

VB2000

An *ab initio* Valence Bond Program based
on the Generalized Product Function
Method and the Algebrant Algorithm

Version 2.6

Jiabo Li ¹, Brian Duke ², and Roy McWeeny ³

¹ *SciNet Technologies, 9943 Fieldthorn St., San Diego CA 92127, USA*

² *Monash Institute of Pharmaceutical Sciences, Monash University
381 Royal Pde, Parkville, Victoria, 3052, Australia*

³ *Department of Chemistry, University of Pisa, 56100 Pisa, ITALY*

Date code finalized: January, 2012

Date of most recent change in manual: January, 2012

*Copyright © 2000-2012 by Jiabo Li, Brian Duke and Roy McWeeny
All Rights Reserved.*

Citation of VB2000 and the Theory

Publications with results obtained from using VB2000 should cite the software and the theory in the following way.

Reference for the software:

Jiabo Li, Brian Duke, and Roy McWeeny, VB2000 Version 2.5, SciNet Technologies, San Diego, CA, 2010. URL: <http://www.scinetec.com>

References for the theory:

Jiabo Li, and Roy McWeeny, "VB2000: Pushing Valence Bond Theory to New Limits", *Int. J. Quantum Chem.*, 89 (2002) 208-216.

Jiabo Li, Brian J. Duke, Thomas M. Klapötke, and Roy McWeeny, "Spin Density of Spin-Free Valence Bond Wavefunction and Its Implementation in VB2000", *J. Theor. Comput. Chem.*, 7 (2008) 853-867.

Contents

1 Introduction	8
1.1 What is VB2000?	8
1.2 What is new in VB2000 version 2.6	10
1.3 What is new in VB2000 version 2.5	10
1.4 What is new in VB2000 version 2.1	11
1.5 What is new in VB2000 version 2.0	11
1.6 What is new in VB2000 version 1.8(R2)?	12
1.7 What is new in VB2000 version 1.8?	12
2 Installation Guide	15
2.1 Download VB2000 packages	15
2.2 Install VB2000 as a stand alone program on a UNIX-like platform	16
2.3 Install WinVB2000 on Windows platform	16
2.4 Install WinGAMESS (with VB2000 built-in)	18
2.4.1 Run WinGAMESS installer	18
2.4.2 Run WinGAMESS from WinVB2000 interface	18
2.5 Install GAMESS-VB2000	19
2.5.1 Step by step instructions	19
2.5.2 Run GAMESS-VB	20
2.6 Install Gaussian-VB2000	20
2.7 Systems supported	21
2.7.1 Linux	21
2.7.2 Cygwin	22
2.7.3 Silicon Graphics IRIX64	22
2.7.4 Windows platform	22
2.7.5 Max OS X	22
2.7.6 The future	22
2.8 Running tests on UNIX-like platforms	23
3 Capabilities of VB2000	24
3.1 Hartree-Fock molecular orbital theory	24
3.2 Pipek-Mezey localization method	24
3.3 Modern <i>ab initio</i> Valence Bond method	24
3.4 Group Function Theory and its combination with VB method	25
3.5 Energy profile of chemical reactions	25
3.6 Structure weights	26
3.7 Spin density	27
3.8 Visualization	27
3.9 Strict localization and enhanced localization of VB orbitals	27
3.10 Extended functionality with GAMESS-VB2000 and Gaussian-VB2000	28

4 Quick Start Tutorial	29
4.1 Simple VB calculation of CH ₄	29
4.2 How does it work?	30
4.2.1 Hartree-Fock calculation	30
4.2.2 Pipek-Mezey MO localization	30
4.2.3 LMO group assignment	30
4.2.4 Initial VB orbitals	31
4.3 Two-group VB wave function of CH ₃ -CH ₂ OH	31
4.4 GVB calculation of ethane	32
4.5 CASVB(4,4) calculation of H ₂ O	32
4.6 Run VB2000 in GAMESS	33
4.7 Tips and tricks	34
5 Terminology	35
5.1 Electron groups	35
5.2 Group function	35
5.3 System wave function function	35
5.4 Macro iteration	35
5.5 Rigid rotation of a system wave function	35
5.6 VB, VBSCF, SC, BOVB, CASVB and CASSCF	36
5.7 Group Function Theory (GFT) vs. ORMAS	37
6 Input Description	38
6.1 General consideration	38
6.2 Basic controls and molecule specification	39
6.2.1 Method	39
HF	39
VB(m)	39
SC(m)	39
SC(m,n)	40
CASVB(m,n)	40
GPF(n _g)	40
Dot notation	40
6.2.2 Basis set	41
6.2.3 Options	42
PRINTALL	42
RESTART	42
CIONLY	42
NOROT	42
FROZEN	42
UNITS=BOHR	42

SPDEN.....	43
DIIS.....	43
SPHER.....	43
TEST.....	43
6.3 Group Function controls.....	44
6.3.1 \$GENCTL.....	44
6.3.2 \$GRPDIM.....	44
6.3.3 \$VBGA (\$VBGROUPASSIGNMENT).....	45
6.3.4 \$LMOGRPMODIFY.....	46
6.3.5 \$MACROITER.....	46
6.3.6 \$ECONV.....	47
6.3.7 \$ROTATION.....	47
6.3.8 \$NOTROT.....	47
6.3.9 \$FULLHESS.....	48
6.3.10 \$DAMPROT.....	48
6.4 VB orbital optimization controls.....	48
6.4.1 \$VBSCF.....	48
6.4.2 \$##VBSCF.....	49
6.4.3 \$##VBSTR.....	49
6.4.4 \$HESSCONST.....	50
6.4.5 \$BRILMASK.....	51
6.4.6 \$##LENHANCE.....	51
6.4.7 \$DPWEIGHT.....	53
6.4.8 \$CMAXCUT.....	53
6.4.9 \$NOLOCVBO.....	53
6.5 Initial VB orbital generation.....	53
6.5.1 \$##VBORB.....	54
6.5.2 General expression.....	54
6.5.3 \$##PIVBO.....	55
6.5.4 \$LMODISTORTION.....	55
6.5.5 \$WRITEGUESS.....	56
6.5.6 \$READGUESS.....	56
6.5.7 \$RESTARTFILE.....	57
6.5.8 \$RESTARTMAPPING.....	57
6.5.9 \$AOGROUP (original \$PIORB).....	60
6.6 Control for chemical reaction.....	61
6.6.1 \$REACTION.....	61
6.7 Advanced controls.....	61
6.7.1 \$##ROOT.....	61
6.7.2 \$##STRSYMM.....	62
6.7.3 \$MEMORY.....	62
6.7.4 \$VBOLIBGEN.....	63
6.7.5 Controls for canonicalization of LMOs.....	65
6.7.6 \$PRINTHS.....	65
6.8 Visualization of VB orbitals and molecules.....	66
6.8.1 \$MOLPLT.....	66

