                no_conf,confomer,nat,hank,charge,ato,mat,    &
                                sybylType, substID, substName, isReadCharge, &
                                atomStatus, bondStatus, isUnited)
```

引数：

```
  integer*4,intent(in)::numatom                                ! 原子数
  integer*4,intent(in),dimension(MAX_ATOM)::number             ! 当該原子のソート後の ID
  real*4,intent(inout),dimension(MAX_ATOM,3)::co               ! 原子座標
  integer*4,intent(in)::no_conf                                ! conformer 数
  real*4,intent(inout),dimension(MAX_LIGAND,3,MAX_CONF)::confomer
                                                               ! conformer 座標
  integer*4,intent(inout),dimension(MAX_ATOM)::nat             ! 原子タイプ
  real*4,intent(inout),dimension(MAX_ATOM)::hank               ! 原子半径
  real*4,intent(inout),dimension(MAX_ATOM)::charge             ! 原子電荷
  character*1,intent(inout),dimension(MAX_ATOM)::ato           ! 原子名の１文字目
  integer*4,intent(inout),dimension(MAX_ATOM)::mat             ! 元素番号
  character*5,intent(inout),dimension(MAX_ATOM)::sybylType     ! SYBYL 原子タイプ
  integer*4,intent(inout),dimension(MAX_ATOM)::substID         ! 残基番号
  character*10,intent(inout),dimension(MAX_ATOM)::substName    ! 残基名
  logical*4,intent(inout),dimension(MAX_ATOM)::isReadCharge    ! 電荷読み込みフラグ
  integer*4,intent(inout),dimension(MAX_ATOM)::atomStatus      ! 原子情報
  integer*4,intent(inout),dimension(MAX_ATOM)::bondStatus      ! 結合情報
  logical,intent(in)::isUnited                                 ! 原子モデル
```

戻り値：

なし

機能：

conformer の原子情報の並べ替えを行う。

1.5.Get_SortIndex

呼び出し形式：

```
  subroutine Get_SortIndex(numatom,number,number_rev)
```

引数：

```
  integer*4,intent(in)::numatom                          ! 原子数
  integer*4,intent(in),dimension(MAX_ATOM)::number       ! 当該原子のソート後の ID
  integer*4,intent(out),dimension(MAX_ATOM)::number_rev  ! 逐次番号の原子 ID の index
```

戻り値：

　なし

機能：

当該原子のソート後の ID から、逐次番号に対応する原子 ID のリストを作成する。

1.6.Get_AtomSortResult

呼び出し形式：

```
  subroutine Get_AtomSortResult(numatom,numbond,nbond,n_seria,number,n_start)
```

引数：

```
  integer*4,intent(in)::numatom                          ! 原子数
  integer*4,intent(inout)::numbond                       ! bond 数
  integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond   ! bond ペアの原子 ID
  integer*4,intent(out),dimension(MAX_ATOM)::n_seria     ! 当該原子のソート後の ID
  integer*4,intent(out),dimension(MAX_ATOM)::number      ! 当該逐次番号の原子 ID
  integer*4,intent(out)::n_start                         ! ソートを開始する原子の ID
```

戻り値：

　なし

機能：

木構造に従って原子のソートを行う。

1.7.Set_TreeDepth

呼び出し形式：

```
  subroutine Set_TreeDepth(numatom,n_start,                                    &
                           nbond_a,lbond_a,lengh_a,number,n_seria)
```

引数：

```
  integer*4,intent(in)::numatom                  ! 原子数
  integer*4,intent(inout)::n_start               ! 木構造の先頭原子 ID
  integer*4,intent(in),dimension(MAX_ATOM)::nbond_a
                                                 ! 各原子の bond 数
  integer*4,intent(in),dimension(MAX_ATOM,9)::lbond_a
                                                 ! 各原子の bond 相手原子 ID
  integer*4,intent(in),dimension(MAX_LIGAND,MAX_LIGAND)::lengh_a
                                                 ! 原子間の経路長
  integer*4,intent(inout),dimension(MAX_ATOM)::number
                                                 ! 当該逐次番号の原子 ID
  integer*4,intent(out),dimension(MAX_ATOM)::n_seria
                                                 ! 当該原子のソート後の ID
```

戻り値：

　なし

機能：

先頭原子を根とした木構造での番号を設定する。

1.8.Change_UnitedAtom

呼び出し形式：

```
subroutine Change_UnitedAtom(numatom,cseq,ato,co,nat,                    &
                             hank,charge,nbond,bondorder,numbond,mat)
```

引数：

```
integer*4,intent(inout)::numatom                          ! 原子数
character*3,intent(inout),dimension(MAX_ATOM)::cseq       ! 残基名
character*1,intent(inout),dimension(MAX_ATOM)::ato        ! 原子名の１文字目
real*4,intent(inout),dimension(MAX_ATOM,3)::co            ! 原子座標
integer*4,intent(inout),dimension(MAX_ATOM)::nat          ! 原子タイプ
real*4,intent(inout),dimension(MAX_ATOM)::hank            ! 原子半径
real*4,intent(inout),dimension(MAX_ATOM)::charge          ! 原子電荷
integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond      ! bond ペアの原子 ID
real*4,intent(inout),dimension(MAX_ATOM)::bondorder       ! bond の結合度
integer*4,intent(inout)::numbond                          ! bond 数
integer*4,intent(inout),dimension(MAX_ATOM)::mat          ! 元素番号
```

戻り値：

なし

機能：

C-H を一つの原子にまとめる。

1.9.Set_AtomInfo

呼び出し形式：

```
subroutine Set_AtomInfo(numatom,mat,ato2)
```

引数：

```
integer*4,intent(in)::numatom                         ! 原子数
integer*4,intent(in),dimension(MAX_ATOM)::mat         ! 元素番号
character*2,intent(out),dimension(MAX_ATOM)::ato2     ! 原子名
```

戻り値：

なし

機能：

元素番号から原子名を割り当てる。

1.10.Make_BondList

呼び出し形式：

```
subroutine Make_BondList(numatom,numbond,nbond,                    &
                         nbond_a,mbond_a,nweig_a,lbond_a,lengh_a)
```

引数：

```
integer*4,intent(in)::numatom                    ! 原子数
integer*4,intent(in)::numbond                    ! bond 数
integer*4,intent(in),dimension(MAX_ATOM,2)::nbond    ! bond ペアの原子 ID
integer*4,intent(out),dimension(MAX_ATOM)::nbond_a
                                                 ! 各原子の bond 数
integer*4,intent(out),dimension(MAX_ATOM)::mbond_a
                                                 ! 処理終了フラグ
integer*4,intent(out),dimension(MAX_ATOM)::nweig_a
                                                 ! 当該原子の重み
integer*4,intent(out),dimension(MAX_ATOM,9)::lbond_a
                                                 ! 各原子の bond 相手原子 ID
integer*4,intent(out),dimension(MAX_LIGAND,MAX_LIGAND)::lengh_a
                                                 ! 原子間の経路長
```

戻り値：

なし

機能：

bond ペア情報から、各原子の bond 相手原子のリストを作成する。

1.11.Make_TreeStructure

呼び出し形式：

```
subroutine Make_TreeStructure(numatom,nbond_a,mbond_a,nweig_a,lbond_a,lengh_a)
```

引数：

```
integer*4,intent(in)::numatom                ! 原子数
integer*4,intent(in),dimension(MAX_ATOM)::nbond_a
                                             ! 各原子の bond 数
integer*4,intent(in),dimension(MAX_ATOM)::mbond_a
                                             ! 処理終了フラグ
integer*4,intent(inout),dimension(MAX_ATOM)::nweig_a
                                             ! 当該原子の重み
integer*4,intent(in),dimension(MAX_ATOM,9)::lbond_a
                                             ! 各原子の bond 相手原子 ID
integer*4,intent(inout),dimension(MAX_LIGAND,MAX_LIGAND)::lengh_a
                                             ! 原子間の経路長
```

戻り値：

なし

機能：

原子間の機構増情報を作成する。

1.12.Make_OpenedStructure

呼び出し形式：

```
  subroutine Make_OpenedStructure(numatom,numbond,                              &
                                  nbond_a,mbond_a,lbond_a,mm,list_cut)
```

引数：

```
  integer*4,intent(in)::numatom                  ! 原子数
  integer*4,intent(out)::numbond                 ! bond 数
  integer*4,intent(in),dimension(MAX_ATOM)::nbond_a
                                                 ! 各原子の bond 数
  integer*4,intent(in),dimension(MAX_ATOM)::mbond_a     ! 処理終了フラグ
  integer*4,intent(in),dimension(MAX_ATOM,9)::lbond_a  ! 各原子の bond 相手原子 ID
  integer*4,intent(out)::mm                      ! 開環 bond 数
  integer*4,intent(out),dimension(MAX_LIGAND,2)::list_cut
                                                 ! 開環 bond の原子 ID
```

戻り値：

　なし

機能：

　当該分子の環を開く bond を登録する。

1.13.Generate_Conf

呼び出し形式：

```
  subroutine Generate_Conf(numatom,nframe,rflag,no_conf,                        &
                           co,confomer,ato,rotNum)
```

引数：

```
  integer*4,intent(in)::numatom                          ! 原子数
  integer*4,intent(in),dimension(MAX_ATOM,3)::nframe     ! 原子フレーム情報
  real*4,intent(in),dimension(MAX_LIGAND,MAX_LIGAND)::rflag   ! 環情報
  integer*4,intent(inout)::no_conf                       ! conformer 数
  real*4,intent(inout),dimension(MAX_ATOM,3)::co         ! 原子座標
  real*4,dimension(MAX_LIGAND,3,MAX_CONF)::confomer      ! conformar 座標
  character*1,intent(in),dimension(MAX_ATOM)::ato        ! 原子名の１文字目
  integer*4,intent(out)::rotNum                          ! 回転可能結合数
```

戻り値：

　なし

機能：

　conformer の座標設定を行う。

1.14.Check_VdwContact

呼び出し形式：

```
  subroutine Check_VdwContact(atomNum,co,ato,rflag,atom_is_contact)
```

引数：

```
  integer*4,intent(in):: atomNum                          ! 原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co             ! 原子座標
  character*1,intent(in),dimension(MAX_ATOM)::ato         ! 原子名１文字目
  real*4,intent(in),dimension(MAX_LIGAND,MAX_LIGAND)::rflag  ! 環情報
  logical,intent(out)::atom_is_contact                    ! van der Waals 接触の有無
```

戻り値：

　なし

機能：

当該原子座標での van der Waals 接触を検出する。

1.15.Make_NewCord

呼び出し形式：

```
  subroutine Make_NewCord(atomNum,N1,N2,N3,R,A,D,co)
```

引数：

```
  integer*4,intent(in):: atomNum                    ! 原子数
  integer*4,intent(in),dimension(999)::N1           ! フレームの 1-2 原子 ID
  integer*4,intent(in),dimension(999)::N2           ! フレームの 1-3 原子 ID
  integer*4,intent(in),dimension(999)::N3           ! フレームの 1-4 原子 ID
  real*4,intent(in),dimension(999)::R               ! 1-2 距離
  real*4,intent(in),dimension(999)::A               ! 1-2-3 角(angle)
  real*4,intent(in),dimension(999)::D               ! 1-2-3-4 角(torsion)
  real*4,intent(inout),dimension(MAX_ATOM,3)::co    ! 生成した原子座標
```

戻り値：

　なし

機能：

フレーム情報と Z-matrix 情報から、原子の座標を生成する。

1.16.Set_RingStructure

呼び出し形式：

```
  subroutine Set_RingStructure(numatom,numbond,ato,nbond,bondorder,rflag)
```

引数：

```
  integer*4,intent(in)::numatom                           ! 原子数
  integer*4,intent(in)::numbond                           ! bond 数
  character*1,intent(in),dimension(MAX_ATOM)::ato         ! 原子名１文字目
  integer*4,intent(in),dimension(MAX_ATOM,2)::nbond       ! bond ペアの原子 ID
  real*4,intent(in),dimension(MAX_ATOM)::bondorder        ! bond 結合度
  real*4,intent(out),dimension(MAX_LIGAND,MAX_LIGAND)::rflag  ! 環情報
```

戻り値：

　なし

機能：

原子の bond 情報から環を検出し、環情報を設定する。

1.17. Is_ConnectedGraph

呼び出し形式：

```
subroutine Is_ConnectedGraph(numatom,numbond,nbond,one_graph)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
integer*4,intent(in)::numbond                          ! bond 数
integer*4,intent(in),dimension(MAX_ATOM,2)::nbond      ! bond ペアの原子 ID
logical,intent(out)::one_graph                         ! 連結判定結果
```

戻り値：

なし

機能：

当該分子がグラフとして連結かどうかを調べる。

1.18. Is_BridgeOfGraph

呼び出し形式：

```
subroutine Is_BridgeOfGraph(numatom,numbond,nbond,nc1,nc2,one_graph)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
integer*4,intent(in)::numbond                          ! bond 数
integer*4,intent(in),dimension(MAX_ATOM,2)::nbond      ! bond ペアの原子 ID
integer*4,intent(in)::nc1                              ! 切断原子 1
integer*4,intent(in)::nc2                              ! 切断原子 2
logical,intent(out)::one_graph                         ! 切断後のグラフ連結性
```

戻り値：

なし

機能：

入力した原子間の bond を切断した場合、当該分子がグラフとして連結かどうかを調べる。

1.19. Make_ConformerRecursive

呼び出し形式：

```
 recursive subroutine Make_ConformerRecursive(rotate, pointNum, phaseNum,        &
                                              location, atomNum, cord, rotNum,   &
                                              conformer, generateNum,            &
                                              atomID1, atomID2, atomID3,         &
                                              R, A, D, iatom, ato,               &
                                              rflag, conf, topAtom, lastAtom)
```

引数：

