XMVB 3.0

A Right Way To Do Valence Bond Calculations

Zhenhua Chen
Xiamen University

XMVB 3.0

Xiamen Valence Bond

An Ab Initio Non-orthogonal Valence Bond Program

Version 3.0

Lingchun Song, Zhenhua Chen, Fuming Ying, Jinshuai Song,

Xun Chen, Peifeng Su, Yirong Mo, Qianer Zhang, Wei Wu*

Center for Theoretical Chemistry,

State Key laboratory for Physical Chemistry of Solid Surfaces,

and Department of Chemistry Xiamen University, Xiamen Fujian 361005, CHINA

weiwu@xmu.edu.cn

History

- 1986–1998 pregnancy of the program

  Algorithms: SGA, PPD etc.
- 1999 Xiamen-99

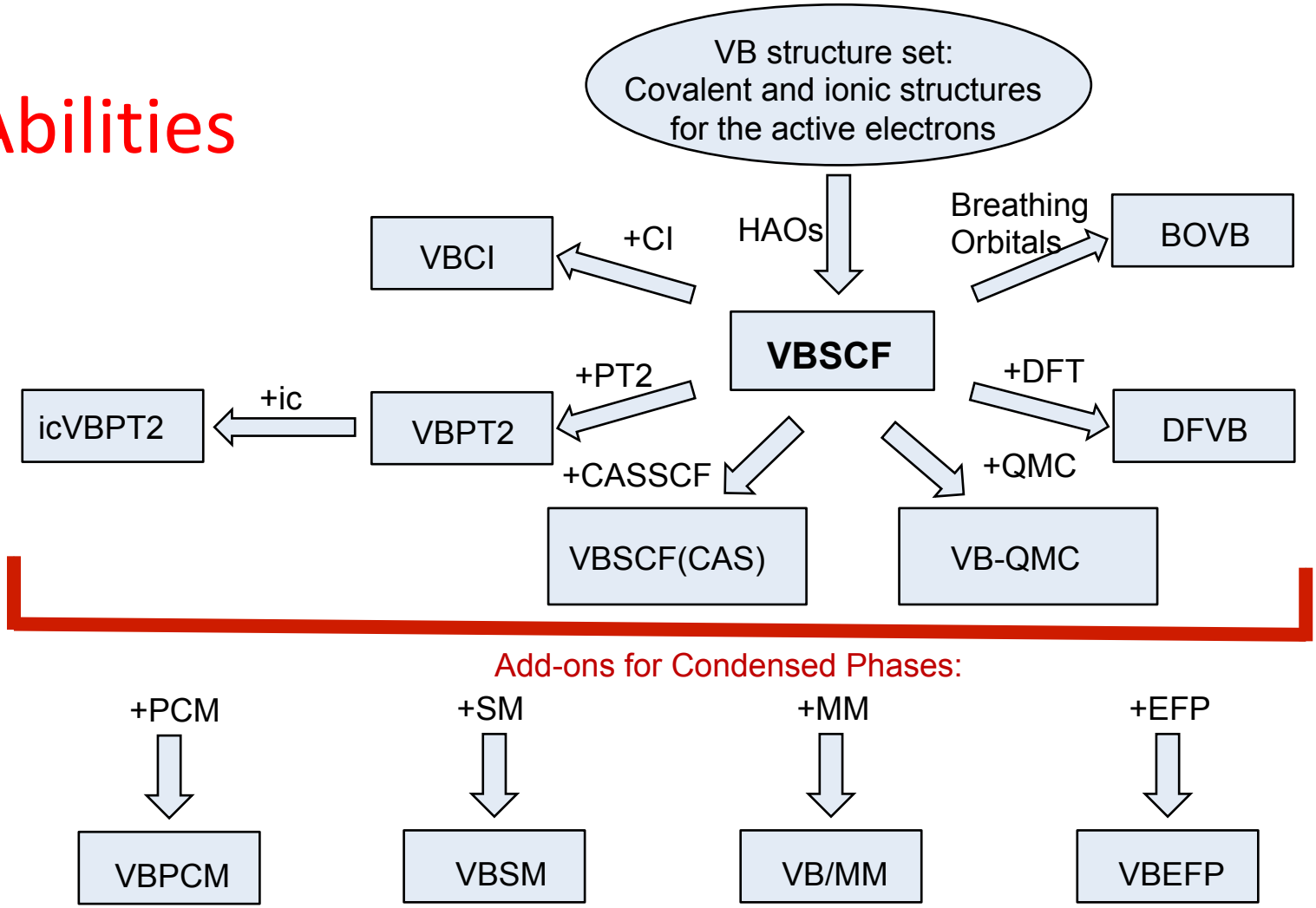
  VBSCF, BOVB, VBCI
- 2003 XMVB Version 1.0 was released