6.8.2 \$PLTORB.....	66
6.8.3 \$CUBE.....	67
6.8.4 \$GENGRID or \$GRID.....	68
6.8.5 \$XYZFILE.....	68
6.8.6 \$MOLEKEL.....	69
6.8.7 \$MOLDEN.....	69
6.8.8 Using the visualization directives in TEST runs.....	69
6.8.9 3D pictures of two VB orbitals of a O-H bond	69
7 Construction of Initial Wave Functions	71
7.1 Notations of localized molecular orbitals.....	71
7.1.1 Lone pair.....	71
7.1.2 Two-center bond LMO.....	73
7.1.3 Multi-center bond LMO.....	75
7.2 Notations of VB orbitals.....	75
7.2.1 σ VB orbital.....	75
7.2.2 π VB orbital.....	76
7.2.3 Multiple VBOs.....	77
7.3 Construction of initial VB orbitals.....	78
7.3.1 Initial VBOs by splitting LMOs.....	78
7.3.2 Initial VBOs by VBO libraries.....	78
7.3.3 Linear combination of standard VBOs.....	78
7.3.4 A method for construction of CASSCF space.....	79
8 Test Cases and Output Descriptions	80
8.1 VB(4) calculation of H ₂ O.....	80
8.2 CASVB(4,4) calculation of H ₂ O.....	90
8.3 CASVB(8,6) calculation of H ₂ O.....	92
8.4 VB(6) calculation of C ₆ H ₆	95
8.5 VB(6) calculation of the triple bond of C ₂ H ₂	97
8.6 VB(6) calculation of C ₂ H ₂ with three equivalent banana bonds.....	99
8.7 VB(10) calculation of C ₂ H ₂	100
8.8 VB(12) electron VB calculation of C ₂ H ₄	101
8.9 VB(14) electron VB calculation of C ₂ H ₆	102
8.10 Triplet state of trimethylmethene (TMM) and spin density.....	104
8.11 Singlet state of TMM.....	107
8.12 VB(8)•VB(8) calculation of ethanol.....	109
8.13 All-valence electron VB calculation of C ₆ H ₆	113
8.14 VB(8)•CASVB(6,6)•VB(8) calculation of Si ₄ H ₆	119
8.15 VB calculation of the transition state of [Cl---CH ₃ ---Cl] ⁻	121
8.16 Energy profile of a S _N 2 reaction.....	122

8.17 Visualization of VB orbitals of H ₂ O.....	125
8.18 Calculation with dynamic VB orbitals.....	127
8.19 VB calculation with strictly localized orbitals.....	129
8.20 VB calculation of structure weights with localization enhanced VB orbitals.....	132
 9 Acknowledgments	 137
 10 References	 139

1 Introduction

1.1 What is VB2000?

VB2000 is an *ab initio* electronic structure package for performing modern Valence Bond calculations based on a highly efficient VB algorithm—the so called [Algebrant Algorithm](#)[1-5]. It has a general implementation of the [Group Function \(GF\) Theory](#) [6-8], in which a large molecule is described in terms of its constituent parts of physically identifiable "electron groups". The modern VB method implemented in VB2000 produces CASSCF(N,N) quality wave functions with only a small fraction of CPU cost for a CASSCF calculation for $N > 10$. The combination of efficient VB methods and the general implementation GF theory in VB2000 makes it a very powerful tool with the investigation of the nature of chemical bonds[9-13].

The development of VB2000 has been motivated by two main considerations. First, there is a need to obtain high-precision electronic wave functions, capable of giving quantitative substance to the empirical and intuitive ideas distilled from more than two hundred years of experimental chemistry. There exist within molecules "structural units" such as chemical bonds and functional groups, often with highly individual properties, which are transferable from one environment to another with little change; the behavior of such units conforms, in many cases, to empirical "additivity rules" which have never received a convincing theoretical explanation. Second, although valence bond theory, as developed in the 'thirties and forties', was brilliantly successful in rationalizing such ideas, within the conceptual framework of quantum mechanics, it failed completely to provide rigorous quantitative predictions. The reason was simply that for a system of N electrons the computational cost of *ab initio VB* calculations tended to grow like N factorial.

With use of the Algebrant Algorithm, *ab initio* VB calculations on systems with up to about 14 electrons are now becoming 'routine' — even though $N!$ is then of the order 9×10^{10} . However, the majority of molecules of real chemical interest possess far more than 14 electrons. It therefore becomes mandatory to make some kind of conceptual 'separation' of a larger molecule into smaller and more tractable components. By using the [GF](#) approach for this purpose, VB2000 brings molecules with many groups within the scope of rigorous non-empirical investigation. Even though the current release has a limitation to a maximum of 16 electrons for each VB group, the full VB calculation of up to a 16-electron group is technically feasible with even a home PC. With the VB2000 program, innovative VB calculations on relatively large molecules can be performed.

The following special features of VB2000 should be noted:

- The program has been carefully engineered and is highly modular. There is in principle no limit to the number of electron groups that can be recognized; the larger the molecule, the larger the number of groups to be selected. The actual

form of each group function is entirely at the discretion of the user. VB2000 provides a general platform for GF calculations of any kind (e.g. using a VB function for one group and a CASVB (CASSCF equivalent) function for another). Both CASSCF and GVB methods can be considered as special cases of GF Theory.

- The methodology used ensures that calculated energies are true variational upper bounds and that the results obtained are size consistent. The detailed forms of the group functions are *not* prescribed at the beginning of a calculation (as they were in early GF calculations [7], where the global basis was partitioned into subsets, each considered adequate for describing a particular group); they are determined variationally as a result of optimization of the wave function for the entire system, subject only to a strong-orthogonality constraint. This permits the study of chemical reactions, for instance, in which the subsystems representing reactants and products may have forms which change quite dramatically during the reaction process.
- In most cases, even a simple one-structure VB wave function can recover most of the corresponding CASSCF correlation energy with only a small fraction of the CPU cost for the CAS calculation. The use of wave functions of VB type for the groups of greatest chemical interest ensures correct behavior when bond-breaking processes occur within a group, and also ensures that the wave functions describing separate fragments possess a high degree of localization.
- The GF theory approach used in VB2000 leads to wave functions which are highly compact and therefore provide clear links with chemistry and 'chemical intuition'. For example, the electron density for the entire molecule is always the sum of the densities of its constituent parts -- even though the forms of the group densities may change appreciably in some chemical processes. This provides a firm basis for the discussion of the 'additivity rules' which apply to many molecular properties.
- Very often only a small part of a large molecule is responsible for certain chemical and physical properties. The great flexibility of VB2000 allows one to study such a limited region, with a high-precision wave function, using more approximate group functions (such as Hartree-Fock) to calculate the mean field provided by the rest of the molecule.
- One important feature of the VB method is that the optimized orbitals tend to be highly localized. The GF framework combined with localized VB group functions provides the possibility of developing a linear scaling algorithm for truly large systems by harnessing the exponentially increasing computer power.