```
   integer*2,intent(inout),dimension(MAX_LIGAND)::rotate ! torsion の回転パターン
   integer*4,dimension(MAX_ATOM)::pointNum               ! 回転対象原子
   real*4,intent(in)::phaseNum                           ! 回転角数
   integer,intent(in)::location                          ! 回転対象位置
   integer,intent(in)::atomNum                           ! リガンドの原子数
   integer,intent(in)::rotNum                            ! 回転対象の個数
   real*4,intent(inout),dimension(MAX_ATOM,3)::cord      ! 原子座標
   real*4,intent(out),dimension(MAX_LIGAND,3,MAX_CONF)::conformer
                                                         ! コンフォーマ
   integer,intent(inout)::generateNum                    ! 生成コンフォーマ数
   integer*4,dimension(999)::atomID1,atomID2,atomID3     ! 1-2, 1-3, 1-4 の原子 ID
   real*4,intent(in),dimension(999)::R, A, D             ! bond 距離 angle 角 torsion 角
   integer*4,dimension(MAX_ATOM,5)::iatom                ! torsion 構成原子 ID
   character*1,intent(in),dimension(MAX_ATOM)::ato       ! 原子名
   real*4,intent(in),dimension(MAX_LIGAND,MAX_LIGAND)::rflag
                                                         ! 環内フラグ
   type(Conf_t),intent(in)::conf                         ! conf 情報
   integer,intent(in)::topAtom                           ! 座標作成原子先頭
   integer,intent(in)::lastAtom                          ! 座標作成原子最後
```

戻り値：

なし

機能：

回転対象の原子から次の回転対象原子までの区間の原子座標を生成し、次の区間を回転させて当該処理を呼び出す。

生成した原子座標が既に生成済みの原子と vdW 接触する場合には処理を中断し、次のパターンを試行する。

当該区間が最後の回転区間の場合には、当該原子座標をコンフォーマとして登録する。

1.20. Make_NewCordSingle

呼び出し形式：

```
 subroutine Make_NewCordSingle(i, atomID1, atomID2, atomID3, R, A, D,          &
                               co, rflag, isContact, damping, ato)
```

引数：

```
  integer*4,intent(in)::i                          ! 当該原子
  integer*4,intent(in),dimension(999)::atomID1     ! Z-matrix の 1-2 原子 ID
  integer*4,intent(in),dimension(999)::atomID2     ! Z-matrix の 1-3 原子 ID
  integer*4,intent(in),dimension(999)::atomID3     ! Z-matrix の 1-4 原子 ID
  real*4,intent(in),dimension(999)::R              ! Z-matrix の 1-2 距離
  real*4,intent(in),dimension(999)::A              ! Z-matrix の 1-2-3 角
  real*4,intent(in),dimension(999)::D              ! Z-matrix の 1-2-3-4 角
  real*4,intent(inout),dimension(MAX_ATOM,3)::co   ! リガンド原子座標
```

戻り値：

なし

機能：

当該原子の座標を Z-matrix と作成済みの原子座標から生成する。

1.21. Initialize_PreAllAtomData

呼び出し形式：

```
  subroutine Initialize_PreAllAtomData(numatom, numbond)
```

引数：

```
  integer,intent(in)::numatom                      ! 原子数
  integer,intent(in)::numbond                      ! bond 結合数
```

戻り値：

なし。

機能：

以下のデータ領域の動的確保および初期化を行う；

```
(all atom 時の Z-matrix 情報)
  integer*4,dimension(:,:)::nframe_all             ! Z-matrix 作成用原子 ID リスト
  integer*4,dimension(:)::R_all, A_all, D_all      ! Z-matrix
(all atom 時の原子情報)
  integer*4::numatom_all                           ! 原子数
  character*2,dimension(:)::atomName_all           ! 原子名
  character*1,dimension(:)::ato_all                ! 原子名（１文字目）
  integer*4,dimension(:)::nat_all                  ! 原子タイプ
  real*4,dimension(:)::hank_all                    ! 原子半径
  real*4,dimension(:)::charge_all                  ! 電荷
  integer*4,dimension(:)::mat_all                  ! 元素番号
(all atom 時の bond 情報)
  integer*4::numbond_all                           ! bond 結合数
  real*4,dimension(:)::bondorder_all               ! bond 結合次数
  integer*4,dimension(:,:)::nbond_all              ! bond 結合原子 ID リスト
(その他)
  logical,dimension(:)::isDelete_all               ! 水素原子削除フラグ
  integer*4,dimension(:)::atomID_org               ! 原子 ID
```

1.22. Create_UnitedAtom

呼び出し形式：

```
subroutine Create_UnitedAtom(numatom,atomName,ato,co,nat,                    &
                             hank,charge,nbond,bondorder,numbond,mat,        &
                             sybylType, substID, substName, isReadCharge,    &
                             atomStatus, bondStatus)
```

引数：

```
integer*4,intent(inout)::numatom                                  ! 原子数
character*2,intent(inout),dimension(MAX_ATOM)::atomName           ! 原子名
character*1,intent(inout),dimension(MAX_ATOM)::ato                ! 原子名（１文字目）
real*4,intent(inout),dimension(MAX_ATOM,3)::co                    ! 原子座標
integer*4,intent(inout),dimension(MAX_ATOM)::nat                  ! 原子タイプ
real*4,intent(inout),dimension(MAX_ATOM)::hank                    ! 原子半径
real*4,intent(inout),dimension(MAX_ATOM)::charge                  ! 各原子の電荷
integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond              ! bond 結合原子 ID リスト
real*4,intent(inout),dimension(MAX_ATOM)::bondorder               ! bond 結合次数
integer*4,intent(inout)::numbond                                  ! bond 結合数
integer*4,intent(inout),dimension(MAX_ATOM)::mat                  ! 元素番号
character*5,intent(inout),dimension(MAX_ATOM)::sybylType          ! SYBYL 原子タイプ
integer*4,intent(inout),dimension(MAX_ATOM)::substID              ! 残基番号
character*10,intent(inout),dimension(MAX_ATOM)::substName         ! 残基名
logical*4,intent(inout),dimension(MAX_ATOM)::isReadCharge         ! 電荷読み込みフラグ
integer*4,intent(inout),dimension(MAX_ATOM)::atomStatus           ! 原子情報
integer*4,intent(inout),dimension(MAX_ATOM)::bondStatus           ! 結合情報
```

戻り値：

なし。

内容：

all atom モデルから united atom モデルへの変換を行う。具体的な処理方法は以下の通り。

（１）サブルーチン Make_Frame を用いて all atom 時の Z-matrix 作成用の ID リストを作成し、保存する。

（２）nframe_all を基に、サブルーチン Create_ZMatrixParameter を用いて all atom 時の Z-matrix を作成し、保存する。

（３）各原子をループして -CH, -CH2, -CH3 を検索し、当該水素原子の水素原子削除フラを .true. に設定する。

（４）削除対象水素原子の原子情報を削除する。

（５）削除対象水素原子の bond 情報を削除する。

1.23. Restore_AllAtom

呼び出し形式：

```
subroutine Restore_AllAtom(numatom,atomName,ato,co,nat,                    &
                           hank,charge,nbond,bondorder,numbond,mat,        &
                           sybylType, substID, substName, isReadCharge,    &
                           atomStatus, bondStatus)
```

引数：

```
integer*4,intent(inout)::numatom                                  ! 原子数
character*2,intent(inout),dimension(MAX_ATOM)::atomName           ! 原子名
character*1,intent(inout),dimension(MAX_ATOM)::ato                ! 原子名（１文字目）
real*4,intent(inout),dimension(MAX_ATOM,3)::co                    ! 原子座標
integer*4,intent(inout),dimension(MAX_ATOM)::nat                  ! 原子タイプ
real*4,intent(inout),dimension(MAX_ATOM)::hank                    ! 原子半径
real*4,intent(inout),dimension(MAX_ATOM)::charge                  ! 各原子の電荷
integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond              ! bond 結合原子 ID リスト
real*4,intent(inout),dimension(MAX_ATOM)::bondorder               ! bond 結合次数
integer*4,intent(inout)::numbond                                  ! bond 結合数
integer*4,intent(inout),dimension(MAX_ATOM)::mat                  ! 元素番号
character*5,intent(inout),dimension(MAX_ATOM)::sybylType          ! SYBYL 原子タイプ
integer*4,intent(inout),dimension(MAX_ATOM)::substID              ! 残基番号
character*10,intent(inout),dimension(MAX_ATOM)::substName         ! 残基名
logical*4,intent(inout),dimension(MAX_ATOM)::isReadCharge         ! 電荷読み込みフラグ
integer*4,intent(inout),dimension(MAX_ATOM)::atomStatus           ! 原子情報
integer*4,intent(inout),dimension(MAX_ATOM)::bondStatus           ! 結合情報
```

戻り値：

なし。

内容：

united atom モデルから all atom モデルの復元を行う。具体的な処理方法は以下の通り。

（１）サブルーチン Restore_Coordinate を呼び出し、united atom モデル作成時に削除した水素原子の座標を復元する。

（２）各上位スコア座標に対して、サブルーチン Restore_Coordinate を呼び出し、united atom モデル作成時に削除した水素原子の座標を復元する。

（３）参照用座標に対して、サブルーチン Restore_Coordinate を呼び出し、united atom モデル作成時に削除した水素原子の座標を復元する。

（４）all atom 時の原子情報（**1.21.** 中の「all atom 時の原子情報」）を復元する。

（５）all atom 時の bond 情報（**1.21.** 中の「all atom 時の bond 情報」）を復元する。

1.24. Restore_Coordinate

呼び出し形式：

```
subroutine Restore_Coordinate(co)
```

引数：

```
real*4,intent(inout),dimension(MAX_ATOM,3)::co          ! 原子座標
```

戻り値：

なし。

内容：

united atom モデル作成時に削除した水素原子座標の復元を行う。具体的な処理方法は以下の通り。

（１）各原子をループし、水素原子削除フラグを参照する。

（２）i 番目の原子の水素原子削除フラグが.true.の場合、i 番目以降の座標データを１つ後ろにずらす。
（３）all atom 時の Z-matrix を基に、サブルーチン GENXYZ を用いて i 番目の座標データを復元する。

2.Convert_Data

ファイル名：Convert_Data.f90

cosgene 用トポロジーデータ生成のメソッドを定義する。

2.1.Generate_Tpl

呼び出し形式：

```
subroutine Generate_Tpl(atomName, cord, charge, bondPair, atomNum, bondNum,&
                        type,                                               &
                        atomInfo, molInfo, chainInfo, residInfo,            &
                        bondInfo, angleInfo, torsionInfo, improperInfo,     &
                        nonBondedInfo, functionInfo, acceptDonarInfo,       &
                        donarNum, donarID, acceptNum, acceptID)
```

引数：

```
character*2,intent(inout),dimension(MAX_ATOM)::atomName     ! 原子タイプ
real*4,intent(in),dimension(MAX_ATOM,3)::cord               ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::charge               ! 原子電荷
integer,intent(inout),dimension(MAX_ATOM,2)::bondPair       ! bond atom list
integer,intent(in)::bondNum                                 ! bond 数
integer,intent(in)::atomNum                                 ! 原子数
integer,intent(in),dimension(MAX_ATOM)::type                ! ASA の型
integer,intent(in),dimension(MAX_ATOM)::donarID             ! donar 数
integer,intent(in)::donarNum                                ! donar の原子 ID
integer,intent(in),dimension(MAX_ATOM)::acceptID            ! acceptor 数
integer,intent(in)::acceptNum                               ! acceptor の原子 IDID
type(MolInfo_t),intent(out)::molInfo                        ! molecule 情報(cosgene 用)
type(ChainInfo_t),intent(out)::chainInfo                    ! chain 情報(cosgene 用)
type(ResidInfo_t),intent(out)::residInfo                    ! residue 情報(cosgene 用)
type(AtomInfo_t),intent(out)::atomInfo                      ! atom 情報(cosgene 用)
type(BondInfo_t),intent(out)::bondInfo                      ! bond 情報(cosgene 用)
type(AngleInfo_t),intent(out)::angleInfo                    ! angle 情報(cosgene 用)
type(TorsionInfo_t),intent(out)::torsionInfo                ! torsion 情報(cosgene 用)
type(TorsionInfo_t),intent(out)::improperInfo               ! improper 情報(cosgene 用)
type(NonBondedInfo_t),intent(out)::nonBondedInfo            ! non-bond 情報(cosgene 用)
type(Function_t),intent(out)::functionInfo                  ! function 情報(cosgene 用)
type(AcceptDonar_t),intent(out)::acceptDonarInfo            ! acceptor donar 情報(cosgene 用)
```

戻り値：

なし

機能：

原子情報、bond 情報、原子座標から cosgene 用のトポロジーの内部表現形式を作成する

2.2.Create_AtomList

呼び出し形式：

```
  subroutine Create_AtomList(bondPair, bondNum, bondCount, atomList)
```

引数：

```
  integer,intent(inout),dimension(MAX_ATOM,2)::bondPair ! bond ペア原子 ID
  integer,intent(in)::bondNum                           ! bond 数
  integer,intent(out),dimension(MAX_ATOM,9)::atomList   ! 当該原子の BOND 相手原子 ID
  integer,intent(out),dimension(MAX_ATOM)::bondCount    ! 当該原子の BOND 数
```

戻り値：

　なし

機能：

当該原子に対する BOND のリストを作成する。

2.3.Decide_AtomInformation

呼び出し形式：

```
  subroutine Decide_AtomInformation(bondCount, atomNum, atomList,            &
                                    numSP, atomName)
```

引数：

```
  integer,intent(in),dimension(MAX_ATOM)::bondCount     ! 当該原子の BOND 数
  integer,intent(in)::atomNum                           ! 原子数
  integer,intent(in),dimension(MAX_ATOM,9)::atomList    ! 当該原子の BOND 相手原子 ID
  character*2,intent(inout),dimension(MAX_ATOM)::atomName  ! 原子タイプ
  integer,intent(out),dimension(MAX_ATOM)::numSP        ! 混成軌道数
```

戻り値：

　なし

機能：

当該原子タイプおよび BOND 数から原子タイプを更新し、混成軌道数を設定する。

2.4.Set_ObjectInformation

呼び出し形式：

```
  subroutine Set_ObjectInformation(atomName, atomNum, charge, cord,          &
                                   molInfo, chainInfo, residInfo, atomInfo)
```

引数：

```
  character*2,intent(in),dimension(MAX_ATOM)::atomName  ! 原子名
  integer,intent(in)::atomNum                           ! 原子数
  real*4,intent(in),dimension(MAX_ATOM)::charge         ! 原子電荷
  real*4,intent(in),dimension(MAX_ATOM,3)::cord         ! 原子座標
  type(MolInfo_t),intent(out)::molInfo                  ! molecule 情報(cosgene 用)
  type(ChainInfo_t),intent(out)::chainInfo              ! chain 情報(cosgene 用)
  type(ResidInfo_t),intent(out)::residInfo              ! residue 情報(cosgene 用)
  type(AtomInfo_t),intent(out)::atomInfo                ! atom 情報(cosgene 用)
```

戻り値：

　なし

機能：

cosgene のオブジェクト情報 (molecule, chain, residue, atom)を設定する。

2.5.Set_InteractInformation

呼び出し形式：

```
subroutine Set_InteractInformation(bondPair, atomList, bondNum, bondCount, &
                                   numSP, atomNum, cord,                   &
                                   atomInfo,                               &
                                   bondInfo, angleInfo, torsionInfo,       &
                                   improperInfo, nonBondedInfo)
```

引数：