  VBPT2, DFVB, VBPCM,

  Parallelization, Modularization, RDM
- 2012 XMVB Version 2.0 was released

  RDM, AFCG, icVBPT2, CD Int, SN
- In 2017, XMVB Version 3.0 was released

Abilities

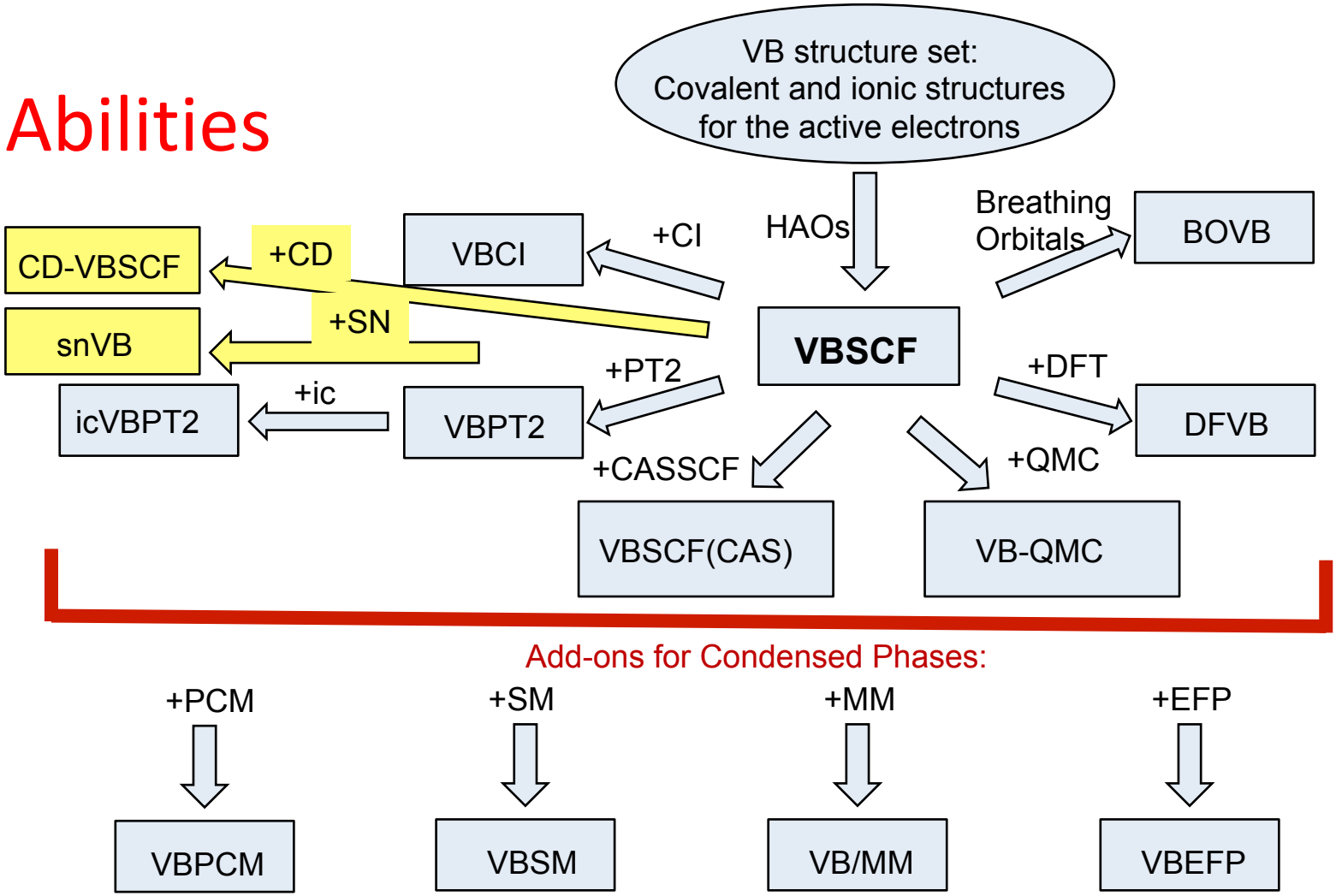


All classical VB methods are available!

What's New in Version 3.0?

- Integrals can be calculated directly.
- More basis sets and elements are supported.
- Cholesky decomposition for ERI is available.
- Tensor transformation is implemented for RDM-based approach.
- Seniority number truncated VBSCF method

Abilities



All classical VB methods are available!

Useful documents

XMVB manual

http://xmvb.org/docs/xmvb-3.0-manual.pdf

Online Tutorials

http://xmvb.org/tutorials.html

Distributions of XMVB 3.0

Stand-alone distribution

A stand-alone program can be used to do most VB calculations.

Module distribution

A module embedded in GAMESS-US can be used to accomplish VB calculations in condense phase

How to Run An XMVB Job?