1.2 What is new in version 2.6?

The VB2000 2.6 release is primarily due to the fact that from this point VB2000 will be released as part of the official GAMESS(US) release. Everything needed to get the GAMESS(US)/VB2000 program working will be found in the directory “vb2000” below the “gamess” directory.

There are however some new features:-

- \$MOLDEN added to the visualization directives to produce a Molden format file for use by the Molden program.
- Extension of the \$PLTORB visualization directive to read data so that a “ready to go” input file is prepared for the pltorb program. Previous the file produced had to be manually edited.
- Make the visualization routines prepare files for the LMOs, rather than the VB orbitals, if the TEST directive is on the command line.
- Improved memory allocation in GAMESS(US)/VB2000. It now uses the maximum of the remaining memory from the GAMESS code or that specified by \$MEMORY

1.3 What is new in version 2.5?

The new functionality and enhancements in VB2000 2.5 release are listed as follows:

- 64 bit Linux system support. This allows access to much large memory.
- Extension for Spin-Coupled method. A new method SC(m,n) is added, where m is the number of electrons, and n is the number of orbitals. In the new definition, SC(m,n) specifies all spin-coupled states of all electron configurations of m electrons in n orbitals. The configurations are defined as those that have the maximum number of occupied orbitals. When m=n, SC(m,n) becomes the original SC method.
- Localized non-orthogonal Hartree-Fock calculations. Better localized orbitals with better transferability can be obtained. Overlap penalty is used for the optimization of non-orthogonal Hartree-Fock molecular orbitals. This is necessary to avoid the collapsing of non-orthogonal MOs.
- Grid format of orbital visualization files which can be displayed with Accelry's Discovery Studio Version 2.5 and up.
- Projection to use spherical harmonic basis functions. GAMESS and VB2000 stand-alone use Cartesian basis functions internally. Some basis sets are defined in spherical harmonics, such as cc-pVnZ basis sets.
- Computation of VB orbital centroids and extents. This function is only

available in GAMESS-VB2000.

- Extension for the maximum number of electrons in each VB group. Up to 16 electrons per VB group can be handled.

1.4 What is new in version 2.1?

The new functionality and enhancements in VB2000 2.1 release are listed as follows:

- New expression of initial VB orbitals using atomic orbitals, LMOs and their linear combinations.
- New VBO libraries for more basis sets.
- Many enhancements, including better memory management for large number of basis functions, increasing the maximum number of basis functions in the whole molecule as well as in an individual VB group and several bug fixes.
- Improvement of convergence for VBSCF, especially for enhanced localization of VB orbitals and BOVB calculations.
- VB2000 2.1 is updated for the comparability with the current version of GAMESS (2009).

1.5 What is new in version 2.0?

The new functionality and enhancements in VB2000 2.0 release are listed as follows:

- Adjustable control for the localization enhancement of VB orbital optimization.
- Strict localization of VB orbital optimization.
- Interface to GAMESSPLUS for VB-solvation calculation in GAMESS. \
- New user interface (UI) of VB2000 on Windows platform can now handle both VB2000 and GAMESS inputs. For GAMESS calculation, a user should have WinGAMESS installed. Since WinGAMESS has VB2000 module compiled in, a user can also run VB calculations with WinGAMESS once a VB2000 license has been obtained.

1.6 What is new in version 1.8(R2)?

The following functionality and enhancements have been added in version 1.8(R2)

- Spin-density of VB wavefunction. A direct summation method for spin-density calculation of VB wavefunctions has been implemented.
- DIIS convergence control. The Direct Inversion of Iterative Subspace method has been implemented for the convergence control of VB orbital optimization.
- Enhancement for multi-structure VB calculations with dynamic VB orbitals (BOVB type calculations). The VB orbital optimizer allow user to use a set of VB orbitals which can be linearly dependent.
- Extended map files. The map files for N=1 and 13 are added to the MAP directory. (N is the number of electrons in a VB group).
- Many bug fixing. The new version allows VB(1) calculation. Two crashing bugs have been fixed: the VB calculation on a single atom and triplet state of two electron VB.

1.7 What is new in version 1.8?

Since the first formal release of VB2000 (Version 1.4) in 2000, we have made numerous improvements and added new functionalities. The most significant developments that have been made since version 1.7 are:

- VB2000 has been fully integrated into two major quantum chemistry programs, GAMESS(US) and Gaussian so that the some of the functionalities of GAMESS(US) and Gaussian can be used for VB wave function. See the **Installation Guide**
- Valence Bond Orbital (VBO) Library Files have been introduced for construction of initial VB orbitals. The simple intuitive expression of VBOs provides extreme flexibility for VBO construction.
- The limitation for the number of basis functions for the integral module has been removed so that larger molecules can be treated in the stand-alone version.
- Graphic utilities have been added for VB orbital visualization.
- There have been other technical enhancements:
 - ◆ Introduced a 'dot' notation for group function calculations.
 - ◆ Introduced dynamic memory allocation.

- ◆ Added many more built-in basis sets.
- ◆ General efficiency has been Improved.
- ◆ Added many new controls.
- ◆ Enhanced the portability on many more platforms/compilers.
- ◆ Enhanced Pipek-Mezey localization routine.

The Pipek-Mezey MO localization algorithm becomes ill conditioned when orbitals are nearly localized on the same atom; this is because the localized molecular orbital (LMO) function value changes very little for the mixing of two orbitals nearly localized on the same atom. For this reason, the final LMO orbital coefficients and the LMO orbital energies becomes very sensitive to the convergence criteria and platforms. Even though in most VB calculations, the small differences of LMO do not affect the final results, it can cause confusion. To reduce the numerical instability of the PM localization, we added some numerical controls; if the change in value of the objective function of an LMO is smaller than a certain threshold, then do not perform orbital rotation. Another condition to improve the stability is to first prevent the orbital mixing of energetically close MOs. Thus the inner atom core orbitals of different energies on the same atoms are not allowed to be mixed at the beginning. In this way, the mixing of orbitals localized on the same atom is reduced significantly.

- ◆ We have introduced maximum localization of the VBOs in CASVB calculations.

Since the CASVB is a full CI wave function in a subspace, the orbitals that span the orbital space could be totally arbitrary. One method to reduce the arbitrariness is to use a localization scheme. In CASVB calculations, the orbitals are not necessary orthogonal to each other, therefore this gives some extra flexibility for the orbital localization. To take advantage of this, a method to achieve the maximum localization of an VB orbitals has been introduced and implemented in VB2000. In this method, the CASVB calculation starts from a set of good initial VBOs, and the localization is performed at the last macro iteration to maximize the localization object function for each VBO within the CAS orbital subspace. This procedure leads to a set of highly localized AO-like orbitals, and the CASVB wave functions are usually very compact (i.e. only a few VB structures give significant contributions).