```
integer,intent(in),dimension(MAX_ATOM,2)::bondPair     ! bond ペアの原子 ID
integer,intent(in),dimension(MAX_ATOM,9)::atomList     ! 当該原子の BOND 相手原子 ID
integer,intent(in),dimension(MAX_ATOM)::bondCount      ! 当該原子の BOND 数
integer,intent(in)::bondNum                            ! bond 数
integer,intent(in),dimension(MAX_ATOM)::numSP          ! 混成軌道数
integer,intent(in)::atomNum                            ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::cord          ! 原子座標
type(AtomInfo_t),intent(out)::atomInfo                 ! atom 情報(cosgene 用)
type(BondInfo_t),intent(out)::bondInfo                 ! bond 情報(cosgene 用)
type(AngleInfo_t),intent(out)::angleInfo               ! angle 情報(cosgene 用)
type(TorsionInfo_t),intent(out)::torsionInfo           ! torsion 情報(cosgene 用)
type(TorsionInfo_t),intent(out)::improperInfo          ! improper 情報(cosgene 用)
type(NonBondedInfo_t),intent(out)::nonBondedInfo       ! non-bond 情報(cosgene 用)
```

戻り値：

なし

機能：

BOND 情報、混成軌道数から原子の bond, angle, torsion, improper 情報を生成する。
また、amber 形式の 1-5 相互作用情報を作成する。

2.6.Set_AtomInteraction

呼び出し形式：

```
subroutine Set_AtomInteraction(bondInfo, angleInfo, torsionInfo, atomInfo)
```

引数：

```
type(AtomInfo_t),intent(inout)::atomInfo               ! atom 情報(cosgene 用)
type(BondInfo_t),intent(in)::bondInfo                  ! bond 情報(cosgene 用)
type(AngleInfo_t),intent(in)::angleInfo                ! angle 情報(cosgene 用)
type(TorsionInfo_t),intent(in)::torsionInfo            ! torsion 情報(cosgene 用)
```

戻り値：

なし

機能：

当該原子の 1-2, 1-3, 1-4 原子の情報を作成する。

2.7.Get_Distance

呼び出し形式：
```
function Get_Distance(cord1, cord2)
```
引数：
```
real*4,intent(in),dimension(3)::cord1
real*4,intent(in),dimension(3)::cord2
```
戻り値：
```
real*8::dist
```
機能：
原子間の距離を求める。

2.8.Get_Angle

呼び出し形式：
```
function Get_Angle(cord1, cord2, cord3)
```
引数：
```
real*4,intent(in),dimension(3)::cord1
real*4,intent(in),dimension(3)::cord2
real*4,intent(in),dimension(3)::cord3
```
戻り値：
```
real*8::cosVal, angle
```
機能：
３原子が生成する angle を求める。

3.Dock_Method

ファイル名：Dock_Method.f90

グローバルサーチのメソッドを定義する。

3.1.Check_EdgeCount

呼び出し形式：

```
subroutine Check_EdgeCount(numatom,nat,ddtable,dist_min,                    &
                           dist_max, number)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
integer*4,intent(in),dimension(MAX_ATOM)::nat          ! 原子タイプ
real*4,intent(in),dimension(200,200)::ddtable          ! 原子間距離テーブル
real*4,intent(in)::dist_min                            ! 距離下限
real*4,intent(in)::dist_max                            ! 距離上限
integer*4,intent(out)::number                          ! 距離内辺の個数
```

戻り値：

なし

機能：

２原子の距離が入力した範囲内にある組み合わせの個数をカウントする。

3.2.Dock_Main

呼び出し形式：

```
subroutine Dock_Main(dock,numatom,co,xyz_psas,nsas2atop,                    &
                     nat,natp,charge,num_don_l,list_don,                    &
                     num_acc_l,list_acc,use_atom,ngh,nghl,                  &
                     number)
```

引数：

```
type(Dock_t),intent(in)::dock                          ! dock 情報
integer*4,intent(in)::numatom                          ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 原子座標
real*4,intent(in),dimension(MAX_SURFACE,3)::xyz_psas   ! ASA の座標
integer*4,intent(in),dimension(MAX_SURFACE)::nsas2atop ! ASA の属する原子の ID
integer*4,intent(in),dimension(MAX_ATOM)::nat          ! リガンド原子タイプ
integer*4,intent(in),dimension(MAX_ATOM)::natp         ! 蛋白原子タイプ
real*4,intent(in),dimension(MAX_ATOM)::charge          ! リガンド原子電荷
integer*4,intent(in)::num_don_l                        ! ドナー数
integer*4,intent(in),dimension(MAX_ATOM)::list_don     ! ドナーの原子 ID
integer*4,intent(in)::num_acc_l                        ! アクセプタ数
integer*4,intent(in),dimension(MAX_ATOM)::list_acc     ! アクセプタの原子 ID
logical,intent(in),dimension(MAX_ATOM)::use_atom       ! リガンド側原子使用フラグ
integer*4,intent(in),dimension(18,18,18,3,1500)::ngh   ! ポケット接合面座標テーブル
integer*4,intent(in),dimension(18,18,18)::nghl         ! ポケット接合面座標数
integer*4,intent(out)::number                          ! ドッキング試行回数
```

戻り値：

なし

機能：

リガンドと蛋白ポケットとのグローバルサーチを行う。

3.3.Evaluate_Pot

呼び出し形式：

```
  subroutine Evaluate_Pot(numatom,co,nat,asaPot, elePot, hydPot,              &
                          charge,num_don_l,list_don,num_acc_l,list_acc)
```

引数：

```
  integer*4,intent(in)::numatom                          ! リガンド原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co            ! リガンド原子座標
  integer*4,intent(in),dimension(MAX_ATOM)::nat          ! リガンド原子タイプ
  real*4,intent(out)::asaPot                             ! A,S.A.ポテンシャル
  real*4,intent(out)::elePot                             ! 静電ポテンシャル
  real*4,intent(out)::hydPot                             ! 水素結合ポテンシャル
  real*4,intent(in),dimension(MAX_ATOM)::charge          ! 電荷
  integer*4,intent(in)::num_don_l                        ! ドナー数
  integer*4,intent(in),dimension(MAX_ATOM)::list_don     ! ドナー原子 ID
  integer*4,intent(in)::num_acc_l                        ! アクセプタ数
  integer*4,intent(in),dimension(MAX_ATOM)::list_acc     ! アクセプタ原子 ID
```

戻り値：

　なし

機能：

　当該リガンドが当該座標に存在する場合のポテンシャルを計算する。

3.4.Evaluate_PotH

呼び出し形式：

```
subroutine Evaluate_PotH(co,cop,cop_pre,hydPot,num_don_l,list_don,          &
                         list_do2,list_do3,list_do4,                        &
                         num_acc_l,list_acc,n_type_lhd,n_type_lha,          &
                         no_id,n_l_stem,n_n_br,                             &
                         n_l_br,n_s_a,n_s_d,n_l_a,n_l_d1,n_l_d2,h_rotate,   &
                         n_type_a,n_type_d,isRotSideChain,isRotLigandOH)
```

引数：

```
real*4,intent(in),dimension(MAX_ATOM,3)::co          ! リガンド原子座標
real*4,intent(inout),dimension(MAX_ATOM,3)::cop      ! 蛋白原子座標
real*4,intent(in),dimension(MAX_ATOM,3)::cop_pre     ! 蛋白原子座標
real*4,intent(out)::hydPot                           ! ポテンシャルエネルギー
integer*4,intent(in)::num_don_l                      ! リガンドドナー数
integer*4,intent(in),dimension(MAX_ATOM)::list_don   ! リガンドドナーリスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do2   ! torsion リスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do3   ! torsion リスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do4   ! torsion リスト
integer*4,intent(in)::num_acc_l                      ! アクセプタ数
integer*4,intent(in),dimension(MAX_ATOM)::list_acc   ! アクセプタリスト
integer*4,intent(in),dimension(MAX_ATOM)::n_type_lhd ! リガンドドナー種別
integer*4,intent(in),dimension(MAX_ATOM)::n_type_lha ! リガンドアクセプタ種別
integer*4,intent(in)::no_id                          ! 蛋白質側鎖数
integer*4,intent(in),dimension(MAX_SC,MAX_SCATM)::n_l_stem ! 土台部分のリスト
integer*4,intent(in),dimension(MAX_SC)::n_n_br              ! 回転部分の原子数
integer*4,intent(in),dimension(MAX_SC,MAX_SCATM)::n_l_br    ! 回転部分のリスト
integer*4,intent(in),dimension(MAX_SC)::n_s_a               ! 回転部分のアクセプタ数
integer*4,intent(in),dimension(MAX_SC)::n_s_d               ! 回転部分のドナー数
integer*4,intent(in),dimension(MAX_SC,MAX_SCATM)::n_l_a  ! 回転部分のアクセプタリスト
integer*4,intent(in),dimension(MAX_SC,MAX_SCATM)::n_l_d1 ! 回転部分のドナーリスト
integer*4,intent(in),dimension(MAX_SC,MAX_SCATM)::n_l_d2 ! 回転部分のドナーリスト
logical*4,intent(in),dimension(MAX_SC)::h_rotate         ! 側鎖の回転フラグ
integer*4,intent(in),dimension(MAX_SC)::n_type_a         ! アクセプタ種別
integer*4,intent(in),dimension(MAX_SC)::n_type_d         ! ドナー種別
logical,intent(in)::isRotSideChain                       ! 蛋白質側鎖の回転フラグ
logical,intent(in)::isRotLigandOH                        ! リガンド-OH 基の回転フラグ
```

戻り値：

なし

機能：

蛋白とリガンドの水素結合エネルギー（水素結合の異方性を考慮した項)を計算する。

3.5.Optimize_Coordinate

呼び出し形式：

```
subroutine Optimize_Coordinate(numatom,co,cop,cop_pre,nat,charge,num_don_l,   &
                               list_don,list_do2,list_do3,list_do4,           &
                               num_acc_l,list_acc,n_type_lhd,n_type_lha,      &
                               no_id,n_l_stem,n_n_br,n_l_br,n_s_a,n_s_d,      &
                               n_l_a,n_l_d1,n_l_d2,h_rotate,n_type_a,         &
                               n_type_d,isCalcAnHydPot,isRotSideChain,        &
                               isRotLigandOH,moveNum)
```

引数：

```
integer*4,intent(in)::numatom                          ! リガンド原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! リガンド原子座標
real*4,intent(inout),dimension(MAX_ATOM,3)::cop        ! 蛋白原子座標
real*4,intent(in),dimension(MAX_ATOM,3)::cop_pre       ! 蛋白原子座標
integer*4,intent(in),dimension(MAX_ATOM)::nat          ! リガンド原子タイプ
real*4,intent(in),dimension(MAX_ATOM)::charge          ! 電荷
integer*4,intent(in)::num_don_l                        ! リガンドドナー数
integer*4,intent(in),dimension(MAX_ATOM)::list_don     ! リガンドドナーリスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do2     ! torsion リスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do3     ! torsion リスト
integer*4,intent(in),dimension(MAX_ATOM)::list_do4     ! torsion リスト
integer*4,intent(in)::num_acc_l                        ! リガンドアクセプタ数
integer*4,intent(in),dimension(MAX_ATOM)::list_acc     ! リガンドアクセプタリスト
integer*4,intent(in),dimension(MAX_ATOM)::n_type_lhd   ! リガンドドナー種別
integer*4,intent(in),dimension(MAX_ATOM)::n_type_lha   ! リガンドアクセプタ種別
integer*4,intent(out)::no_id                           ! 蛋白質側鎖数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_stem ! 土台部分のリスト
integer*4,intent(out),dimension(MAX_SC)::n_n_br              ! 回転部分の原子数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_br    ! 回転部分のリスト
integer*4,intent(out),dimension(MAX_SC)::n_s_a               ! 回転部分のアクセプタ数
integer*4,intent(out),dimension(MAX_SC)::n_s_d               ! 回転部分のドナー数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_a ! 回転部分のアクセプタリスト
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_d1    ! 回転部分のドナーリスト
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_d2    ! 回転部分のドナーリスト。
logical*4,intent(out),dimension(MAX_SC)::h_rotate            ! 蛋白質側鎖の回転フラグ
integer*4,intent(out),dimension(MAX_SC)::n_type_a            ! アクセプタ種別
integer*4,intent(out),dimension(MAX_SC)::n_type_d            ! ドナー種別
```

戻り値：

なし

機能：

ドッキング後の座標の最適化を行う。

4.Grid_Method

ファイル名：Grid_Method.f90

Gridポテンシャル生成およびGridポテンシャルに対するポテンシャル計算のメソッドを定義する。

4.1.Set_GridScale

呼び出し形式：

```
  subroutine Set_GridScale(npoint,point, mergin)
```

引数：

```
  integer*4,intent(in)::npoint                        ! ポケット点数
  real*4,intent(in),dimension(MAX_ATOM,3)::point      ! ポケット点座標
  real*4,intent(in)::mergin                           ! マージン
```

戻り値：

なし

機能：

グリッドのサイズを決定する。(ポケット点の直交座標での最大/最小 ± マージン)

4.2.Generate_Grid

呼び出し形式：

```
  subroutine Generate_Grid(numatom,co,hank,charge,natp,                          &
                           numdonar_p,numaccep_p,don_cop,acc_cop,list_hot)
```

引数：