For Stand-alone distribution

For jobs use the same integrals.

1. Compose "job.inp"
2. Run "preint job.inp"
3. Compose "job.xmi"
4. Run "XMVB job.xmi"

To get 1/2-electron integrals
x1e.int, x2e.int

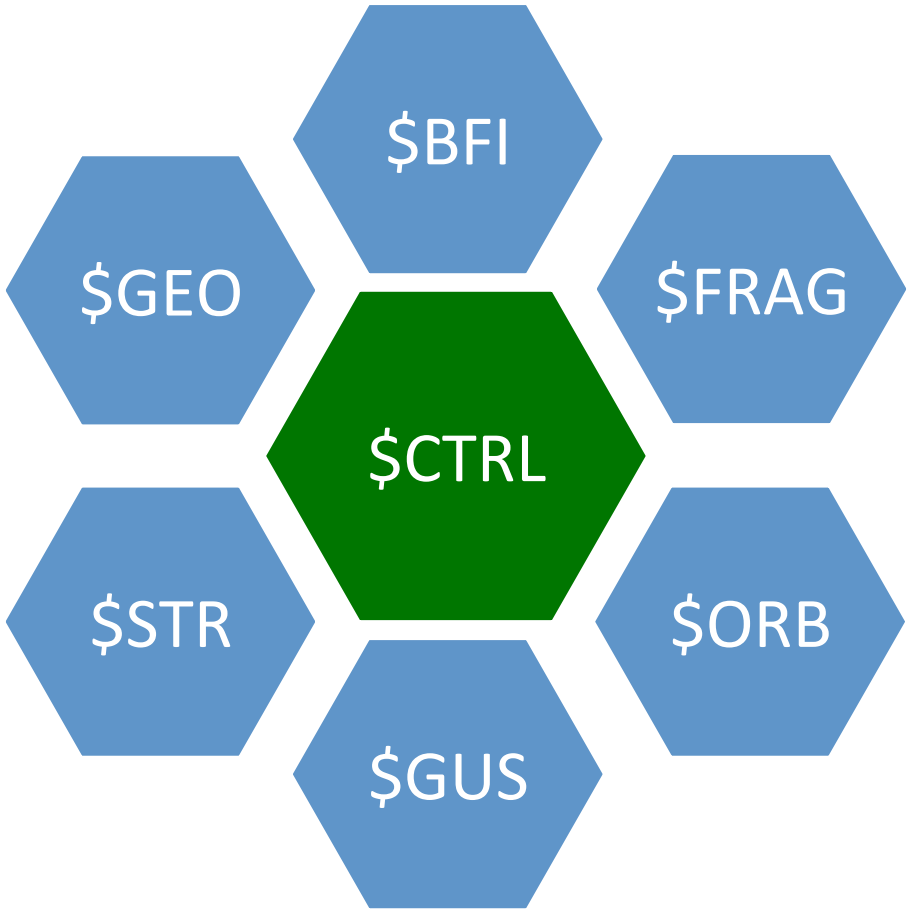
do VB calculation
to get XMVB output files:
job.xmo, job.orb, job.den, job.xdat

1. job.xmi += job.inp
2. Run "XMVB job.xmi"

For jobs have large number of basis functions.

Part 2. How to compose input files?

Main structure of XMI file



Optional

Essential

How to write INT File

```
H2 cc-pvdz
0 1
H 0.000000  0.000000  0.000000
H 0.000000  0.000000  0.740000
```

← Job name; basis set

← Charge; multiplicity

← Elements and Cartesian coordinates

preint H2.inp

job.xmi += job.inp

$GEO
……
$END

Example 1. H_2 molecule

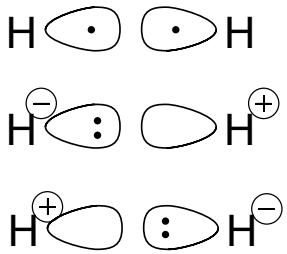
```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$str
1 2
1 1
2 2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
```

Flexible format

Example 1. H_2 molecule

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```



How to compose an input file?