- ◆ We have added more Map files.

Table 1: Map files with the following combinations of number of electrons (N) in the VB groups and the number of unpaired spins (S).

<i>S/N</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>
0		x		x		x		x		x		x		x
1	x		x		x		x		x		x		x	
2		x		x		x		x		x				
3			x		x		x		x					
4				x		x		x						
5					x		x							
6						x								

The first row is the numbers of the electrons (N) in VB groups, and the first column is the numbers of unpaired spins (S). The mark "X" indicates that the map file of the corresponding combination of N and S is included in the package. Map files for (N=15, S=1/2) and (N=16, S= 0) are available on VB2000 website for download. If you need a map file for a combination of N and S not in the above table, consult the authors.

2 Installation Guide

2.1. Downloading VB2000 packages

VB2000 is distributed in two different packages: VB2000 and WinVB2000. The first one is for all UNIX-like systems, which include UNIX, Linux, and Cygwin. The second is for pure Windows, which contains the VB2000 executable on Windows platforms, the VB2000.NET user graphic interface based on Microsoft .NET technology and the associated run scripts. This section describes the steps for the installation on a UNIX-like platforms. The following instructions assume that a user has already received the tar file of VB2000 package. If not, the package is available (without charge) on the web at <http://www.scinetec.com>. Please follow the instructions on the web site and sign a user's license agreement and fax it to (international) +1-858-675-9686, SciNet Technologies, or mail to vb@scinetec.com (the scanned copy of the signed paper, either in PDF format or any image format, send by e-mail is preferred), and you will receive the password for the access of the download link, usually within 24 hours.

Use the following commands to unpack the compressed tar file on UNIX-like platforms (Unix, Linux and Cygwin):

```
tar -zxvf vb2000.tar.gz
```

If the -z flag OS not supported on your particular version of UNIX, type:-

```
gunzip vb2000.tar.gz  
tar -xvf vb2000.tar
```

Once the tar file is unpacked successfully, one should see a new directory VB2000 created in the current directory. This is the main directory of VB2000. Go to this directory and you should see the following subdirectories:-

BASET	Basis set files.
DOC	Directory contains this manual.
MAP	Auxiliary files of algebrant algorithm
SRC	Source code.
VBOLIB	VBO library files
TESTINP	Input files of test cases.
TESTOUT	Output files.
TOOLS	utilities

For WinVB2000, see section 2.3.

2.2 Installing VB2000 as a stand alone program on a Unix-like platform

To install VB2000 as a stand alone version, one can use the installation script *install* in the VB2000 root directory and follow the instructions for various platforms and compilers. For GAMESS-VB or Gaussian-VB installation, please read the corresponding specific instructions.

To compile the source code, one can run the installation by issuing the following command:

./install

When the compilation is completed, an executable file *vb2000.exe* will be found in the current directory, i.e. the main directory *VB2000*. Check the log file for possible error messages. The installation takes about 1-2 minutes. If your system is not one of the ones included in the install script, you can probably amend one of the choices to fit your system. Please pass on any successful examples to us.

2.3 Install WinVB2000 on Windows platform

The package for Windows platform is WinVB2000. The package contains two major parts: the WinVB2000 executable and the VB2000.NET GUI. The whole package is packed into a Windows Installer file, VB2000NET.msi. The installation of WinVB2000 on Windows is simple and fully automatic. One should follow the instruction below:

1. Download the WinVB2000 installation package VB2000NET.msi from VB2000 website at <http://www.scinetec.com> (make sure you have applied VB2000 license and have received the password for the current version).
2. Close other Windows applications.
3. Double-click on the VB2000NET.msi file that was downloaded.
4. Follow the directions given during the Setup procedure.

The installation takes less than a couple of minutes to complete, and an icon for the VB2000.NET Tool is placed on your Desktop:



VB2000.NET

By default, the package will be installed in the C:\VB2000NET\ . You can change the

default installation location to any folders (however, if you wish to run GAMESS job with VB2000.NET interface properly, you should not change the default location). If your C drive does not have enough free disk space, you may want to install VB2000 on a different hard drive. You will find in that folder:

- VB2000 (folder for WinVB2000 executable and run scripts)
- VB2000Jobs (folder for holding inputs of jobs in a queue and completed jobs)
- Examples (folder for inputs and outputs of examples)
- VB2000.NET executable and README file in HTML format.

There are two ways to run VB2000 program on Windows. The simplest way is to run VB2000 with the VB2000.NET GUI. Double click the VB2000.NET icon on your desktop, a simple VB2000 graphic interface will show on your screen. Prepare your input file with any text file editor, and drag-and-drop the input file into the upper box of the graphic interface. You can even drag-and-drop multiple input files into the GUI. You can also click the Help link on the control panel for quick start help.

It should be noted that the VB2000.NET GUI needs .NET frameworks to be installed on your Windows system, which can be downloaded for free. You should download version 1.1, **NOT** version 2.0. The 1.1 version can be download from VB2000 website at

www.scinetec.com/~vb/public/download.html

One can also use WinVB2000 in non-graphic mode in case you have difficulty for the installation of .NET frameworks. If you installed a wrong version of .NET framework, you may need to uninstall that version using

Control Panel->Add or Remove Programs

Another way to run VB2000 is in console mode. The VB2000 folder under the main VB2000.NET folder is the directory to run the calculation in DOS mode . You will find in VB2000 folder:

- winvb2000.exe – executable
- 6 DOS bat files
- BASET – folder contains basis set information.
- MAP – folder contains the VB2000 map files.
- VBOLIB – folder contains the VBO library files.
- TEMPOUT – folder to hold the output files of tests.
- WORK – a work directory.

To run the tests, type "VBRUNALL" in a DOS Window, or double click the file vbrunall.bat if you are using Windows Explore to browse the files. This will delete any output files in TEMPOUT and run the input files as specified in vbrunall.bat. There are 16 tests in the test script. It will take 10-20 minutes for the tests. When it has finished, use Wordpad to look at the output files in TEMPOUT. They can be compared with the files

in TESTOUT. Most of the scratch files that are used in the VB2000 folder will have been deleted. However files, with an extension "V84" are used as restart files, so they are not deleted. If you are generating some visualization files, all those files will also be copied to TESTOUT. You can delete other scratch files by typing "CLEAR".

To run your own jobs, you must create the input files in the WORK folder. You then run them from the VB2000 folder by typing "VB FILE", where FILE.INP is your input file. The WORK folder already contains 18 input files that you can also use as tests.

You should note that if you are running the calculation from VB2000.NET GUI, all input and output files, including the visualization files, are moved to the CompletedJobs directory. The CompletedJobs directory can be accessed from the VB2000.NET interface.