```
  integer*4,intent(in)::numatom                         ! 蛋白原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co           ! 蛋白原子座標
  real*4,intent(in),dimension(MAX_ATOM)::hank           ! 原子半径
  real*4,intent(in),dimension(MAX_ATOM)::charge         ! 原子電荷
  integer*4,intent(in),dimension(MAX_ATOM)::natp        ! 原子タイプ
  integer*4,intent(in)::numdonar_p                      ! ドナー数
  integer*4,intent(in)::numaccep_p                      ! アクセプタ数
  real*4,intent(in),dimension(MAX_ATOM,3)::don_cop      ! ドナー座標
  real*4,intent(in),dimension(MAX_ATOM,3)::acc_cop      ! アクセプタ座標
  integer*4,intent(in),dimension(MAX_ATOM)::list_hot ! host point フラグ
```

戻り値：

なし

機能：

GridPotential 生成処理の主制御を行う。

4.3.Smooth_Grid

呼び出し形式：
```
  subroutine Smooth_Grid(ntime,weight)
```
引数：
```
  integer*4,intent(in)::ntime                              ! スムージング回数
  real*4,intent(in),dimension(GRID_SIZE,GRID_SIZE,GRID_SIZE)::weight
                                                           ! グリッドの格子点の重み
```
戻り値：
　なし
機能：
GridPotential のスムージングを行う。

4.4.Generate_AsaPot

呼び出し形式：
```
  subroutine Generate_AsaPot(numatom,co,hank,n_sas,xyz_sas,natp,list_hot)
```
引数：
```
  integer*4,intent(in)::numatom                            ! 蛋白原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co              ! 蛋白原子座標
  real*4,intent(in),dimension(MAX_ATOM)::hank              ! 原子半径
  integer*4,intent(in)::n_sas                              ! A.S.A 数
  real*4,intent(in),dimension(MAX_SURFACE,3)::xyz_sas ! A.S.A 座標
  integer*4,intent(in),dimension(MAX_ATOM)::natp       ! 原子タイプ
  integer*4,intent(in),dimension(MAX_ATOM)::list_hot  ! hotspot フラグ
```
戻り値：
　なし
機能：
Grid の格子上に蛋白原子に対する A.S.A ポテンシャルを生成する。

4.5.Generate_ElePot

呼び出し形式：
```
  subroutine Generate_ElePot(numatom,co,hank,n_sas,xyz_sas,charge)
```
引数：
```
  integer*4,intent(in)::numatom                            ! 蛋白原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co              ! 蛋白原子座標
  real*4,intent(in),dimension(MAX_ATOM)::hank              ! 原子半径
  integer*4,intent(in)::n_sas                              ! A.S.A 数
  real*4,intent(in),dimension(MAX_SURFACE,3)::xyz_sas ! A.S.A 座標
  real*4,intent(in),dimension(MAX_ATOM)::charge        ! 原子電荷
```
戻り値：
　なし
機能：
Grid の格子上に蛋白原子に対する静電ポテンシャルを生成する。

4.6.Generate_HydPot

呼び出し形式：
```
  subroutine Generate_HydPot(numdonar_p,don_cop,numaccep_p,acc_cop)
```
引数：
```
  integer*4,intent(in)::numdonar_p                    ! ドナー数
  real*4,intent(in),dimension(MAX_ATOM,3)::don_cop    ! ドナー座標
  integer*4,intent(in)::numaccep_p                    ! アクセプタ数
  real*4,intent(in),dimension(MAX_ATOM,3)::acc_cop    ! アクセプタ座標
```
戻り値：
　なし
機能：
　Grid の格子上にドナー/アクセプタ対する水素結合ポテンシャルを生成する。

4.7.Calc_AsaPot

呼び出し形式：
```
  subroutine Calc_AsaPot(x,y,z,asaPot,elePot,grad,ity,qcharge,                &
                         nswt,nerror)
```
引数：
```
  real*4,intent(in)::x, y, z                          ! 当該原子の座標
  real*4,intent(out)::asaPot                          ! A.S.A ポテンシャル
  real*4,intent(out)::elePot                          ! 静電ポテンシャル
  real*4,intent(out),dimension(3)::grad               ! 当該原子に対するエネルギー勾配
  integer*4,intent(in)::ity                           ! 当該原子の A.S.A の型
  real*4,intent(in)::qcharge                          ! 当該原子の電荷
  integer*4,intent(in)::nswt                          ! エネルギー勾配計算フラグ
  integer*4,intent(out)::nerror                       ! 範囲外エラーフラグ
```
戻り値：
　なし
機能：
　当該原子に対する静電、A.S.A ポテンシャルを計算する。
　nwst が 1 の場合、エネルギー勾配を計算する。

4.8.Calc_HydPot

呼び出し形式：
```
  subroutine Calc_HydPot(x,y,z,epot,grad,mty,nswt,nerror)
```
引数：
```
  real*4,intent(in)::x, y, z                          ! 当該原子座標
  real*4,intent(out)::epot                            ! 水素結合ポテンシャル
  real*4,intent(inout),dimension(3)::grad             ! 当該原子に対するエネルギー勾配
  integer*4,intent(in)::mty                           ! 当該原子の水素結合タイプ
  integer*4,intent(in)::nswt                          ! エネルギー勾配計算フラグ
  integer*4,intent(out)::nerror                       ! 範囲外エラーフラグ
```
戻り値：
　なし
機能：
　当該原子に対する水素結合ポテンシャルを計算する。
　nwst が 1 の場合、エネルギー勾配を計算する。

4.9.Set_MeshWeight

呼び出し形式：

```
  subroutine Set_MeshWeight(yuragi_list,numatop,cop,weight)
```

引数：

```
  real*4,intent(in),dimension(MAX_ATOM)::yuragi_list ! 各原子の重み係数
  integer*4,intent(in)::numatop                      ! 蛋白原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::cop       ! 蛋白原子座標
  real*4,intent(inout),dimension(GRID_SIZE,GRID_SIZE,GRID_SIZE)::weight
                                                     ! Grid 格子点の重み
```

戻り値：

なし

機能：

当該蛋白原子の重みを Grid 上の格子点に反映させる。

4.10.Check_GridLocation

呼び出し形式：

```
  subroutine Check_GridLocation(cord, intCord, atomNum, errorCode)
```

引数：

```
  real,intent(in),dimension(MAX_ATOM,3)::cord          ! リガンド原子座標
  integer,intent(out),dimension(MAX_ATOM,3)::intCord   ! 各原子が対応する格子点
  integer,intent(in)::atomNum                          ! リガンド原子数
  integer,intent(out)::errorCode                       ! 範囲外フラグ
```

戻り値：

なし

機能：

当該リガンド分子の全原子が Grid 内に存在するかチェックする。
また、各原子が対応する Grid の格子点を計算する。

4.11.Calc_AllAsaPot

呼び出し形式：

```
  subroutine Calc_AllAsaPot(cord,intCord,atomNum,asaPot,elePot,atomType,      &
                            charge)
```

引数：

```
  real*4,intent(in),dimension(MAX_ATOM,3)::cord        ! リガンド原子座標
  integer,intent(in),dimension(MAX_ATOM,3)::intCord    ! 原子の格子点
  integer,intent(in)::atomNum                          ! 原子数
  real*4,intent(out)::asaPot                           ! A.S.A ポテンシャル
  real*4,intent(out)::elePot                           ! 静電ポテンシャル
  integer,intent(in),dimension(MAX_ATOM)::atomType     ! リガンドの原子タイプ
  real*4,intent(in),dimension(MAX_ATOM)::charge        ! リガンドの電荷
```

戻り値：

なし

機能：

当該リガンド内の全原子に対する静電、A.S.A ポテンシャルを計算する。

4.12.Calc_AllHydPot

呼び出し形式：

```
  subroutine Calc_AllHydPot(cord,intCord,atomList,atomNum,hydPot,atomType)
```

引数：

```
  real*4,intent(in),dimension(MAX_ATOM,3)::cord        ! リガンド原子座標
  integer,intent(in),dimension(MAX_ATOM,3)::intCord    ! 原子の格子点
  integer*4,intent(in),dimension(MAX_ATOM)::atomList ! donar / acceptor リスト
  integer*4,intent(in)::atomNum                        ! donar / acceptor 数
  real*4,intent(out)::hydPot                           ! 水素結合ポテンシャル
  integer,intent(in)::atomType                         ! donar / acceptor の型
```

戻り値：

なし

機能：

当該リガンド内の全ドナー･アクセプタに対する水素結合ポテンシャルを計算する。

4.13.Grid_Main

呼び出し形式：

```
  subroutine Grid_Main(grid, numatop, atp, cop, hankp, point, npoint,        &
                       xyz_psas, acc_cop, nsas2atop, n_sas, natp, chargp,    &
                       numaccep_p, listp_acc, ngh, nghl, numdonar_p,         &
                       ntable, list_hot, don_cop, yuragi_list)
```

引数：

```
  type(Grid_t),intent(in)::grid                    ! Grid 情報
  integer,intent(in)::numatop                      ! 蛋白原子数
  character*1,intent(in),dimension(MAX_ATOM)::atp    ! 蛋白原子タイプ
  real*4,intent(in),dimension(MAX_ATOM,3)::cop   ! 蛋白原子座標
  real*4,intent(out),dimension(MAX_ATOM)::hankp  ! 蛋白原子半径
  real*4,intent(in),dimension(MAX_ATOM,3)::point ! ポケット点座標
  integer,intent(inout)::npoint                    ! ポケット点数
  real*4,intent(in),dimension(MAX_ATOM)::chargp  ! 蛋白原子電荷
  real*4,intent(out),dimension(MAX_SURFACE,3)::xyz_psas
                                                     ! 蛋白 A.S.A 座標
  real*4,intent(in),dimension(MAX_ATOM,3)::acc_cop
                                                     ! 蛋白アクセプタ座標
  integer,intent(out),dimension(MAX_SURFACE)::nsas2atop ! A.S.A 対応原子 ID
  integer,intent(out)::n_sas                       ! A.S.A 数
  integer,intent(in),dimension(MAX_ATOM)::natp   ! 蛋白原子タイプ
  integer,intent(in)::numaccep_p                   ! 蛋白アクセプタ数
  integer,intent(in),dimension(MAX_ATOM)::listp_acc
                                                    ! 蛋白アクセプタ原子 ID
  integer,intent(out),dimension(18,18,18,3,1500)::ngh ! 蛋白接合面座標リスト
  integer,intent(out),dimension(18,18,18)::nghl  ! 蛋白接合面数
  integer,intent(in)::numdonar_p                   ! 蛋白ドナー数
  integer,intent(out),dimension(MAX_ATOM)::ntable    ! 蛋白ポケット点近接フラグ
  integer,intent(in),dimension(MAX_ATOM)::list_hot   ! hot-spot 原子フラグ
  real*4,intent(in),dimension(MAX_ATOM,3)::don_cop
                                                     ! 蛋白ドナー座標
  real*4,intent(in),dimension(MAX_ATOM)::yuragi_list ! 蛋白の重み
```

戻り値：

　なし

機能：

GridPotential 生成の主制御を行う。
GridPotential の入出力指定がある場合は、ファイルへの入出力を行う。

4.14.Calc_GridPotential

呼び出し形式：

```
  subroutine Calc_GridPotential(atomInfo, acceptDonarInfo, dyn)
```

引数：

```
  type(AtomInfo_t),intent(inout)::atomInfo           ! 原子情報
  type(AcceptDonar_t),intent(in)::acceptDonarInfo    ! donar / acceptor 情報(cosgene 用)
  type(Dynamics_t),intent(inout)::dyn                ! エネルギー情報
```

戻り値：

　なし

機能：

MINIMIZE 処理時に GridPotential を計算する。

5.PairwiseASA.Method

ファイル名：PairwiseASA_Method.f90

Pairwise 法の AccessibleSurfaceArea のメソッドを定義する。

5.1.Make_Asa

呼び出し形式：

```
subroutine Make_Asa(numatom,co,hank,n_sas,xyz_sas)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::hank            ! 原子半径
integer*4,intent(out)::n_sas                           ! A.S.A 数
real*4,intent(out),dimension(MAX_SURFACE,3)::xyz_sas   ! A.S.A 座標
```

戻り値：

なし

機能：

当該原子群が構成する A.S.A の点座標を生成する。

5.2.asacheck

呼び出し形式：

```
subroutine asacheck(numatom,co,hankei,asa_list)
```

引数：

```
integer*4,intent(in)::numatom                        ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co          ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::hankei        ! 原子半径
real*4,intent(out),dimension(MAX_ATOM)::asa_list     ! A.S.A 原子リスト
```

戻り値：

なし

機能：

各原子が保持する A.S.A 上の点の個数をカウントする

5.3.Calc_AsaScore

呼び出し形式：

```
subroutine Calc_AsaScore(numatom,numatoc,co,hankei,score)
```

引数：

```
integer*4,intent(in)::numatom                  ! リガンド+蛋白原子数
integer*4,intent(in)::numatoc                  ! リガンド原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co    ! リガンド+蛋白原子座標
real*4,intent(in),dimension(MAX_ATOM)::hankei  ! 原子半径
real*4,intent(out)::score                      ! スコア
```

戻り値：

なし

機能：

リガンドと蛋白の A.S.A の点の個数から、スコアを計算する。

6.Score_Method

ファイル名：Score_Method.f90

sievgeneのスコア登録に関するメソッドを定義する。

6.1.Regist_TotalScore

呼び出し形式：
```
subroutine Regist_TotalScore
```
引数：
なし
戻り値：
なし
機能：
当該リガンドでの上位ポテンシャルのスコアを、全体のスコアに登録する。

6.2.Regist_LocalScore

呼び出し形式：
```
subroutine Regist_LocalScore(asaPot,elePot,hydPot,vdwPot,cord,atomNum)
```
引数：
```
real*4,intent(in)::asaPot              ! accessible surface area ポテンシャル
real*4,intent(in)::elePot              ! 静電ポテンシャル
real*4,intent(in)::hydPot              ! 水素結合ポテンシャル
real*4,intent(in)::vdwPot              ! van der Waals ポテンシャル
real*4,intent(in),dimension(MAX_ATOM,3)::cord ! 当該conformer座標
integer,intent(in)::atomNum                   ! 当該confomer原子数
```
戻り値：
なし
機能：
当該リガンドのスコアをローカルスコアに登録する。

6.3.Regist_Score

呼び出し形式：

```
  subroutine Regist_Score(asaPot, elePot, hydPot, vdwPot, surface, cord,     &
                          atomNum,                                           &
                          atomicName,                                        &
                          name, currentRmsd,                                 &
                          score, energy, rankCord, ligandName,               &
                          atomName, refRmsd,                                 &
                          ligandAtomNum, scoreNum)
```

引数：

```
  real*4,intent(in)::asaPot                          ! A.S.A ポテンシャル
  real*4,intent(in)::elePot                          ! 静電ポテンシャル
  real*4,intent(in)::hydPot                          ! 水素結合ポテンシャル
  real*4,intent(in)::vdwPot                          ! van der Waals ポテンシャル
  real*4,intent(in)::surface                         ! 表面確率
  real*4,intent(in),dimension(MAX_ATOM,3)::cord      ! conformer 座標
  integer,intent(in)::atomNum                        ! 原子数
  character*2,intent(in),dimension(:)::atomicName    ! リガンド原子名
  character*80,intent(in)::name                      ! リガンド名
  real*4,intent(in)::currentRmsd                     ! rmsd
  real*4,intent(inout),dimension(:)::score           ! 上位スコア
  real*4,intent(inout),dimension(:,:)::energy        ! 上位スコアポテンシャル
  real*4,intent(inout),dimension(:,:,:)::rankCord    ! 上位スコア座標
  character*80,intent(inout),dimension(:)::ligandName ! 上位スコアリガンド名
  character*2,intent(inout),dimension(:,:)::atomName ! 上位スコア原子名
  real*4,intent(inout),dimension(:)::refRmsd         ! 上位スコア rmsd
  integer,intent(inout),dimension(:)::ligandAtomNum  ! 上位スコア原子数
  integer,intent(inout)::scoreNum                    ! 登録スコア数
```

戻り値：

　なし

機能：

入力したスコアが上位スコアである場合にスコア登録し、スコアの並べ替えを行う。

6.4.Display_Score

呼び出し形式：