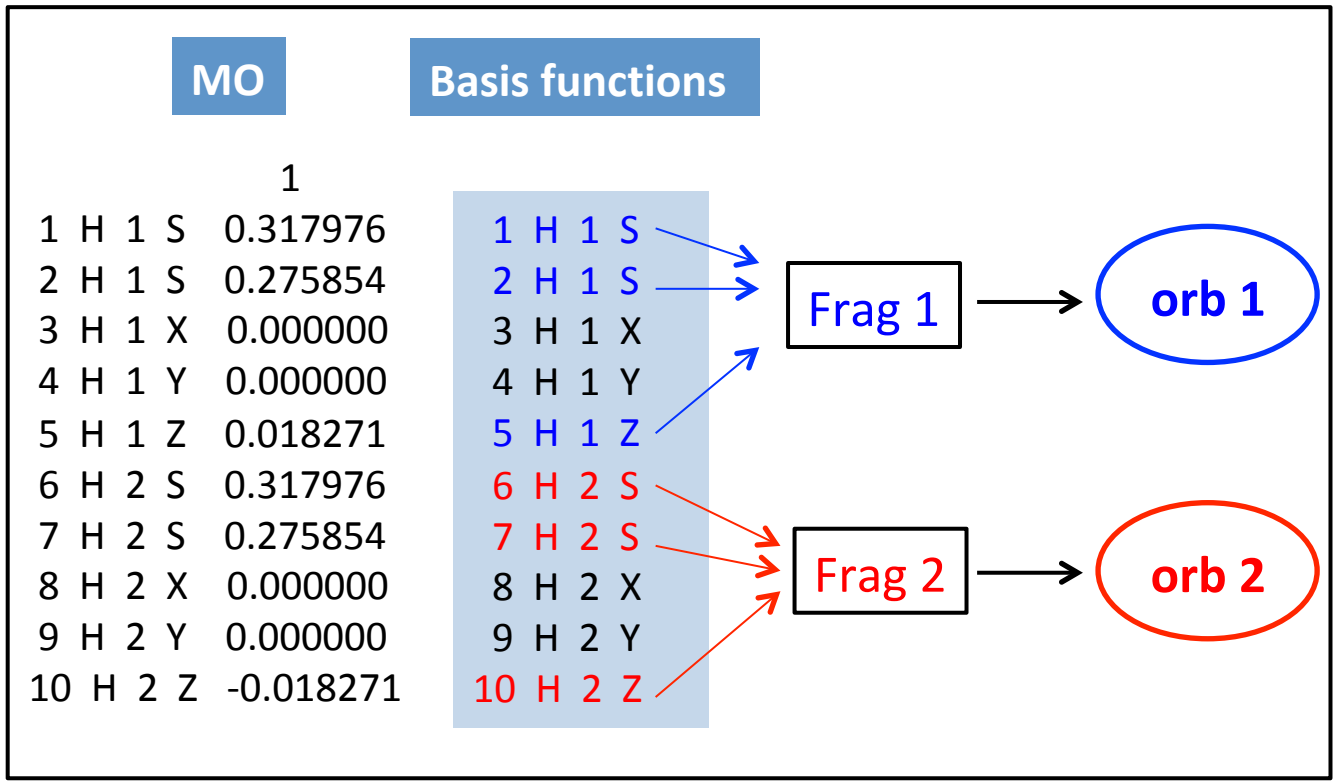
- Obtain molecular geometry
- Divide molecule into fragments
- Specify orbitals on each fragment
- Distribute electrons on the orbitals to give structures.

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```

Fragments are a series of basis function sets.

frgtyp=atom or frgtyp=sao, fragments are defined with **s**ymmetrized **a**tomic **o**rbitals.

orbtyp=hao, the hybrid atomic orbitals are used.



Fragments are a series of basis function sets.

frgtyp=sao, fragments are defined with **s**ymmetrized **a**tomic **o**rbitals.

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```

$frag: description for fragments
Line 1: the number of atoms in each fragment.
Line 2 specifies which atoms and their basis functions:
atom 1, and the s and pz types basis functions.
Line 3 : the basis functions of s and pz types in atom 2.

$orb: description for VB orbitals
Line 1: the number of fragments in each orbital.
Line 2 specifies orbital 1 is localized on which fragments:
fragment 1.
Line 3 denotes orbital 2 is localized on fragment 2.

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```

Fragments are a series of basis function sets.

frgtyp=sao, fragments are defined with **s**ymmetrized **a**tomic **o**rbitals.

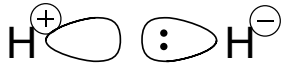
$frag: description for fragments
Line 1: the number of atoms in each fragment.
Line 2 : the basis functions of s and pz types in atom 1.
Line 3 : the basis functions of s and pz types in atom 2.

$orb: description for VB orbitals
Line 1: the number of fragments in each orbital.
Line 2: orbital 1 is localized on fragment 1.
Line 3: orbital 2 is localized on fragment 2.

covalent

ionic

ionic



nstr =3, the number of structures
In $str, a typical structure is written as:

i i j j k k … m n

i j k: doubly occupied inactive orbitals.
m n: variable occupied active orbitals.