2.4 Installing WinGAMESS (with VB2000 built-in)

The latest version of WinGAMESS, the Windows version of GAMESS is VB2000 ready. The VB2000 code is compiled in the current version of WinGAMESS. To be able to take the full advantages of GAMESS and its combination with VB methods, it is recommend to have both WinGAMESS and WinVB2000 installed.

2.4.1 Run WinGAMESS installer

The WinGAMESS installer can be obtained from GAMESS website. The installation on Windows is just a few simple mouse clicks. The whole procedure is self-explained and easy to follow.

2.4.2 Run WinGAMESS from WinVB2000 Interface

WinVB2000 can run VB2000 jobs by drag-and-drop the input files into the user graphic interface. It not only handles VB2000 inputs but also GAMESS inputs as well. The WinVB2000 interface is capable to detect whether a input file is a VB2000 or a GAMESS job. If it is a GAMESS job, it will automatically invoke WinGAMESS program for the computation. The results can also be retrieved from VB2000 interface in the same way as for a VB2000 job. Running GAMESS jobs from WinVB2000 interface is a little bit more user-friendly than the WinGAMESS itself, which requires additional steps to prepare run-scripts by another program.

2.5 Installing GAMESS-VB2000

From version 2.6, VB2000 is part of the standard GAMESS release. The **vb0000** directory, below the **gamess** directory defined by \$GMSPATH, contains the VB2000 files. Note that this is GAMESS(US) not GAMESS(UK). The following instructions assume that you have installed GAMESS released in 2012 or later.

2.5.1 Step by step instructions

After running "config", carry out the following steps:

Step 1. Compile two VB2000 modules by typing the following commands:

```
./comp vb2000
./comp vb2gms
```

If you decide to compile all GAMESS and VB2000 modules, you can modify the script **compall** by setting VB2000=true, and type

```
./compall
```

Step 2. Link GAMESS with VB2000

You need modify the script **lked** by setting VB2000=true and then type

```
./lked
```

You may want to alter the VERN0 to give a different executable from the one linked without VB2000. This will create a GAMESS executable with VB2000 module included and now it is ready to run the VB test jobs. The compile and link steps can be combined by using the **compvb** script in the **vb2000** directory, but you need to move it up to the **gamess** directory. Make sure you take a look at its contents first. This is also useful if you want to keep the stand-alone sources synchronised with the vb2000 sources in the gamess tree.

Step 3. Test GAMESS with VB2000

Move to the **vb2000** directory and type

```
./runallvb
```

This will run all the vb2000 tests from the tests directory. You will need to read **gamess.README** and **test.README**. The **runallvb** script uses the **rungms** script in the main GAMESS directory. You need to check the **runallvb** script to ensure it is using the correct version and scratch file. When the jobs have finished, a quick check is to type:-

```
grep TERMINATED exam-vb*.log
```

This should show that all jobs have TERMINATED NORMALLY. Then type:

`./checkvbtst`

which will carry out a more extensive check of the outputs. This script works like `checktst` for the normal GAMESS tests.

2.5.2 Run GAMESS-VB2000

The GAMESS-VB2000 executable can be used for both normal GAMESS jobs and VB calculations as well. **gamess.README** gives a more detailed account. To run VB calculations, you need to ensure that two variables are set in your environment. The **rungms** script should already include the line:

```
setenv GMSJOBNAME $JOB
```

You can set `VB2000PATH` in `rungms` as:-

```
setenv VB2000PATH $GMSPATH/vb2000
```

but make sure that entry is just below where `$GMSPATH` is defined. Alternatively you can set `VB2000PATH` in your `.cshrc` or `.bash`, depending on which SHELL you are using. It must point to `$GMSPATH/vb2000rc` file, but with `$GMSPATH` expanded of course to the actual path.

2.6 Installing Gaussian-VB2000

The `vb-gaussian` directory below the main VB directory is where you will find what you need to compile the Gaussian-VB2000 version. You must have the full source code of Gaussian and the appropriate Gaussian Inc. supported compiler (normally the Portland compiler). First, read the file **gaussian.README**. The VB2000 code is introduced into Gaussian as a new module – `l511.exe`. You need to add a small section in `bsd/main.F` for the new module, and then recompile the main program (called `g98` or `g03` or `g09`). Then run the script called '`c-l511`' after editing it to define the correct path to the VB2000 source files. This could be done by using global variable `$VB2000PATH`, which must be defined for running VB2000. This creates `l511.F`. You then compile the `l511` module in the standard way for compiling one module – '`mg l511.exe >& l511.log &`', if you are using the C shell. Check for the presence of `l511.a` and `l511.exe` and read `l511.log`. More details are in the **gaussian.README** file. In this file, you will also find details of how to run Gaussian-VB2000 and, in particular, how to run a series of tests. The Gaussian-VB2000 version works with Gaussian 98, Gaussian03 and Gaussian09. To run the tests, check that the `rungau-vb` script uses “`g98`”, or “`g03`” or “`g09`” as appropriate.

Warning: We do not have access to the Gaussian03 or Gaussian09 code and we can not therefore easily track errors. We are aware that some large cases that work with Gaussian98, fail with no error message with both Gaussian03 and Gaussian09.

2.7 Systems supported

Considerable efforts have been put into the improvement of the portability of the VB2000 code. VB2000 has been tested on several platforms with different compilers. It is expected that the code can be ported on to many other platforms without major changes. The details of machines, operating systems and compilers that VB2000 has been tested on are as follows:

2.7.1 Linux

Linux is pretty dominant these days for all codes and most of the development of VB2000 has been under Linux or the related Cygwin. We have used several versions of linux, including Redhat, Fedora Core, Fedora Enterprise, SUSE and Ubuntu. Several compilers have been used:-

- a) **gfortran**. Many versions of this compiler have been used with success.
- b) **g77**. In the past, we used most versions of this compiler and had no problems.
- c) **Intel** compiler. We have used several versions of this compiler, going back to version 6.0. Both the intel C compiler, icc, and gcc can be used for the C routines. This appears to work fine for the stand-alone version. For gamess, some versions cause diagonalisation problems in the initial guess for SCF.
- d) **Portland** compiler. This is not free, but is the only compiler under Linux supported by Gaussian Inc for Gaussian. We have used version 3.2-4, and more recently 10.6 on 64 bit linux both the Portland C compiler and gcc can be used for the C routines. Like ifort, with GAMESS, there are some diagonalisation problems in the initial guess for SCF.
- e) **g95**. We have some experience of this compiler using similar flags to g77.

With the stand-alone version, the appropriate flags are given in the install script. It is difficult to select the best choice of flags for a particular piece of code. It is not clear that our choice is the optimal. Our choice is often based on the choice in the GAMESS(US) comp script.

With GAMESS/VB2000, we made no major changes to the standard flags for all the above compilers.