```
  subroutine Display_Score(i)
```

引数：

```
  integer,intent(in)::i     ! 順位
```

戻り値：

　なし

機能：

当該リガンドに対する当該順位の登録スコアを表示する。

6.5.Display_TopScore

呼び出し形式：
```
subroutine Display_TopScore(unit, outNum, isUnited, rotNum)
```
引数：
```
integer,intent(in)::unit     ! 装置番号
integer,intent(in)::outNum   ! 出力スコア数
logical,intent(in)::isUnited ! 原子モデル
integer,intent(in)::rotNum   ! 回転可能結合数
```
戻り値：
　なし
機能：
全体の上位スコア（outNum 番目まで）を unit で指定された装置番号に出力する。

6.6.Display_LocalScore

呼び出し形式：
```
subroutine Display_LocalScore(unit, title, outNum)
```
引数：
```
character(*),intent(in)::title      ! ヘッダ行
integer,intent(in)::unit            ! 装置番号
```
戻り値：
　なし
機能：
当該リガンドの上位スコア（outNum 番目まで）を unit で指定された装置番号に出力する。

6.7.Add_AsaScore

呼び出し形式：
```
subroutine Add_AsaScore(numatom, numatop, cop, hank, hankp)
```
引数：
```
integer*4,intent(in)::numatom                   ! リガンド原子数
integer*4,intent(in)::numatop                   ! 蛋白原子数
real*4,intent(in),dimension(MAX_ATOM,3)::cop    ! 蛋白＋リガンド座標
real*4,intent(in),dimension(MAX_ATOM)::hank     ! リガンド原子半径
real*4,intent(in),dimension(MAX_ATOM)::hankp    ! 蛋白原子半径
```
戻り値：
　なし
機能：
当該リガンドと蛋白に対し、A.S.A の表面積のスコアを加算する。

6.8. Initial_TotalScore

呼び出し形式：

```
subroutine Initial_TotalScore(rankNum,ligandNum,dock,center,distance)
```

引数：

```
integer,intent(in)::rankNum          ! 全体の登録スコア数
integer,intent(in)::ligandNum        ! リガンドの原子数
type(Dock_t),intent(in)::dock        ! グローバルサーチ情報
real*4,dimension(3)::center          ! ポケット点中心座標
real*4,intent(in)::distance          ! ポケット内判定距離
```

戻り値：

なし

機能：

全体のスコアランキングの初期化を行う。

6.9. Initial_LocalScore

呼び出し形式：

```
subroutine Initial_LocalScore(rankNum, ligandName, atomName, atomNum)
```

引数：

```
integer,intent(in)::rankNum                          ! 各リガンドの登録スコア数
character*80,intent(in)::ligandName                  ! 当該リガンドの名称
character*2,intent(in),dimension(MAX_ATOM)::atomName ! 当該リガンドの各原子の名称
integer,intent(in)::atomNum                          ! 当該リガンドの原子数
```

戻り値：

なし

機能：

各リガンドでのスコア登録の初期化を行う。

6.10. Output_TopScoreCoordinate

呼び出し形式：

```
subroutine Output_TopScoreCoordinate(file, outNum, isUnited, rotNum)
```

引数：

```
type(File_t),intent(in)::file        ! スコアファイル情報
integer,intent(in)::outNum           ! 出力スコア数
logical,intent(in)::isUnited         ! 原子モデル
integer,intent(in)::rotNum           ! 回転可能結合数
```

戻り値：

なし

機能：

スコアファイルに全体のスコア（上位 outNum 番目まで）を出力する。

6.11.Write_TopScoreRanking

呼び出し形式：
```
  subroutine Write_TopScoreRanking(name, expID)
```
引数：
```
  character(*),intent(in)::name       ! 出力リスタートファイル名
  integer,intent(in)::expID           ! 実験 ID
```
戻り値：
　　なし
機能：
　リスタートファイルに現在の全体スコアと当該実験 ID を出力する。

6.12.Read_TopScoreRanking

呼び出し形式：
```
  subroutine Read_TopScoreRanking(name, expID)
```
引数：
```
  character(*),intent(in)::name       ! 入力リスタートファイル名
  integer,intent(out)::expID          ! 実験 ID
```
戻り値：
　　なし
機能：
　リスタートファイルから途中までのスコアと実行した実験 ID を読み込む。

6.13.Set_ReferenceCoordinate

呼び出し形式：
```
  subroutine Set_ReferenceCoordinate(cord, atomNum)
```
引数：
```
  real*4,intent(in),dimension(MAX_ATOM,3)::cord     ! 参照座標
  integer,intent(in)::atomNum                       ! 原子数
```
戻り値：
　　なし
機能：
　RMSD(root mean square distance)計算用のリガンド参照座標を設定する。

6.14.Get_TopScoreNum

呼び出し形式：
```
  function Get_TopScoreNum()
```
引数：
　　なし
戻り値：
```
  integer::スコア数
```
機能：
　現在のリガンドに対する上位スコアの数を読み込む。

6.15.Get_TopScoreCoordinate

呼び出し形式：
```
  subroutine Get_TopScoreCoordinate(cord, atomNum, rank)
```
引数：
```
  real*4,intent(out),dimension(MAX_ATOM,3)::cord    ! 原子座標
  integer,intent(in)::atomNum                       ! 原子数
  integer,intent(in)::rank                          ! 順位
```
戻り値：
　なし
機能：
　指定された順位の原子座標を読み込む。

6.16.Set_TopScoreCoordinate

呼び出し形式：
```
  subroutine Set_TopScoreCoordinate( cord, atomNum, rank )
```
引数：
```
  real*4,intent(in),dimension(MAX_ATOM,3)::cord    ! 原子座標
  integer,intent(in)::atomNum                      ! 原子数
  integer,intent(in)::rank                         ! 順位
```
戻り値：
　なし。
内容：
　指定された順位の原子座標を設定する。

6.17.Set_RefCoordinate

呼び出し形式：
```
  subroutine Get_RefCoordinate( cord, atomNum )
```
引数：
```
  real*4,intent(out),dimension(MAX_ATOM,3)::cord   ! 原子座標
  integer,intent(in)::atomNum                      ! 原子数
```
戻り値：
　なし。
内容：
　参照用座標を設定する。

6.18.Add_RichmondAsaScore

呼び出し形式：subroutine Add_RichmondAsaScore(ligandNumber, proteinNumber,
proteinCord)

引数：

```
integer*4,intent(in)::ligandNumber              リガンド原子数
integer*4,intent(in)::proteinNumber             蛋白原子数
real*4,intent(in),dimension(MAX_ATOM,3)::proteinCord     蛋白座標
```

戻り値：

なし

機能：

ローカルのトップスコアに登録されたリガンド原子の配置に対する Richmond での ASA 計算結果を現在のスコアに反映する。

ASA エネルギー =
系全体 ASA エネルギー - 蛋白内 ASA エネルギー - リガンド内 ASA エネルギー

6.19.Output_TopScoreMol2File

呼び出し形式：

```
subroutine Output_TopScoreMol2File(file,outNum,isUnited,rotNum,      &
                                   molName, numBond, numSubst, numFeat,      &
                                   numSets, molType, chargeType, molStatus, &
                                   sybylType, substID, substName,            &
                                   isReadCharge, charge, atomStatus,         &
                                   bondPair, bondOrderStr, bondStatus )
```

引数：

```
type(File_t),intent(in)::file                          ! 出力ファイル情報
integer,intent(in)::outNum                             ! 出力構造数
logical,intent(in)::isUnited                           ! 原子モデル
integer,intent(in)::rotNum                             ! 回転可能結合数
character(NAME_SIZE),intent(in)::molName               ! 分子名
integer,intent(in)::numBond                            ! 結合数
integer,intent(in)::numSubst                           ! 残基数
integer,intent(in)::numFeat                            ! number of features
integer,intent(in)::numSets                            ! number of sets
integer,intent(in)::molType                            ! 分子タイプ
integer,intent(in)::chargeType                         ! 分子電荷タイプ
integer,intent(in)::molStatus                          ! 分子情報
character*5,dimension(MAX_ATOM),intent(in)::sybylType  ! SYBYL 原子タイプ
integer,dimension(MAX_ATOM),intent(in)::substID        ! 残基番号
character*10,dimension(MAX_ATOM),intent(in)::substName ! 残記名
logical,dimension(MAX_ATOM),intent(in)::isReadCharge   ! 電荷読み込みフラグ
real*4,dimension(MAX_ATOM),intent(in)::charge          ! 電荷
integer,dimension(MAX_ATOM),intent(in)::atomStatus     ! 原子情報
integer,dimension(MAX_ATOM,2),intent(in)::bondPair     ! 結合原子 ID リスト
integer,dimension(MAX_ATOM),intent(in)::bondOrderStr   ! 結合次数
integer,dimension(MAX_ATOM),intent(in)::bondStatus     ! 結合情報
```

戻り値：

なし

機能：

上位スコア（outNum 番目まで）の構造を unit で指定された装置番号に mol2 形式で出力する。

6.20. Calc_VariousScore

呼び出し形式：Calc_VariousScore(dG_Score, hit_Score, MTS_score , &
rotNum, isUnited , i)

引数：

```
  real*4,intent(out)::dG_Score    ! dG-score
  real*4,intent(out)::hit_Score   ! hit-optimized score
  real*4,intent(out)::MTS_Score   ! MTS score
  integer,intent(in)::rotNum      ! number of rotative atoms
  logical,intent(in)::isUnited    ! united atom model flag
                                  ! ( true for united atom model )
  integer,intent(in)::i           ! rank
```

戻り値：

なし

機能：

上位 i 番目のスコアを読み込み、⊿G スコア、hit-optimized スコア、MTS スコアを計算する。

7.SievIO_Method

ファイル名：SievIO_Method.f90

sievgene のファイル入出力に関するメソッドを定義する。

7.1.Read_PDBFile

呼び出し形式：

```
subroutine Read_PDBFile(numatom,cseq,ato,at2,co,charge, file)
```

引数：

```
integer*4,intent(out)::numatom                          ! 原子数
character*3,intent(out),dimension(MAX_ATOM)::cseq       ! 残基名
character*1,intent(out),dimension(MAX_ATOM)::ato        ! 原子名１文字目
character*1,intent(out),dimension(MAX_ATOM)::at2        ! 原子名２文字目
real*4,intent(out),dimension(MAX_ATOM,3)::co            ! 原子座標
real*4,intent(out),dimension(MAX_ATOM)::charge          ! 電荷
type(File_t),intent(in)::file                           ! PDB ファイル情報
```

戻り値：

なし

機能：

PDB 形式ファイルから原子座標、原子名を読み込む。
また、各原子名に対応した電荷を設定する。

7.2.Read_PocketPoint

呼び出し形式：

```
subroutine Read_PocketPoint(numatom,co, file)
```

引数：

```
integer*4,intent(out)::numatom                          ! ポケット点数
real*4,intent(inout),dimension(MAX_ATOM,3)::co          ! ポケット点座標
type(File_t),intent(in)::file                           ! ポケット点ファイル情報
```

戻り値：

なし

機能：

PDB 形式ファイルから、ポケット点の座標を読みこむ。

7.3.Input_Mol2File

呼び出し形式：

```
function Input_Mol2File(file,                                                &
                        molName, numBond, numSubst, numFeat, numSets,        &
                        molType, chargeType, molStatus,                      &
                        atomNum, atomName, cord, atomType, radius, charge,   &
                        sybylType, substID, substName,                       &
                        isReadCharge, atomStatus,                            &
                        bondNum, bondPair, bondOrder, bondOrderStr,          &
                        bondStatus, atomNo)
```

引数：

```
type(File_t),intent(in)::file                              ! mol ファイル情報
character(NAME_SIZE),intent(out)::molName                  ! 分子名
integer,intent(out)::numBond                               ! 結合数
integer,intent(out)::numSubst                              ! 残基数
integer,intent(out)::numFeat                               ! number of features
integer,intent(out)::numSets                               ! number of sets
integer,intent(out)::molType                               ! 分子タイプ
integer,intent(out)::chargeType                            ! 分子電荷タイプ
integer,intent(out)::molStatus                             ! 分子情報
integer,intent(out)::atomNum                               ! リガンド原子の原子数
character*2,intent(out),dimension(MAX_ATOM)::atomName      ! リガンド原子の原子名称
real*4,intent(out),dimension(MAX_ATOM,3)::cord             ! リガンド原子の座標
integer,intent(out),dimension(MAX_ATOM)::atomType          ! リガンド原子の原子タイプ
real*4,intent(out),dimension(MAX_ATOM)::radius             ! リガンド原子の半径
real*4,intent(out),dimension(MAX_ATOM)::charge             ! リガンド原子の電荷
character*5,intent(out),dimension(MAX_ATOM)::sybylType     ! SYBYL 原子タイプ
integer,intent(out),dimension(MAX_ATOM)::substID           ! 残基番号
character*10,intent(out),dimension(MAX_ATOM)::substName    ! 残基名
logical,intent(out),dimension(MAX_ATOM)::isReadCharge      ! 電荷読み込みフラグ
integer,intent(out),dimension(MAX_ATOM)::atomStatus        ! 原子情報
integer,intent(out)::bondNum                               ! bond 数
integer,intent(out),dimension(MAX_ATOM,2)::bondPair        ! bond ペアの原子 ID
real*4,intent(out),dimension(MAX_ATOM)::bondOrder          ! bond 結合度
integer,intent(out),dimension(MAX_ATOM)::bondOrderStr      ! bond 結合度（文字列）
integer,intent(out),dimension(MAX_ATOM)::bondStatus        ! bond 情報
integer,intent(out),dimension(MAX_ATOM)::atomNo            ! リガンド原子の元素番号
```

戻り値：

```
logical::endFile                                           ! 読み込み終了フラグ
```

機能：

MOL2 形式のファイルから分子単位にリガンドの分子情報、原子情報、結合情報を読み込む。
ファイルの終りを検出した場合、endFile フラグを true に設定する。

7.4. Input_ProteinData

呼び出し形式：

```
subroutine Input_ProteinData(proteinFile,pocketFile,                    &
                             numatop,residName,atomName,cord,chargp,    &
                             hankp,numdonar_p,numacce_p,don_cop,acc_cop, &
                             yuragi_list,isHotSpot,atp,atp2,getap,      &
                             atomType,listp_acc,                        &
                             pointNum,point)
```

引数：