Orbital optimization algorithms for VBSCF and BOVB: iscf options.

```
$ctrl
frgtyp=sao orbtyp=hao
nstr=3  iscf=5 nao=2 nae=2
$end
$frag
1 1
spz 1
spz 2
$end
$orb
1 1
1
2
$end
$str
1 2
1 1
2 2
$end
```

When iscf=5, the two keywords are necessary.
nao: number of active orbitals
nae: number of active electrons

iscf	Gradient	Performance
1	Numerical	Slow
2	Analytical	Fast, but some time not stable
3	Numerical	Slow
4	Numerical	Very slow
5	**Analytical**	**Very Fast, but not for BOVB**
6	Full Hessian	Slow, quadratically converged

Summary (1)

- $GEO Obtain molecular geometry
- $FRA Divide molecule into fragments
- $ORB Specify orbitals on each fragment
- $STR Distribute electrons on the orbitals to give structures
- $CTRL Assign an algorithm (i.e. iscf=5) to do the orbital optimization in valence bond calculation.

Run the job by:

XMVB job.xmi

Summary (1)

- $GEO Obtain molecular geometry
- $FRA Divide molecule into fragments
- $ORB Specify orbitals on each fragment
- $STR Distribute electrons on the orbitals to give structures
- $CTRL Assign an algorithm (i.e. iscf=5) to do the orbital optimization in valence bond calculation.

Run the job by:

XMVB job.xmi

Make modifications until XMVB terminates graciously.

Example 2. HF molecule

hf.inp

```
HF 6-31G
 0 1
 H   1.0   0.000000  0.000000  0.000000
 F   9.0   0.000000  0.000000  0.917000
```

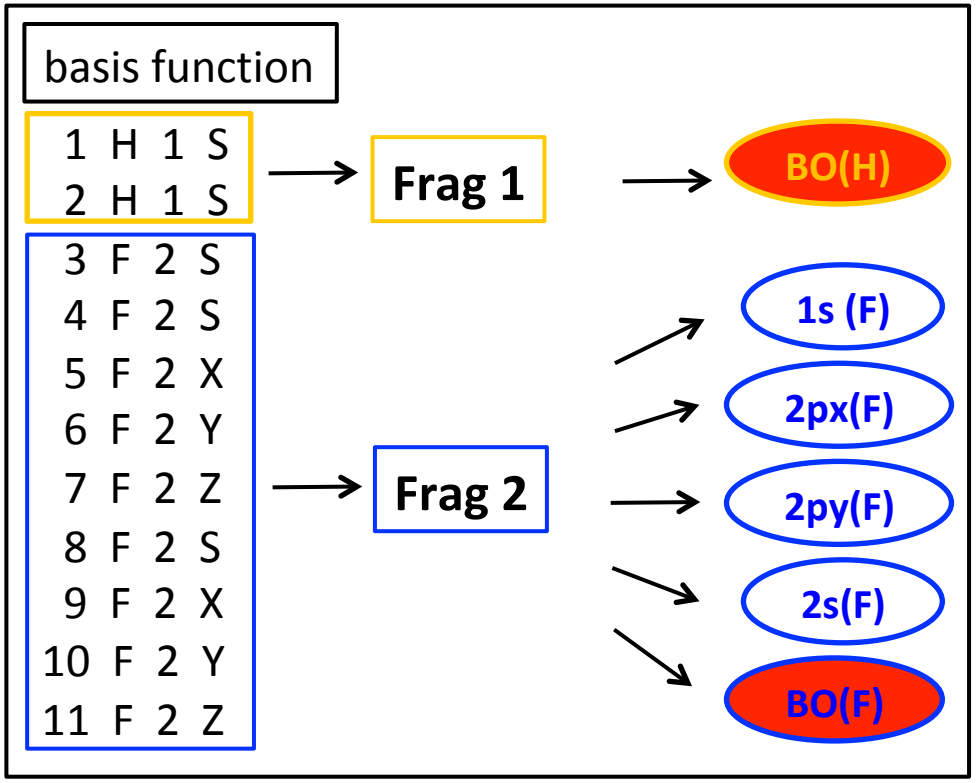
HF molecule, 3 structures

hf.xmi

```
$ctrl
frgtyp=atom orbtyp=hao str=full
iscf=5 nae=2 nao=2
iprint=3 guess=mo
$end
$orb
1 1 1 1 1 1
2
2
2
2
2
1
$end
```

frgtyp=atom
The fragments of system are defined with atoms. By Default, each atom forms one fragment.

orbtyp=hao
Line 1 gives numbers of fragments for six orbitals. The first 5 orbitals are localized on the second fragment of F atom. The last orbital is localized on the first fragment of H atom.