The Gaussian98/VB2000 version on 32 bit linux machines was compiled with the Portland compiler supported by Gaussian Inc. Gaussian03C/VB2000 test was compiled on a 64 bit SGI Altix running SUSE SLES9 linux with the Intel 7.1 compiler (specifically intel-cc/7.1.040 intel-fc/7.1.042 intel-mkl/7.2.008). Recently we have successfully compiled G03D and G09 with the Portland compiler version 10.6.

2.7.2 Cygwin

Cygwin is pretty much like Linux. It provides great convenience to work in both Windows and a Unix-like environment. Both g77 and gfortran compilers are used for our developments. Our very recent tests show that the gfortran compiler has excellent performance, especially for GAMESS.

2.7.3 Silicon Graphics IRIX64

[The experience with this system is now quite old and we no longer have access.]

The flags used for the SGI f77 compiler with the stand-alone version are in the install script. Note that these include '-i4', not '-i8'. With the GAMESS/VB code we used the same flags but with '-i8' as suggested in the comp file, but lowered the optimization to '-O2' from '-O3'. Some tests suggested that this made virtually no difference to run times, but speeded up the compilation times significantly. There are problems in the module vb2000ints.src that prevent use of this code with '-i8' on 64 bit computers, so '-i4' should always be used for the stand-alone version. This does not apply to the GAMESS and Gaussian versions which do not use the code in this module.

After some changes to ensure both 32 bit and 64 bit compatibility, the Gaussian98/VB2000 version worked fine with the f77 compiler

2.7.4 Windows platform

The version distributed as WinVB2000 is compiled under cygwin with the “-no-cygwin” flag so it uses the MinGW include files and libraries. It gives a stand-alone executable. Note this is different from WinGamess. Here the cygwin dll is needed. Look out on the VB2000 web site for details of WinGamess including VB2000.

We also tried, some years ago, to compile VB2000 with the Salford compiler but failed to obtain a working executable.

2.7.5 Mac OS X

VB2000 can be successfully installed on the Mac OS with the gfortran compiler, but we have limited experience.

2.7.6 The future

We look to users to provide feedback on the use of VB2000 on other machines and under other environments. Information on other machines and compilers will be available on the VB2000 web site as it becomes available from us or others.

2.8 Running the test cases on UNIX-like platforms

The script **runtest** can be used to run the test jobs. To do this, first go to the directory **VB2000**, and type the command:

./runtest &

The test jobs run sequentially, and the output files will be put in a temporary directory **TEMPOUT**. The test run output files are also included in the package, in the directory **TESTOUT**. One can compare the output files in **TESTOUT** and **TEMPOUT** to verify the correctness of the installation. Use the script **vbqa.pl** to do a simple comparison of the results. Just type **./TOOLS/vbqa.pl**. You must have Perl. You may have to check the first line of the script to make sure it points to the correct location of Perl. You may have to make the file executable.

3 Capabilities of VB2000

3.1 Hartree-Fock molecular orbital theory

A Hartree-Fock molecular orbital calculation is the starting point for any other type of calculation. Although VB2000 is not designed for pure Hartree-Fock (HF) method, we have made a number of improvements over the last few years. One major improvement is that the maximum number of basis functions that can be handled by the program has been significantly increased. The HF orbitals are used for electron group assignment, initial guess of VB orbitals and initial one-electron densities of all groups based on Pipek-Mezey localization as described in the next section.

3.2 Pipek-Mezey localization

The Pipek-Mezey (PM) method for localization of molecular orbitals (MOs) has been implemented. This utility plays a critical role in VB2000. First, the LMOs provide a solid base for dividing electrons into spatially separable groups. The LMOs of each electron groups are the starting points for generating the initial group functions and the corresponding one-electron densities. One of the initial guess methods for VB orbitals is to split each bonding LMO into two overlapping AO-like orbitals. See more details in Chapter 7. The PM method is based on a simple criterion: the localized MOs should maximize the sum of the squares of Mulliken charges of all atoms from each individual MO. The PM algorithm is also very simple, being based on successive rotations of orbital pairs. This method usually leads to results which are remarkably similar to those obtained from more sophisticated methods, and the cost in CPU time is quite negligible.

3.3 Modern *ab initio* VB method

A major feature of VB2000 is the implementation of modern valence bond theory at *ab initio* level using the algebrant algorithm. The VB engine in VB2000 has been implemented in the most general form. It can be used to perform very general VB calculations[9-12] such as:

- *Multiple-Structure VB (VB)*
- *Spin-Coupled Theory (SC)*
- *Complete Active Space VB (CASVB)*
- *VB with Localized Electron Pairs (GVB)*
- *VB Configuration Interaction (VBCI)*
- *Breathing Orbital VB (BOVB)*

VBCI implements a multi-structure VB calculation without orbital optimization. Usually one starts with the optimized VB orbitals obtained from a single structure VB calculation.

This option can be useful when the further optimization of VB orbitals with multi-structures becomes expensive. Our experiences show that further optimization with multi-structures has little gain as compared with the corresponding VBCI calculation with optimized orbitals of a single structure calculation. The output from VB, SC and VBCI calculations normally includes the weights of all resonance structures.

3.4 Group Function Theory and its combination with VB method

As indicated in the Introduction, an essential feature of VB2000 is the implementation of the Group Function Theory, in which a molecular wave function is expressed as an antisymmetrized product of individual group functions [6-8]. The spin-orbitals (one electron 'groups') of a Slater determinant are thus replaced by many-electron group functions - each of which may allow a considerable degree of intra group electron correlation. In VB2000, the electron-correlated group functions are obtained by the VB method. Even though the general mathematical formulation of this approach was completed more than four decades ago and it provides a unifying frame work for many varieties of electron correlation calculations, the general implementation of GF Theory, especially its combination with VB method, was introduced for the first time in VB2000. The general implementation of GF Theory in VB2000 provides tremendous flexibility for this type of computation: a molecular system can be divided into any number of electron groups and each group can be treated at any level of electron correlation of the user's choice. A common scenario is that a higher level of theory (such as CASVB) is used for the most critical regions in a molecule, for instance the bond breaking and forming region, while for the remaining parts some less accurate methods (such as simple VB, even Hartree-Fock) are used. Even though the general idea of using different levels of theory for different part of a molecule is somewhat similar to QM/MM or ONIOM, the approach in VB2000 is fundamentally different from QM/MM and ONIOM. In QM/MM, the system is not described in full quantum mechanics, and the QM/MM boundary is usually a source of trouble due to the lack of consistency for treating a molecule system. ONIOM is basically a highly sophisticated interpolation method based on the computation of different sizes of model systems and different levels of theory. In contrast, the GFT method is a fully quantum mechanical method with a consistent Hamiltonian. Moreover, the approach as a whole is *variational*.