```
type(File_t),intent(in)::proteinFile                    ! 蛋白ファイル情報
type(File_t),intent(in)::pocketFile                     ! ポケット点ファイル情報
integer,intent(out)::numatop                            ! 蛋白原子数
character*3,intent(out),dimension(MAX_ATOM)::residName  ! 蛋白残基名
character*2,intent(out),dimension(MAX_ATOM)::atomName   ! 蛋白原子名
real*4,intent(out),dimension(MAX_ATOM,3)::cord          ! 蛋白原子座標
real*4,intent(out),dimension(MAX_ATOM)::chargp          ! 蛋白原子電荷
real*4,intent(out),dimension(MAX_ATOM)::hankp           ! 蛋白原子半径
integer,intent(out)::numdonar_p                         ! 蛋白ドナー数
integer,intent(out)::numacce_p                          ! 蛋白アクセプタ数
real*4,intent(out),dimension(MAX_ATOM,3)::don_cop       ! ドナー座標
real*4,intent(out),dimension(MAX_ATOM,3)::acc_cop       ! アクセプタ座標
real*4,intent(out),dimension(MAX_ATOM)::yuragi_list     ! 蛋白原子の重み
integer,intent(out),dimension(MAX_ATOM)::isHotSpot      ! hotspot フラグ
character*1,dimension(MAX_ATOM)::atp                    ! 原子名１文字目
character*1,dimension(MAX_ATOM)::atp2                   ! 原子名２文字目
real*4,intent(in)::getap                                ! 原子半径オフセット
integer,intent(out),dimension(MAX_ATOM)::atomType       ! 原子タイプ
integer,intent(out),dimension(MAX_ATOM)::listp_acc      ! アクセプタ原子 ID
integer,intent(out)::pointNum                           ! ポケット点数
real*4,intent(out),dimension(MAX_ATOM,3)::point         ! ポケット点座標
```

戻り値：

なし

機能：

蛋白ファイルおよびポケット点読み込みの主制御を行う。

7.5. Input_TargetFile

呼び出し形式：

```
Input_TargetFile(target, grid)
```

引数：

```
type(File_t),intent(in)::target          ! ファイル情報
type(Grid_t),intent(inout)::grid         ! grid 情報
```

戻り値：

なし

機能：

Hash 探索の対象原子指定ファイルを読み込む。

8.SievInput_Option

ファイル名：SievInput_Option.f90

sievgeneの制御ファイル入力に関するメソッドを定義する。

8.1.Input_SievInputOption

呼び出し形式：
```
subroutine Input_SievInputOption(input, main)
```
引数：
```
type(SievInput_t),intent(out)::input          ! sievgene用input情報
type(Main_t),intent(out)::main                ! sievgeneの実験全体情報
```
戻り値：
　なし
機能：
sievgeneのINPUTフェーズのオプションを入力し、input情報に値を設定する。

8.2.Input_GridOption

呼び出し形式：
```
subroutine Input_GridOption(grid)
```
引数：
```
type(Grid_t),intent(out)::grid          ! gridフェーズ情報
```
戻り値：
　なし
機能：
sievgeneのGRIDフェーズのオプションを入力し、grid情報に値を設定する。

8.3.Input_ConfOption

呼び出し形式：
```
subroutine Input_ConfOption(conf)
```
引数：
```
type(Conf_t),intent(out)::conf          ! conf情報
```
戻り値：
　なし
機能：
sievgeneのCONFフェーズのオプションを入力し、conf情報に値を設定する。

8.4.Input_DockOption

呼び出し形式：
```
  subroutine Input_DockOption(dock)
```
引数：
```
  type(Dock_t),intent(out)::dock          ! dock 情報
```
戻り値：
　なし
機能：
sievgene の DOCK フェーズのオプションを入力し、dock 情報に値を設定する。

8.5.Input_SievOutputOption

呼び出し形式：
```
  subroutine Input_SievOutputOption(output)
```
引数：
```
  type(SievOutput_t),intent(out)::output ! output 情報
```
戻り値：
　なし
機能：
sievgene の OUTPUT フェーズのオプションを入力し、output 情報に値を設定する。

9.SievMath_Method

ファイル名：SievMath_Method.f90

sievgene で使用する数値計算のメソッドを定義する。

9.1.HOUSE

呼び出し形式：

```
SUBROUTINE HOUSE(NDIM,N,A,AL,BE,CO,W,P,Q)
```

引数：

```
INTEGER*4,intent(in)::NDIM                    ! matrix row size
INTEGER*4,intent(in)::N                       ! matrix column size
REAL*4,intent(inout),dimension(NDIM,N)::A     ! target symmetirc matrix
REAL*4,intent(out),dimension(N)::AL           ! i-th diagonal element
REAL*4,intent(out),dimension(N)::BE           ! i-th subdiagonal element
REAL*4,intent(out),dimension(N)::CO           ! normalization factor of 'W'
REAL*4,intent(out),dimension(N)::W            ! K-th step A(J,K)
REAL*4,intent(out),dimension(N)::P            ! CO(K)*(SUM(A(K,1:N)-A(K,K))
REAL*4,intent(out),dimension(N)::Q            ! P(I) - CO(K)*SUM(P*W)/2 - S*W
```

戻り値：

なし

機能：

Householder 法で行列の対角化を行う。

9.2.BISEC

呼び出し形式：

```
SUBROUTINE BISEC(N,NEIG,NLS,EIG,EPS,AL,BE,B2)
```

引数：

```
INTEGER,intent(in)::N                       ! matrix size
INTEGER,intent(in)::NEIG                    ! eigenvalue size
INTEGER,intent(in)::NLS                     ! largest / smallest flag
REAL*4,intent(out),dimension(NEIG)::EIG     ! eigenvalues
REAL*4,intent(in)::EPS                      ! relative error tolerance
REAL*4,intent(in),dimension(N)::AL          ! i-th diagonal element of matrix
REAL*4,intent(inout),dimension(N)::BE       ! i-th subdiagonal element
REAL*4,intent(inout),dimension(N)::B2       ! K-th step matrix(J,K)
```

戻り値：

なし

機能：

Beisection 法で行列の固有地を求める。

9.3.INVITR

呼び出し形式：

```
  SUBROUTINE INVITR(NDIM,N,A,NEIG,EIG,NVEC,VEC,AL,                    &
                    BE,CO,DI,BL,BU,BV,CM,LEX)
```

引数：

```
  INTEGER,intent(in)::NDIM                        ! matrix row size
  INTEGER,intent(in)::N                           ! matrix column size
  REAL*4,intent(in),dimension(NDIM,N)::A          ! symmetric matrix
  INTEGER,intent(in)::NEIG                        ! eigenvalue size
  REAL*4,intent(in),dimension(NEIG)::EIG          ! eigen value
  INTEGER,intent(inout)::NVEC                     ! eigenvector column size
  REAL*4,intent(out),dimension(NDIM,NVEC)::VEC    ! eigenvector
  REAL*4,intent(in),dimension(N)::AL              ! i-th diagonal element
  REAL*4,intent(in),dimension(N)::BE              ! i-th subdiagonal element
  REAL*4,intent(in),dimension(N)::CO              ! normlization factor of 'W'
  REAL*4,intent(out),dimension(N)::DI             ! i-th diagonal EIG(K)-AL(I)
  REAL*4,intent(out),dimension(N)::BL             ! i-th lower subdiagonal
  REAL*4,intent(out),dimension(N)::BU             ! i-th upper subdiagonal
  REAL*4,intent(out),dimension(N)::BV             ! i-th 2nd upper subdiagonal
  REAL*4,intent(out),dimension(N)::CM             ! i-th multiplier of Gauss
  INTEGER,intent(out),dimension(N)::LEX           ! row exchange flag
```

戻り値：

　なし

機能：

inverse iteration 法で行列の固有ベクトルを求める。

9.4.ROTMTX

呼び出し形式：

```
  SUBROUTINE ROTMTX(COR,R)
```

引数：

```
  REAL*4,intent(inout),dimension(3)::COR        ! 回転前／回転後座標
  REAL*4,intent(in),dimension(3,3)::R           ! 回転行列
```

戻り値：

　なし

機能：

入力した座標を入力した行列で回転させる。

9.5.BSTFT2

呼び出し形式：

```
  SUBROUTINE BSTFT2(NTATM,MAXATM,CORD,TCORD,LEX,                    &
                    R,CM,CMT,RMSD,IFLAG,Q)
```

引数：

```
  INTEGER,intent(in)::NTATM                          ! 原子数
  INTEGER,intent(in)::MAXATM                         ! 原子座標配列の１次元目宣言サイズ
  REAL*4,intent(inout),dimension(MAXATM,3)::CORD     ! 入力原子座標
  REAL*4,intent(in),dimension(MAXATM,3)::TCORD       ! 原子の参照座標
  INTEGER,intent(inout),dimension(MAXATM)::LEX       ! 行の変換フラグ
  REAL*4,intent(out),dimension(3,3)::R               ! 回転行列
  REAL*4,intent(out),dimension(3)::CM                ! 入力原子の中心座標
  REAL*4,intent(out),dimension(3)::CMT               ! 参照座標の中心座標
  REAL*4,intent(out)::RMSD                           ! root mean square distance
  INTEGER,intent(in)::IFLAG                          ! 座標の更新フラグ
  REAL*4,intent(out),dimension(4)::Q                 ! 回転行列の４元数
```

戻り値：

なし

機能：

least-square fitting を行い、参照座標と入力座標の RMSD を求める。

座標更新フラグが１の場合、参照座標の固有ベクトルに対し、入力座標の固有ベクトルが一致するように入力座標を更新する。

9.6.EIGEN

呼び出し形式：

```
  SUBROUTINE EIGEN(NDIM,N,A,NEIG,NLS,EIG,NVEC,VEC,EPS,WORK,LEX)
```

引数：

```
  INTEGER,intent(in)::NDIM                           ! dimension of row
  INTEGER,intent(in)::N                              ! dimension of columns
  REAL*4,intent(in),dimension(NDIM,N)::A             ! symmetric matrix
  INTEGER,intent(in)::NEIG                           ! eigenvalues size
  INTEGER,intent(in)::NLS                            ! largest / smallest flag
  REAL*4,intent(in),dimension(NEIG)::EIG             ! eigenvalues
  INTEGER,intent(in)::NVEC                           ! eigenvector column size
  REAL*4,intent(out),dimension(NDIM+1,NVEC)::VEC     ! eigenvector
  REAL*4,intent(in)::EPS                             ! relative error tolerance
  REAL*4,intent(in),dimension(NDIM,8)::WORK          ! intermediate results
  INTEGER,intent(in),dimension(N)::LEX               ! row exchange flag
```

戻り値：

なし

機能：

固有値、固有ベクトル計算の主制御を行う。

9.7.Make_SphereSurfacePoint

呼び出し形式：

```
subroutine Make_SphereSurfacePoint(R1,scal,cir,jmax)
```

引数：

```
real*4,intent(in)::R1                          ! 半径
real*4,intent(in)::scal                        ! A.S.A の点の距離
real*4,intent(out),dimension(maxpot,5)::cir    ! 生成した A.S.A の点座標
integer*4,intent(out)::jmax                    ! 生成した A.S.A の点の個数
```

戻り値：

なし

機能：

原点を中心に、指定された半径の球上に指定された間隔の点を生成する。

9.8.BONDD

呼び出し形式：

```
SUBROUTINE BONDD(X1,Y1,Z1,RR,IUNT)
```

引数：

```
real*4,intent(in),dimension(4)::X1,Y1,Z1     ! 点の座標
real*4,intent(out)::RR                       ! ２点間の距離
integer*4,intent(in)::IUNT                   ! unit flag (1 = atomic unit)
```

戻り値：

なし

機能：

２点間の距離を計算する。

9.9.BONDA

呼び出し形式：

```
SUBROUTINE BONDA(X1,Y1,Z1,AA,No1,No2,No3)
```

引数：

```
real*4,intent(in),dimension(4)::X1,Y1,Z1       ! ３点の座標
real*4,intent(out)::AA                         ! angle 角
integer*4,intent(in)::No1, No2, No3            ! point ID
```

戻り値：

なし

機能：

３点の座標が構成する angle 角を計算する。

9.10.DHDRA

呼び出し形式：
```
  SUBROUTINE DHDRA(X1,Y1,Z1,DD,N1,N2,N3)
```
引数：
```
  real*4,intent(in),dimension(4)::X1,Y1,Z1          ! ４点の座標
  real*4,intent(out)::DD                            ! dihedral 角
  integer*4,intent(in),dimension(999)::N1,N2,N3     ! not used
```
戻り値：
　なし
機能：
　４点の座標が構成する dihedral(torsion)角を計算する。

9.11.GENXYZ

呼び出し形式：
```
  SUBROUTINE GENXYZ(RXYZ,BAT,XYZ)
```
引数：
```
  REAL*8,intent(in),dimension(3,3)::RXYZ            ! bond, angle, torsion の座標
  REAL*8,intent(in),dimension(3)::BAT               ! bond, angle, torsion 値
  REAL*8,intent(out),dimension(3)::XYZ              ! 生成座標
```
戻り値：
　なし
機能：
　bond, angle, torsion のパラメータから当該点の座標を計算する。

9.12.VCTPRD

呼び出し形式：
```
  SUBROUTINE VCTPRD(R1,R2,R3)
```
引数：
```
  REAL*8,intent(in),dimension(3)::R1              ! first vector
  REAL*8,intent(in),dimension(3)::R2              ! second vector
  REAL*8,intent(out),dimension(3)::R3             ! vector product
```
戻り値：
　なし
機能：
　２つのベクトルの外積を計算する。

9.13.modulate

呼び出し形式：
```
  subroutine modulate(q,qo,R)
```
引数：
```
  REAL*4,intent(out),dimension(4)::Q       ! 生成した４元数
  REAL*4,intent(in),dimension(4)::qo       ! 入力４元数
  REAL*4,intent(out),dimension(3,3)::R     ! 生成した４元数に対応する回転行列
```
戻り値：
　　なし
機能：
　入力した４元数に乱数を加え、新規の４元数とこれに対応する回転行列を生成する。

9.14.Create_ZmatrixParameter

呼び出し形式：
```
  SUBROUTINE Create_ZmatrixParameter (K,N1,N2,N3,R,A,D,X,Y,Z)
```
引数：
```
  integer*4,intent(in),dimension(999)::N1        ! 1-2 の原子 ID
  integer*4,intent(in),dimension(999)::N2        ! 1-3 の原子 ID
  integer*4,intent(in),dimension(999)::N3        ! 1-4 の原子 ID
  real*4,intent(out),dimension(999)::R           ! 生成した 1-2 原子間距離
  real*4,intent(out),dimension(999)::A           ! 生成した 1-2-3 angle 角
  real*4,intent(out),dimension(999)::D           ! 生成した 1-2-3-4 torsion 角
  real*4,intent(in),dimension(999)::X            ! 原子の x 座標
  real*4,intent(in),dimension(999)::Y            ! 原子の y 座標
  real*4,intent(in),dimension(999)::Z            ! 原子の z 座標
```
戻り値：
　　なし
機能：
　Z-matrix の bond 距離、angle 角、torsion 角を計算する。

9.15.Exec_QuickSortSingle

名称：Exec_QuickSortSingle
呼び出し形式：subroutine Exec_QuickSortSingle(data, tag, num)
引数：
```
    real*4::data(*)          ソートデータ
    integer::tag(*)          ソート前の位置
    integer::num             ソートデータ数
```
戻り値：
　　なし
機能：
　　短精度の浮動小数点データのクイックソートの初期処理と開始を行う。

9.16.Sort_PartialData

名称：Sort_PartialData
呼び出し形式：recursive subroutine Sort_PartialData (data, tag, low, high)
引数：

real*4::data(*)	ソートデータ
integer::tag(*)	ソート前の位置
integer::low	ソート開始添字
integer::high	ソート最終添字

戻り値：
　なし
機能：
　ソート開始添え字から、ソート最終添字までのデータのクイックソートを行う。
　要素数が少ない場合、クイックソートよりも逐次ソートのほうが高速であるため、ソートする要素が 14 以下の場合、逐次ソートを実行する。
　ソート終了後、最小値：中間値-1、中間値+1：最大値までの要素に対し、再帰的にクイックソートを実行する。

9.17.Exec_SequentialSort

名称：Exec_SequentialSort
呼び出し形式：subroutine Exec_SequentialSort (data, tag, low, high)
引数：

real*4::data(*)	ソートデータ
integer::tag(*)	ソート前の位置
integer::low	ソート開始添字
integer::high	ソート最終添字

戻り値：
　なし
機能：
　ソート開始添え字から、ソート最終添字までのデータの逐次ソートを行う。

10.SievMin_Method

ファイル名：SievMin_Method.f90

sievgene のローカルサーチのメソッドを定義する。

10.1.Exec_SievMinimize

呼び出し形式：