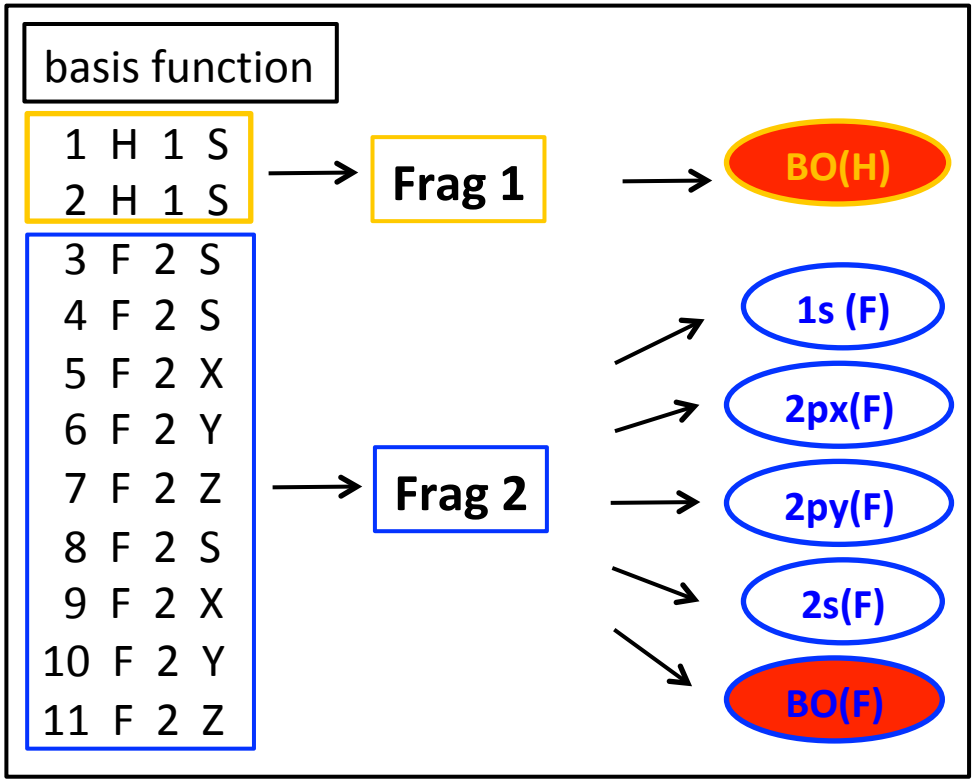
$str?

HF molecule, 3 structures hf.xmi

```
$ctrl
frgtyp=atom orbtyp=hao str=full
iscf=5 nae=2 nao=2
iprint=3 guess=mo
$end
$orb
1 1 1 1 1 1
2
2
2
2
2
1
$end
```

frgtyp=atom
The fragments of system are defined with atoms. By Default, each atom forms one fragment.

orbtyp=hao
Line 1 gives numbers of fragments for six orbitals.
The first 5 orbitals are localized on the second fragment of F atom.
The last orbital is localized on the first fragment of H atom.



$str?

If keyword "str" is used, XMVB will automatically generate structures.
str = full => all VB structures
cov => only covalent
ion => only ionic structures
$STR section can be absent.

HF molecule, 3 structures

hf.xmi

```
$ctrl
frgtyp=atom orbtyp=hao str=full
iscf=5 nae=2 nao=2
iprint=3 guess=mo
$end
$orb
1 1 1 1 1 1
2
2
2
2
2
1
$end
$gus
1 1
2 2
3 4
4 5
5 3
6 3
$end
```

Iprint=3, full print-out message.

XMVB hf.xmi

guess=mo: Molecular Orbitals will be used as the initial guess for VB orbitals.

VB orb		MO
1	1s orbital of F atom	1
2	2s orbital of F atom	2
3	px pair of F atom	4
4	py pair of F atom	5
5	H-F bonding orbital on F atom	3
6	H-F bonding orbital on H atom	3

				1	2	3	4	5
				-26.2757	-1.5897	-0.7391	-0.631	-0.631
				A	A	A	A	A
1	H	1	S	0.000301	0.132423	-0.293056	0.000000	0.000000
2	H	1	S	0.001771	-0.013969	-0.107756	0.000000	0.000000
3	F	2	S	0.995513	-0.230516	-0.061755	0.000000	0.000000
4	F	2	S	0.021368	0.497220	0.137028	0.000000	0.000000
5	F	2	X	0.000000	0.000000	0.000000	0.654168	0.091145
6	F	2	Y	0.000000	0.000000	0.000000	-0.091145	0.654168
7	F	2	Z	-0.001488	-0.084307	0.548328	0.000000	0.000000
8	F	2	S	-0.005826	0.529428	0.244193	0.000000	0.000000
9	F	2	X	0.000000	0.000000	0.000000	0.485512	0.067646
10	F	2	Y	0.000000	0.000000	0.000000	-0.067646	0.485512
11	F	2	Z	0.001400	-0.047087	0.349640	0.000000	0.000000

Example 2b. L-BOVB calculation for HF

```
HF molecule, 3 structures
$ctrl
orbtyp=hao frgtyp=atom str=full
 bovb iscf=2 guess=read
 nae=2 nao=2 iprint=3
$end
$orb
1 1 1 1 1 1
2
2
2
2
2
1
$end
```

For BOVB calculations,
iscf=2 is most efficient;
iscf = 4, 5, 6 are not available.