3.5 Energy profile of chemical reactions

VB2000 provides a functionality for the automatic calculation of the energy profile of a chemical reaction along the reaction path guided by a few intermediate structures (such as reactant, transition state, and product etc.). Only for the simplest cases, the Linear Synchronous Transition (LST) method is a good approximation for the reaction path between reactant and product geometries. However, if a few good representative intermediate structures along the reaction path are available, one can use the linear

interpolation between each two consecutive structures for constructing a reasonably good path for the whole reaction. Assume G_i and G_{i+1} are two consecutive structures, and G_x is the linear interpolation point between them, then the Cartesian coordinates of G_x can be obtained from the following equation:

$$G_x = x G_i + (1-x) G_{i+1}$$

To make the geometry change as smooth and possible, the structures $G_i (i = 1, \dots, N_{\text{Geoms}})$ should be aligned for the best overlap between each two consecutive structures. This is done automatically by the program using the [Kabsch](#) algorithm. The geometries of the representative structures along the reaction path can be obtained from [an independent](#) source (for instance, [an IRC](#) calculation at a lower level of theory, or even created manually according to some basic knowledge of a chemical reaction). VB2000 also provides controls for how fine the resolution is for the geometry changes [along](#) the reaction path. Details of the controls in the input are given in the INPUT section of this manual.

3.6 Structure weights

Important information from a multi-structure VB calculation is provided by the weights of the various VB structures. In VB2000, we have implemented three types of structure weight analysis, i.e. The Mulliken Type (also called Chirgwin-Coulson type[14]), Löwdin type[15] and Hiberty type[16]. The structure weights of the three types are calculated according to the following formulas. Let's assume that the multi-structure VB wave function is expressed as follows:

$$\Psi = \sum_i^M C_i \Phi_i$$

where Φ_i is a normalized VB wave function of structure i . The Chirgwin-Coulson structure weights can be calculated as following:

$$W_i = \sum_j^M C_i C_j S_{ij}$$

where S_{ij} is the overlap of structure i and j .

A drawback of above formalism is that negative values of weights can be obtained, which is not physical. A remedy for this the negative value problem is to use [a Löwdin](#) formalism. This involves the Löwdin orthogonalization of the structure wave functions, i.e.

$$\Psi = \sum_i^M C'_i \Phi'_i$$

where ϕ'_i ($i=1, \dots, M$) are the orthogonalized structure wave functions obtained by using the Löwdin procedure), and C'_i is the new coefficient of the corresponding structure i . The Löwdin weight can be calculated as the square of the corresponding coefficient:

$$W_i = C_i'^2$$

Another remedy for the negative value problem was first used by Hiberty, in which the squares of the original coefficients are used as the structure weights, i.e.

$$W_i = C_i^2 / \sum_j C_j^2$$

In ideal cases where all the structure wave functions are orthogonal, all the above three formulas give the same results. All problems comes from the case when structure wave functions are not orthogonal, and in this case, none of them are perfect. When the structure functions are not orthogonal discrepancies among the results are sometimes found; but these are usually small. If large discrepancies occur, the numbers obtained should be used with caution.

3.7 Spin density

The spin density for open shell VB wave functions can be calculated. Please note that by default, the open-shell VB group must be the last one.

3.8 Visualization

In the stand-alone, GAMESS-VB and Gaussian-VB versions, functionality has been added for plotting 2D VB orbitals, displaying 3D orbitals and molecular geometries. The 3D orbital files in Gaussian Cube format can be generated. The files can be visualized by a variety of visualization programs, such as VMD, Molekel, Ghemical, Pltorb, Molplt, gOpenMol, Pymol etc. To extend the graphic visualization with the industrial leading tool for molecular modeling and simulation, Accelry's Discovery Studio (DS) and its free version DS Visualizer, the 3D orbitals in grid format can be generated and displayed in DS Visualizer.

These are described in more detail in section 6.8

3.9 Strict localization and enhanced localization of VB orbitals

In most cases, VB orbitals are highly localized. The localized AO-like VB orbitals

provide a clear picture that each VB structure corresponds to a Lewis structure unambiguously. However, it is not uncommon that the VB orbitals have considerable delocalization over to the neighbor atoms. Such a situation happens more often in special bonding patterns. Some times, it is not clear whether the delocalization has a strong physical driving force or just a outcome of variational optimization. To study this, one can force the VB orbitals to be localized and to see whether this has significant impact on the energy of VB wavefunction or not. If the energy increases significantly due to the localization constraints, then one may add more ionic VB structures in the wave function, and thus we know the system has considerable contribution of ionic structures.

There are different ways to force VB orbital localization. One straight approach is to divide basis functions into groups, and only allow each VB orbital to be optimized within a particular predefined group of basis functions. This is the strict localization constraints as used in BOVB approach. A control for strict localization has been implemented in VB2000 version 2.0. An example for specifying strict localized VB orbitals is given in Section 8.19.

The strict localization of VB orbitals is a too harsh constraints and may lead to too high energy. It is not only cumbersome to use but also is limited to some very simple systems. Therefore, we have introduced a more flexible control for localization enhancement of VB orbitals. In this control, an adjustable parameter, which can be considered as a parameter for the strength of localization, is used. If this parameter is set to zero, then the VB orbitals are optimized under no constraints. If this parameter is set to infinity, then the VB orbitals are strictly localized. A particular useful application of such a control is for the cases that the structure weight analysis is needed while the VB orbitals are significantly delocalized so that the structure weights becomes somewhat ambiguous. Another application is for the case that the localization patterns of the final optimized VB orbitals are different from the initial guess.

3.10 Extended functionalities in GAMESS-VB and Gaussian-VB

A very significant technical improvement in VB2000 is its ability to be fully integrated with two third-party programs: GAMESS and Gaussian. In addition to all functionalities of the stand-alone version, some additional features have been added for VB calculations.

- ◆ Dipole moments and higher moments calculations, and other properties
- ◆ Geometry optimization
- ◆ Vibrational frequency calculations
- ◆ Effective core potential (pseudo-potential calculation)
- ◆ More built-in basis sets

4 Quick Start Tutorial

The simplest way to start a calculation on a UNIX-like platform is to submit a job in the installation directory. You can also run a job in a different directory by setting environment variable:

```
setenv VB2000PATH <Your VB2000 Directory>
```

or

```
export VB2000PATH=<Your VB2000 Directory>
```

where <Your VB2000 Directory> is the full path of your VB2000 installation directory. You can add VB2000PATH setting in your .cshrc or .bashrc file, depending on which SHELL you are using.

Before doing a VB calculation, the very first decision that a user should make is how many and which electrons (or which chemical bonds and lone pairs) should be treated by VB method. In this chapter, a few examples are provided.

4.1 Simple VB calculation of CH₄

The first example is a VB calculation of 8 electrons on the 4 C-H bonds of methane. To do the calculation, one can follow the instructions below:

1) Prepare an [input](#) file for CH₄

```
#! VB(8)/D95

TEST CH4 5 GROUPS (1 CORE + 4 BONDS)

0 1
6 .000000 .000000 .000000
1 -.880321 .417906 -.470561
1 .259946 .591217 .868290
1 -.202850 -1.018388 .304590
1 .823225 .009265 -.702319
```

Please note that you should always keep at least one blank line at the end of the input file.