```
  subroutine Exec_SievMinimize(minimize, cord)
```

引数：

```
  type(Minimize_t),intent(inout)::minimize          ! minimize 情報
  real*4,intent(in),dimension(MAX_ATOM,3)::cord     ! リガンド原子座標
```

戻り値：

　なし

機能：

　当該リガンドに対し minimization によるローカルサーチを行い、スコア登録を行う。

11.SievMonitor

ファイル名：SievMonitor.f90

sievgene のデータ表示のメソッドを定義する。

11.1.Display_SievInputOption

呼び出し形式：

```
  subroutine Display_SievInputOption(input, main, unit)
```

引数：

```
  type(SievInput_t),intent(in)::input       ! INPUT フェーズ情報
  type(Main_t),intent(in)::main             ! 実験全体情報
  integer,intent(in)::unit                  ! 出力装置番号
```

戻り値：

なし

機能：

INPUT フェーズの情報を指定された装置番号に出力する。

11.2.Display_GridOption

呼び出し形式：

```
  subroutine Display_GridOption(grid, unit)
```

引数：

```
  type(Grid_t),intent(in)::grid         ! GRID フェーズ情報
  integer,intent(in)::unit              ! 論理装置番号
```

戻り値：

なし

機能：

GRID フェーズの情報を指定された装置番号に出力する。

11.3.Display_ConfOption

呼び出し形式：

```
  subroutine Display_ConfOption(conf, unit)
```

引数：

```
  type(Conf_t),intent(in)::conf         ! CONF フェーズ情報
  integer,intent(in)::unit              ! 論理装置番号
```

戻り値：

なし

機能：

CONF フェーズの情報を指定された装置番号に出力する。

11.4.Display_DockOption

呼び出し形式：
```
  subroutine Display_DockOption(dock, unit)
```
引数：
```
  type(Dock_t),intent(in)::dock         ! DOCK フェーズ情報
  integer,intent(in)::unit              ! 論理装置番号
```
戻り値：
　　なし
機能：
　DOCK フェーズの情報を指定された装置番号に出力する。

11.5.Display_SievOutputOption

呼び出し形式：
```
  subroutine Display_SievOutputOption(output, unit)
```
引数：
```
  type(SievOutput_t),intent(in)::output     ! OUTPUT フェーズ情報
  integer,intent(in)::unit                  ! 論理装置番号
```
戻り値：
　　なし
機能：
　OUTPUT フェーズの情報を指定された装置番号に出力する。

12.SievObject_Method

ファイル名：SievObject_Method.f90

原子、分子に関するメソッドを定義する。

12.1.Set_AtomType

呼び出し形式：

```
subroutine Set_AtomType(numatom,cseq,ato,at2,co,nat,geta,                    &
                        hank,charge,numdonar_p,numaccep_p,don_cop,acc_cop,   &
                        listp_acc,list_hot,yuragi_list)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
character*3,intent(in),dimension(MAX_ATOM)::cseq       ! 残基名
character*1,intent(in),dimension(MAX_ATOM)::ato        ! 原子名１文字目
character*1,intent(in),dimension(MAX_ATOM)::at2        ! 原子名２文字目
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 原子座標
integer*4,intent(out),dimension(MAX_ATOM)::nat         ! 原子タイプ
real*4,intent(in)::geta                                ! 原子半径オフセット
real*4,intent(out),dimension(MAX_ATOM)::hank           ! 原子半径
real*4,intent(out),dimension(MAX_ATOM)::charge         ! 原子電荷
integer*4,intent(inout)::numdonar_p                    ! ドナー数
integer*4,intent(inout)::numaccep_p                    ! アクセプタ数
real*4,intent(out),dimension(MAX_ATOM,3)::don_cop      ! ドナー座標
real*4,intent(out),dimension(MAX_ATOM,3)::acc_cop      ! アクセプタ座標
integer*4,intent(out),dimension(MAX_ATOM)::listp_acc   ! アクセプタ原子 ID
integer*4,intent(out),dimension(MAX_ATOM)::list_hot    ! hotspot フラグ
real*4,intent(out),dimension(MAX_ATOM)::yuragi_list    ! 原子の重み
```

戻り値：

なし

機能：

蛋白原子の固有情報設定とドナー、アクセプタ、重みを設定の主制御を行う。

12.2.Set_AcceptDonner

呼び出し形式：

```
  subroutine Set_AcceptDonner(numatom,co,nbond,numbond,mat,                    &
                              num_don_l,list_don,list_do2,list_do3,list_do4,   &
                              num_acc_l,list_acc,n_type_lhd,n_type_lha,        &
                              h_vecl,use_atom,speed)
```

引数：

```
  integer*4,intent(in)::numatom                          ! 原子数
  real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 原子座標
  integer*4,intent(inout),dimension(MAX_ATOM,2)::nbond   ! bond ペア原子 ID
  integer*4,intent(in)::numbond                          ! bond 数
  integer*4,intent(in),dimension(MAX_ATOM)::mat          ! 元素番号
  integer*4,intent(inout)::num_don_l                     ! リガンドドナー数
  integer*4,intent(out),dimension(MAX_ATOM)::list_don    ! リガンドドナー原子 ID
  integer*4,intent(out),dimension(MAX_ATOM)::list_do2    ! リガンドドナー原子 ID
  integer*4,intent(out),dimension(MAX_ATOM)::list_do3    ! リガンドドナー原子 ID
  integer*4,intent(out),dimension(MAX_ATOM)::list_do4    ! リガンドドナー原子 ID
  integer*4,intent(inout)::num_acc_l                     ! リガンドアクセプタ数
  integer*4,intent(out),dimension(MAX_ATOM)::list_acc    ! リガンドアクセプタ ID
  integer*4,intent(out),dimension(MAX_ATOM)::n_type_lha  ! リガンドアクセプタ種別
  integer*4,intent(out),dimension(MAX_ATOM)::n_type_lhd  ! リガンドドナー種別
  real*4,intent(out),dimension(MAX_ATOM,3)::h_vecl       ! 原子間距離
  logical,intent(out),dimension(MAX_ATOM)::use_atom      ! 接合面判定での原子フラグ
  integer*4,intent(in)::speed                            ! ドッキング方法
```

戻り値：

なし

機能：

リガンドのドナー／アクセプタ情報を設定する。

12.3.Select_NewConf

呼び出し形式：

```
  subroutine Select_NewConf(numatom,confomer,no_conf)
```

引数：

```
  integer*4,intent(in)::numatom                                ! 原子数
  real*4,intent(inout),dimension(MAX_LIGAND,3,MAX_CONF)::confomer
                                                               ! 生成 confoemer 座標
  integer*4,intent(inout)::no_conf                             ! 生成 conformer 数
```

戻り値：

なし

機能：

上位スコアに登録された conformer と異なる conformer を抽出する。

12.4.Get_NearAtom

呼び出し形式：

```
subroutine Get_NearAtom(numatom,co,npoint,point,ntable)
```

引数：

```
integer*4,intent(in)::numatom                          ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 原子座標
integer*4,intent(in)::npoint                           ! ポケット点数
real*4,intent(in),dimension(MAX_ATOM,3)::point         ! ポケット点座標
integer*4,intent(out),dimension(MAX_ATOM)::ntable      ! 隣接フラグ
```

戻り値：

なし

機能：

ポケット点に近い(4.0Å以内)の原子を判定し、近い場合は隣接フラグを１とする。

12.5.Generate_Pocket

呼び出し形式：

```
subroutine Generate_Pocket(numatom,ato,co,hank,npoint,point,charge,        &
                           xyz_sas,nsas2atom,n_sas,nat,numaccep_p,         &
                           listp_acc,ngh,nghl)
```

引数：

```
integer*4,intent(in)::numatom                          ! 蛋白原子数
character*1,intent(in),dimension(MAX_ATOM)::ato        ! 蛋白原子名１文字目
real*4,intent(in),dimension(MAX_ATOM,3)::co            ! 蛋白原子座標
real*4,intent(inout),dimension(MAX_ATOM)::hank         ! 蛋白原子半径
integer*4,intent(in)::npoint                           ! ポケット点数
real*4,intent(in),dimension(MAX_ATOM,3)::point         ! ポケット点座標
real*4,intent(in),dimension(MAX_ATOM)::charge          ! 蛋白原子電荷
real*4,intent(inout),dimension(MAX_SURFACE,3)::xyz_sas  ! A.S.A 点座標
integer*4,intent(inout),dimension(MAX_SURFACE)::nsas2atom  ! A.S.A 点の属する原子 ID
integer*4,intent(out)::n_sas                           ! A.S.A 点数
integer*4,intent(in),dimension(MAX_ATOM)::nat          ! 蛋白原子タイプ
integer*4,intent(in)::numaccep_p                       ! 蛋白アクセプタ数
integer*4,intent(in),dimension(MAX_ATOM)::listp_acc     ! アクセプタ原子 ID
integer*4,intent(out),dimension(18,18,18,3,1500)::ngh   ! ポケット接合面座標リスト
integer*4,intent(out),dimension(18,18,18)::nghl        ! ポケット接合面数
```

戻り値：

なし

機能：

蛋白ポケットの以下の情報を作成する。

(1)A.S.A 点

(2)ポケット上の接合面(三角形状)のリスト

12.6.Regenerate_Pocket

呼び出し形式：

```
subroutine Regenerate_Pocket(nat,n_sas,numatom,ngh,nghl,reg_coo)
```

引数：

```
integer*4,intent(in),dimension(MAX_ATOM)::nat            ! 蛋白原子タイプ
integer*4,intent(inout)::n_sas                           ! A.S.A 数
integer*4,intent(in)::numatom                            ! 原子数
integer*4,intent(out),dimension(18,18,18,3,1500)::ngh    ! 接合面座標リスト
integer*4,intent(out),dimension(18,18,18)::nghl          ! 接合面数
real*4,intent(in),dimension(200,300,3)::reg_coo
```

戻り値：

なし

機能：

ポケット上の接合面を再構成する。

12.7.checkmesh

呼び出し形式：

```
subroutine checkmesh(m_sas,xyz_sa1,nsas2ato1,                                    &
                     numatom,co,charge,nat,n_sas,xyz_sas,nsas2atom)
```

引数：

```
integer*4,intent(in)::m_sas                              ! 入力 A.S.A 点数
real*4,intent(in),dimension(MAX_SURFACE,3)::xyz_sa1      ! 入力 A.S.A 点座標
integer*4,intent(in),dimension(MAX_SURFACE)::nsas2ato1   ! 入力 A.S.A 点所属子 ID
integer*4,intent(in)::numatom                            ! 原子数
real*4,intent(in),dimension(MAX_ATOM,3)::co              ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::charge            ! 原子電荷
integer*4,intent(in),dimension(MAX_ATOM)::nat            ! 原子タイプ
integer*4,intent(out)::n_sas                             ! 新規 A.S.A 点数
real*4,intent(out),dimension(MAX_SURFACE,3)::xyz_sas     ! 新規 A.S.A 点座標
integer*4,intent(out),dimension(MAX_SURFACE)::nsas2atom  ! 新規 A.S.A 点所属原子 ID
```

戻り値：

なし

機能：

A.S.A 点でポテンシャルの絶対値の小さい点を削除する。

12.8.Move_AtomCord

呼び出し形式：

```
subroutine Move_AtomCord(numatom,co_new2,co_new,dx,dy,dz)
```

引数：

```
integer*4,intent(in)::numatom                              ! 原子数
real*4,intent(in),dimension(MAX_LIGAND,3)::co_new2         ! 入力原子座標
real*4,intent(out),dimension(MAX_LIGAND,3)::co_new         ! 新規原子座標
real*4,intent(in)::dx                                      ! x 方向オフセット
real*4,intent(in)::dy                                      ! y 方向オフセット
real*4,intent(in)::dz                                      ! z 方向オフセット
```

戻り値：

なし

機能：

入力した原子座標に x, y, z 方向のオフセットを付加した座標を生成する。

12.9.Set_SideChain

呼び出し形式：

```
subroutine Set_SideChain(numatom,cseq,ato,at2,co,hank,npoint,point,          &
                         no_id,n_l_stem,n_n_br,n_l_br,n_s_a,n_s_d,           &
                         n_l_a,n_l_d1,n_l_d2,h_rotate,n_type_a,n_type_d,     &
                         h_vecp)
```

引数：