Using VBSCF orbitals as guess orbitals for BOVB calculation by:

cp hf.orb hf-bovb.gus

XMVB hf-bovb.xmi

Summary (2)

1. Generate structures by *str.*
2. Set the fragment by *FrgTyp=atom*.
3. Set orbital guess by '*guess=mo*' and edit $*gus*.
4. Use VBSCF orbital as guess for BOVB calculation, and set '*guess=read*'.

Example 3, benzene molecule

ben.inp

```
C6H6 6-31G*
 0  1
 C              0.6993466831   1.2113039873   0.0000000000
 C             -0.6993466831   1.2113039873   0.0000000000
 C             -1.3986933663   0.0000000000   0.0000000000
 C             -0.6993466831  -1.2113039873   0.0000000000
 C              0.6993466831  -1.2113039873   0.0000000000
 C              1.3986933663   0.0000000000   0.0000000000
 H              1.2427030147   2.1524247602   0.0000000000
 H             -1.2427030147   2.1524247602   0.0000000000
 H             -2.4854060294   0.0000000000   0.0000000000
 H             -1.2427030147  -2.1524247602   0.0000000000
 H              1.2427030147  -2.1524247602   0.0000000000
 H              2.4854060294   0.0000000000   0.0000000000
```

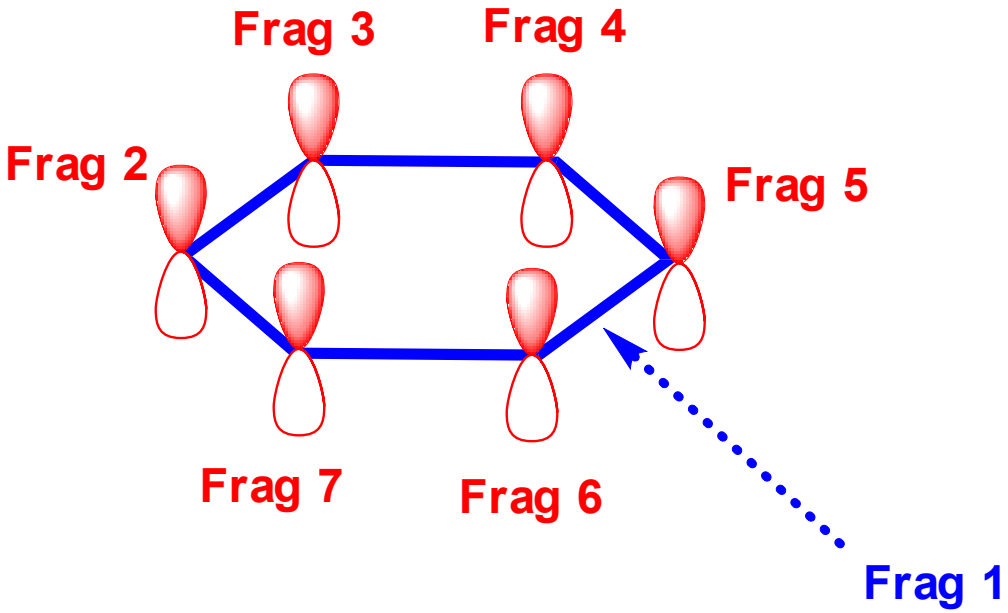
The benzene molecule locates on the XY plane.

Only the π bonds are explored in VB manner.

The σ electrons/orbitals can be treated in MO way.

Only one fragment is assigned for σ-part.

For π -part, pz-type basis functions on each atom forms one fragment.



ben.xmi (part1)

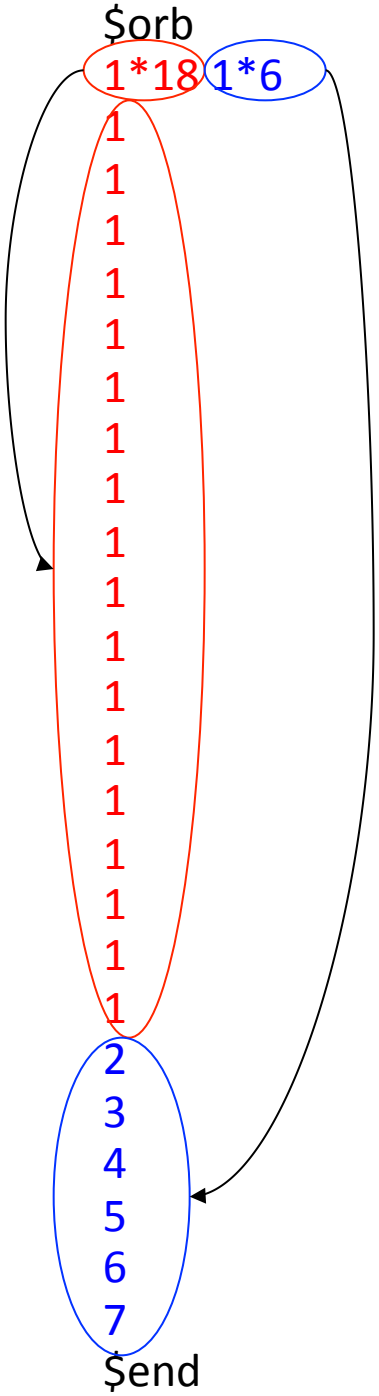
```
benzene
$ctrl
frgtyp=sao orbtyp=hao str=full
iscf=5 nao=6 nae=6
iprint=3 guess=auto
$end
$frag
12 1*6
spxydxxyyzzxy 1-12
pzdxzyz 1
pzdxzyz 2
pzdxzyz 3
pzdxzyz 4
pzdxzyz 5
pzdxzyz 6
$end
```

nao=6, the 6 π-orbitals are selected as active orbitals.
nae=6, the number of active VB electrons
str = full => generate all VB structures
cov => only covalence
Ion => only ionic structures

frgtyp=sao, fragments are defined with symmetrized atomic orbitals.

All σ-type basis function of all atoms (1-12) form the first fragment. Here, "pxy" is a short hand notation of px and py; and "dxxyyzzxy" means Dxx, Dyy, Dzz, Dxy.

pz-type basis functions (pz, dxz, dyz) on each atom forms one fragment.



ben.xmi (part2)

There are 24 VB orbitals, the first 18 orbitals are localized on frag 1; the remain 6 orbitals are localized on frag 2 to frag 7 respectively.

Part 3. How to get information from output files?

Information of XMO file

- Input message → Basic Input, Integrals, Guess etc
- SCF iterative procedure (final energy) → Iteration ** E = **.** DE = ** / VBSCF converged in ** iterations / **Total Energy: ****
- S_{ij} and H_{ij} Matrices
- Information of VB wavefunction → **Coefficients and Weights of structures** / Optimized orbitals
- Properties → Bond order, atomic charge, dipole moments

Optional

Default

hf.xmo

Total Energy: -100.03252880
First Excited: -98.677762
The Last Change in Energy: -0.000000
Number of Iteration: 32

SCF procedures (final energy)

****** MATRIX OF OVERLAP ******

	1	2	3
1	1.000000	-0.694887	-0.694887
2	-0.694887	1.000000	0.318276
3	-0.694887	0.318276	1.000000

****** MATRIX OF HAMILTONIAN ******

	1	2	3
1	-2.554856	2.041815	1.827629
2	2.041815	-2.247238	-1.090411
3	1.827629	-1.090411	-1.685275

S_{ij} and H_{ij} matrices

****** WEIGHTS OF STRUCTURES ******

1	0.64395	******	1:4	5	6
2	0.30554	******	1:4	5	5
3	0.05051	******	1:4	6	6

Weights of VB structures

	1	2	3	4	5	6
1	0.000000	0.000000	0.000000	0.000000	0.000000	-0.835424
2	0.000000	0.000000	0.000000	0.000000	0.000000	-0.227552
3	1.012748	-0.084340	0.084340	-0.084340	0.049531	0.000000
4	-0.025976	0.300588	-0.300588	0.300588	-0.165593	0.000000
5	0.000000	-0.020440	-0.477373	-0.456933	0.000000	0.000000
6	0.000000	0.539422	0.252010	-0.287413	0.000000	0.000000
7	-0.010873	0.069765	-0.069765	0.069765	0.701583	0.000000
8	-0.054032	0.316476	-0.316475	0.316476	-0.042957	0.000000
9	0.000000	-0.015134	-0.353442	-0.338309	0.000000	0.000000
10	0.000000	0.399383	0.186585	-0.212797	0.000000	0.000000
11	-0.006949	0.056493	-0.056493	0.056493	0.419452	0.000000

Optimized orbitals

****** POPULATION AND CHARGE ******

ATOM	MULL.POP.	CHARGE	LOW.POP.	CHARGE
1 H	0.744964	0.255036	0.855733	0.144267
2 F	9.255036	-0.255036	9.144267	-0.144267

XMVB atomic population analysis

****** BOND ORDER ******

ATOM 1	ATOM 2	DIST	BOND ORDER
1 H	2 F	0.900	0.899

Availability of XMVB package

- To performs *ab initio* valence bond calculations
- XMVB version 3.0 is free.
- Send signed license agreement to:
- Professor Wei Wu
  Department of Chemistry
  Xiamen University,
  Xiamen, Fujian 361005
  P. R. China
  Tel: 86-592-2182825 Fax: 86-592-2184708
  E-mail: weiwu@xmu.edu.cn

Thank you for your attentions!