```
integer,intent(in)::numatom                                    ! 原子数
character*3,intent(in),dimension(MAX_ATOM)::cseq               ! 残基名
character*1,intent(in),dimension(MAX_ATOM)::ato                ! 原子名１文字目
character*1,intent(in),dimension(MAX_ATOM)::at2                ! 原子名２文字目
real*4,intent(in),dimension(MAX_ATOM,3)::co                    ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::hank                    ! 原子半径
integer*4,intent(in)::npoint                                   ! ポケット点数
real*4,intent(in),dimension(MAX_ATOM,3)::point                 ! ポケット点座標
integer*4,intent(out)::no_id                                   ! 側鎖数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_stem    ! 土台部分のリスト
integer*4,intent(out),dimension(MAX_SC)::n_n_br                ! 回転部分の原子数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_br      ! 回転部分のリスト
integer*4,intent(out),dimension(MAX_SC)::n_s_a                 ! 回転部分のアクセプタ数
integer*4,intent(out),dimension(MAX_SC)::n_s_d                 ! 回転部分のドナー数
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_a       ! 回転部分のアクセプタリスト
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_d1      ! 回転部分のドナーリスト
integer*4,intent(out),dimension(MAX_SC,MAX_SCATM)::n_l_d2      ! 回転部分のドナーリスト
logical*4,intent(out),dimension(MAX_SC)::h_rotate              ! 側鎖の回転フラグ
integer*4,intent(out),dimension(MAX_SC)::n_type_a              ! アクセプタ種別
integer*4,intent(out),dimension(MAX_SC)::n_type_d              ! ドナー種別
real*8,intent(out),dimension(MAX_ATOM,3)::h_vecp               ! 結合対象原子との距離
```

戻り値：

なし

機能：

蛋白質側鎖に関する情報を設定する。

13.sievgene

ファイル名：sievgene.f90

sievgene メインプログラム

13.1.sievgene

呼び出し形式：
  program sievgene
引数：
　なし
戻り値：
　なし
機能：
  sievgene のスクリーニング処理の全体制御を行う。

14.PBGrid_Method

ファイル名：PBGrid_Method.f90

Posisson-Boltzman 方程式を用いた Grid ポテンシャル生成のメソッドを定義する。

14.1. Calc_PBGridMain

呼び出し形式：

```
subroutine Calc_PBGridMain(numatom,npoint,co,charg,hank,point,              &
                           PBGridInfo, SCB)
```

引数：

```
integer*4,intent(in)::numatom                       ! 原子数
integer*4,intent(in)::npoint                        ! ポケット点数
real*4,intent(in),dimension(MAX_ATOM,3)::co         ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::charg        ! 電荷
real*4,intent(in),dimension(MAX_ATOM)::hank         ! 原子半径
real*4,intent(in),dimension(MAX_ATOM,3)::point      ! ポケット店座標
type(PBGridInfo_t),intent(in)::PBGridInfo           ! PBGrid 情報
type(SCB_t),intent(in)::SCB                         ! SCB 情報
```

戻り値：

なし

機能：

Posisson-Boltzman 方程式を用いた Grid ポテンシャル生成の主制御を行う。
PBGridPotential の入出力指定がある場合は、ファイルへの入出力を行う。

14.2. Set_PBGridPot

呼び出し形式：

```
subroutine Set_PBGridPot(numatom,npoint,co,charg,hank,point,PBGridInfo)
```

引数：

```
integer*4,intent(in)::numatom                       ! 原子数
integer*4,intent(in)::npoint                        ! ポケット点数
real*4,intent(in),dimension(MAX_ATOM,3)::co         ! 原子座標
real*4,intent(in),dimension(MAX_ATOM)::charg        ! 電荷
real*4,intent(in),dimension(MAX_ATOM)::hank         ! 原子半径
real*4,intent(in),dimension(MAX_ATOM,3)::point      ! ポケット店座標
type(PBGridInfo_t),intent(in)::PBGridInfo           ! PBGrid 情報
```

戻り値：

なし

機能：

メッシュを生成し、各メッシュ点について分子の内部、外部、境界であるかの
判定を行う。

14.3. Generate_PBGrid

呼び出し形式：
```
subroutine Generate_PBGrid(scb)
```
引数：
```
type(SCB_t),intent(in)::scb      ! SCB 情報
```
戻り値：
なし
機能：
静電場の計算を SOR 法（Successive Over-Relaxation method）で行う。

14.4. Calc_ElePotByPBGrid

呼び出し形式：
```
subroutine Generate_PBGrid(scb)
```
引数：
```
integer*4,intent(in)::ich
real*8,intent(in),dimension(MAX_SIZE**3,3)::XYZ_T
real*8,intent(out),dimension(MAX_SIZE**3)::result
```
戻り値：
なし
機能：
静電ポテンシャルを PB の Grid から求める。

15. AccessibleSurface_Method

ファイル名：AccessibleSurface_Method.f90

Richmond の方法による AccessibleSurfaceArea のメソッドを定義する。

15.1. Allocate_AccessibleSurfaceData

呼び出し形式：subroutine Allocate_AccessibleSurfaceData (ligandNum, proteinNum)
引数：

```
integer,intent(in)::ligandNum        ! リガンド原子数
integer,intent(in)::proteinNum       ! 蛋白原子数
```

戻り値：
　なし
機能：
　ASA 法で使用する動的な配列を確保する。
　すでに確保されている場合には、この領域を破棄して再確保する。

1) 各原子の ASA 半径　　：radius
2) 各原子の係数　　：coeff
3) ASA 向け係数(保存用)　　：coeffSA
4) GB 向け係数(保存用)　　：coeffGB
5) 接触原子リスト　　：atomIDTable
6) 接触原子数　　：surfaceNum
7) 非水素原子リスト　　：notHydAtomID

15.2. Set_AccessibleSurfaceData

呼び出し形式：subroutine Set_AccessibleSurfaceData
　　(ligandType, ligandAtomNo, ligandNum,&
　　proteinType, proteinAtomNo, proteinNum)
引数：

```
integer,intent(in),dimension(*)::ligandType    ! リガンド原子タイプ
integer,intent(in),dimension(*)::ligandAtomNo  ! リガンド原子番号
integer,intent(in)::ligandNum                  ! リガンド数
integer,intent(in),dimension(*)::proteinType   ! 蛋白原子タイプ
integer,intent(in),dimension(*)::proteinAtomNo ! 蛋白原子番号
integer,intent(in)::proteinNum                 ! 蛋白原子数
```

戻り値：
　なし
機能：
　次の ASA の原子データ情報を設定する。また、非水素原子(原子番号≠1)の個数をカウントし、ASA 情報に設定する。

1) 各原子の ASA 半径　　：radius
2) ASA 向け係数(保存用)　　：coeffSA
3) GB 向け係数(保存用)　　：coeffGB
4) 非水素原子リスト　　：notHydAtomID
5) リガンド原子先頭(非水素原子)　：ligandTop
6) リガンド原子最後(非水素原子)　：ligandLast
7) 蛋白原子先頭(非水素原子)　　：proteinTop
8) 蛋白原子最後(非水素原子)　　：proteinLast

15.3. Set_AccessibleSurfaceParameter

呼び出し形式：subroutine Set_AccessibleSurfaceParameter(option, asa)
引数：

```
type(Option_t),intent(in)::option              ! オプション情報
type(AccessibleSurface_t),intent(in)::asa      ! ASA 情報
```

戻り値：
　なし
機能：
　次の ASA の計算パラメータを設定する。

1) 各原子の ASA 半径　　：radius
2) 各原子の係数　　：coeff
3) プローブ半径　　：asaProbe
4) ASA のカットオフ長　　：asaCutoff
5) ASA の係数　　：asaWeight

15.4. Update_AccessibleTable

呼び出し形式：subroutine Update_AccessibleTable (cord, top, last)
引数：

```
real*4,intent(in),dimension(3,MAX_ATOM)::cord  ! 全体の原子座標
integer,intent(in)::top                        ! 計算対象の非水素原子先頭
integer,intent(in)::last                       ! 計算対象の非水素原子最後
```

戻り値：
　なし
機能：
　入力した原子群に対し、次の条件を満たす原子を相互作用テーブル(atomIDTable)に登録し、個数を surfaceNum に登録する。

原子間距離 <= 原子 1 のプローブ半径 + 原子 2 のプローブ半径 + カットオフ距離

15.5. Calc_AccessibleSurfaceEnergy

名称：Calc_AccessibleSurfaceEnergy
呼び出し形式：function Calc_AccessibleSurfaceEnergy (cord, grad, init, last)
　　result (energy)
引数：

```
integer,intent(in)::init                       ! 計算対象の非水素原子先頭
integer,intent(in)::last                       ! 計算対象の非水素原子最後
real*4,intent(in),dimension(3,MAX_ATOM)::cord  ! 全体の原子座標
real*4,intent(out),dimension(3,MAX_ATOM)::grad ! 全体の原子への力
```

戻り値：

```
real*8::energy                                 ! ポテンシャルエネルギー
```

機能：
　指定した原子に対する ASA 法のエネルギー、力の計算を行う。

15.6. Calc_LigandASAEnergy

名称：Calc_LigandASAEnergy
呼び出し形式：function Calc_LigandASAEnergy(ligandCord, ligandNum,
proteinCord, proteinNum)
result (energy)
引数：

```
  real*4,dimension(MAX_ATOM,3),intent(in)::ligandCord      ! リガンド原子座標
  integer,intent(in)::ligandNum                            ! リガンド原子数
  real*4,dimension(MAX_ATOM,3),intent(in)::proteinCord     ! 蛋白原子座標
  integer,intent(in)::proteinNum                           ! 蛋白原子数
```

戻り値：

```
    real*8::energy                                         ! ポテンシャルエネルギー
```

機能：
リガンド原子のみ存在する場合の ASA 法のエネルギー計算を行う。

15.7. Calc_ProteinASAEnergy

名称：Calc_ProteinASAEnergy
呼び出し形式：function Calc_ProteinASAEnergy(ligandCord, ligandNum,
proteinCord, proteinNum)
result (energy)
引数：

```
  real*4,dimension(MAX_ATOM,3),intent(in)::ligandCord      ! リガンド原子座標
  integer,intent(in)::ligandNum                            ! リガンド原子数
  real*4,dimension(MAX_ATOM,3),intent(in)::proteinCord     ! 蛋白原子座標
  integer,intent(in)::proteinNum                           ! 蛋白原子数
```

戻り値：

```
    real*8::energy                                         ! ポテンシャルエネルギー
```

機能：
蛋白原子のみ存在する場合の ASA 法のエネルギー計算を行う。

15.8. Calc_TotalASAEnergy

名称：Calc_TotalASAEnergy
呼び出し形式：function Calc_TotalASAEnergy(ligandCord, ligandNum,
proteinCord, proteinNum)
result (energy)
引数：

```
  real*4,dimension(MAX_ATOM,3),intent(in)::ligandCord      ! リガンド原子座標
  integer,intent(in)::ligandNum                            ! リガンド原子数
  real*4,dimension(MAX_ATOM,3),intent(in)::proteinCord     ! 蛋白原子座標
  integer,intent(in)::proteinNum                           ! 蛋白原子数
```

戻り値：

```
    real*8::energy                                         ! ポテンシャルエネルギー
```

機能：
系全体の ASA 法のエネルギー計算を行う。

16. BSpline_Method

ファイル名：BSpline_Method.f90

グリッドの B-Spline 補間のメソッドを定義する。

15.1. Set_BSplineIndex

呼び出し形式：subroutine Set_BSplineIndex(gridSize, splineOrder)
引数：

```
integer,intent(in)::gridSize        ! グリッドの１辺の分割数
integer,intent(in)::splineOrder     ! スプラインの次数
```

戻り値：
　なし
機能：
　B-Spline 補間において、グリッドの位置に対して各次数で参照するグリッドの位置を設定する。

15.2. Calc_InterporateWithEle

呼び出し形式：Calc_InterpolationWithEle
(kindSize, gridSize, grid,
x, y, z, charge, typeKind, asaPot, elePot,
grad)
引数：

```
integer,intent(in)::kindSize                    ! ポテンシャル種別数
integer,intent(in)::gridSize                    ! グリッドの１辺の分割数
real*4,intent(in),dimension(kindSize,gridSize,gridSize,gridSize)::grid
                                                ! ポテンシャルグリッド
real*4,intent(in)::x, y, z                      ! 対象原子の座標
real*4,intent(in)::charge                       ! 対象原子の電荷
integer,intent(in)::typeKind                    ! 計算対象原子のポテンシャルタイプ
real*4,intent(out)::asaPot                      ! 計算対象のポテンシャル
real*4,intent(out)::elePot                      ! 静電ポテンシャル
real*4,intent(out),dimension(3)::grad           ! ポテンシャルの微分値
```

戻り値：
　なし
機能：
　当該座標に存在する原子に対して、ポテンシャルグリッドから寄与される計算対象と静電のポテンシャル、およびポテンシャルの１次微分を B-Spline 法で計算する。

　B-Spline では当該点を含むグリッドの 8 点のポテンシャル、およびこの周囲のグリッドのポテンシャルに対し、次数に応じた係数をかけて加算することでスムージングを行う。

15.3. Calc_Interpolation

呼び出し形式：Calc_Interpolation
(kindSize, gridSize, grid,
x, y, z, typeKind, pot, grad)

引数：

```
integer,intent(in)::kindSize                    ! ポテンシャル種別数
integer,intent(in)::gridSize                    ! グリッドの１辺の分割数
real*4,intent(in),dimension(kindSize,gridSize,gridSize,gridSize)::grid
                                                ! ポテンシャルグリッド
real*4,intent(in)::x, y, z                      ! 対象原子の座標
integer,intent(in)::typeKind                    ! 計算対象原子のポテンシャルタイプ
real*4,intent(out)::pot                         ! 計算対象のポテンシャル
real*4,intent(out),dimension(3)::grad           ! ポテンシャルの微分値
```

戻り値：
なし

機能：
当該座標に存在する原子に対して、ポテンシャルグリッドから寄与される計算対象のポテンシャル、およびポテンシャルの１次微分を B-Spline 法で計算する。

15.4. Get_BSplineCoefficient

呼び出し形式：subroutine Get_BSplineCoefficient(floatPart, order, array, darray)

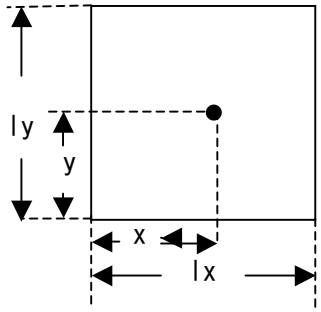
引数：

```
integer,intent(in)::order                       ! B-spline の次数
real*4,intent(in),dimension(3)::floatPart       ! 当該点のグリッド内相対位置
real*4,intent(out),dimension(1:3,order)::array  ! 各グリッドの係数
real*4,intent(out),dimension(1:3,order)::darray ! 各グリッドの１次微分係数
```

なし

機能：
当該点のグリッド内相対位置*1に対する、当該点に寄与する各グリッドの係数を求める。

*1)グリッド内相対位置：
当該座標を含むグリッドの辺の長さに対する、グリッドの原点と当該座標との距離の割合を示す。下図でグリッド内の原点に対する位置を(x,y)とし、グリッドの辺の長さを(lx,ly)とするとグリッド内相対位置は(x/lx, y/ly)となる。



グリッド

－以上－