- 2) Save it as example01.inp
- 3) Run the job
- ./vb2000.exe example01 >example01.out

This calculation produces one core orbital corresponding to the 1s core of carbon atom and 8 VB orbitals corresponding to the 4 C-H bonds.

4.2 How does it work?

Here are the basic steps for a typical VB calculation in VB2000.

4.2.1 Hartree-Fock calculation

The first step of a VB calculation is usually a Hartree-Fock molecular orbital calculation. The RHF method is used for a closed shell system and a quasi-ROHF method is used for an open shell system. In the quasi-ROHF method, the Fock matrix is constructed from the total density in the same as in the RHF method

4.2.2 Pipek-Mezey MO localization

The canonical molecular orbitals from a Hartree-Fock calculation are transformed into a set of localized molecular orbitals (LMOs) by Pipek-Mezey method. The generated LMOs are used for the following two purposes:

1. Group assignment for electrons on LMOs
2. Initial guess of VB orbitals by splitting bonding LMOs

The LMOs are grouped in the following types and order: inner core orbitals, lone pairs and bonding LMOs. For each type of LMOs, the orbitals are ordered in an ascendant order of their orbital energies. Each LMO has a label to indicate its type. The Mulliken population of each LMO is calculated and used to determine its LMO type. If a LMO has approximately equal electron distribution on two directly connected atoms, then this LMO is a bonding LMO between the two. If the population is localized on one atom, then it is either a inner core or a lone pair. If a LMO is localized on more than two atoms, and it is marked as a multi-center (MC) bonding orbital..

4.2.3 LMO group assignment

The program does the LMO group assignment automatically according to the input. For simple VB calculations the default assignment is usually what a user needs. For the above example, the input specifies VB(8), an eight-electron VB calculation. Since the molecule has 10 electrons in total, thus there is a Hartree-Fock core group with two electrons and a VB group with eight electrons. By default, VB2000 starts with the Hartree-Fock group and then the VB group(s) specified in the input. In the CH_4 case, the first LMO is assigned to Hartree-Fock group, and the remaining 4 LMOs, which are bonding orbitals corresponding to the 4 C-H bonds, are assigned to the VB group.

In some cases the default LMO assignment may not be what you need. One should explicitly assign LMOs to the appropriate groups by using \$LMOGRPMODIFY input. A

more advanced control for the same purpose is \$VBGA. See Chapter 6 for more details. One can always do a TEST run (keyword: TEST) run and check the order of LMOs.

4.2.4 Initial VB orbitals

By default, the initial VB orbitals for a VB group are generated by splitting the bonding LMOs assigned to this group, and the lone pair orbitals of the VB group are used directly as part of the VB orbitals. Each LMO is split into two VB orbitals. For example, if a VB group has L lone pairs and M bonding LMOs, then $L + 2M$ VB orbitals will be generated for the $2(L+M)$ electrons. Of course, one can always change the number of VB orbitals and the way for generating the initial VB orbitals by using addition controls in the input. See Chapter 6 for **\$\$\$VBORB** and **\$AOGROUP** controls.

If all above methods fail, there is a last resort: **\$READGUESS**. This option read the initial guess from a text file. The initial guess can be generated from any method. In some cases, you only need to modify a few VB orbitals. In the case, you can first generate such a file by using **\$WRITEGUESS**, and then modify it using any text file editor.

4.3 Two-group VB wave function of CH₃-CH₂OH

Input file:

```

#! VB(8).VB(8)/D95 PRINTALL

TEST C2H5OH (TWO VB GROUPS)

0 1
6 -0.4413 1.9095 -0.1486
1 0.1529 2.7834 -0.4194
1 -1.2624 1.8366 -0.8622
1 -0.8754 2.0884 0.8352
1 -0.2076 -0.2379 0.0887
6 0.4029 0.6280 -0.1666
1 0.8338 0.4543 -1.1531
1 2.0387 1.4420 0.5056
8 1.4477 0.7230 0.7814

$VBGA
1-2 => 2
1-3 => 2
1-4 => 2
1-6 => 2
6-7 => 3
5-6 => 3
6-9 => 3
8-9 => 3

```

In this case, the molecule is divided into two groups: CH₃- and -CH₂OH, each contains 4 covalent bonds and is described by a VB(8) group function. The system wave function is thus the generalized product of two VB(8) functions and the Hartree-Fock function of the remaining electrons. The \$VBGA flag specifies the VB group assignments. The first group is the Hartree-Fock, the second group is the first VB group, and the third the second VB group. The lines following \$VBGA flag can be read as: the covalent bond between atom 1 and atom 2 is assigned to group 2, and so on. For more details, please read Chapter 7.

4.4 GVB calculation of ethane

Input file:

```
#! VB(2) .VB(2) .VB(2) .VB(2) .VB(2) .VB(2) .VB(2) /D95

TEST C2H6 (7 SIGMA BONDS)

0 1
6 .000000 .000000 .000000
6 .000000 .000000 1.540011
1 1.027669 .000000 -.363311
1 -.513834 -.889987 -.363311
1 -.513834 .889987 -.363311
1 -1.027669 .000000 1.903322
1 .513834 .889987 1.903322
1 .513834 -.889987 1.903322
```

As a special case of GFT approach, the 7 GVB pairs can be described by 7 VB group functions each with two electrons in two orbitals. There is a shortcut for GVB specification in the keyword. By getting familiar with these terms, it will be easier to understand the inputs and outputs of VB2000. For the above case, the key word can be simplified as

```
#! GPF(8) /D95
```

By default, if there are no additional flags, the *GPF(n)* key word will be interpreted as n-1 GVB pairs.

4.5 CASVB(4,4) calculation of H₂O

Input file:


```

#! CASVB(4,4)/D95 PRINTALL

TEST H2O

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582

```

In this case, the four electrons of the O-H bonds are included in the CASVB calculation.

4.6 Run VB2000 in GAMESS

To run a VB calculation in GAMESS, you need

1. Add VBTYP=VB2000 in \$CONTRL block.
2. Include a normal VB2000 input in \$VB2000 block.

Here is an example of VB run in GAMESS:

```

$CONTRL SCFTYP=RHF COORD=UNIQUE VBTYP=VB2000 $END
$SYSTEM TIMLIM=20 MEMORY=2000000 $END
$BASIS GBASIS=STO NGAUSS=3 $END
$GUESS GUESS=HUCKEL $END
$DATA
Water STO-3G
C1
OXYGEN 8.0 0.0000000000 0.0000000000 0.0000000000
HYDROGEN 1.0 0.0000000000 -0.7572153434 0.5865355237
HYDROGEN 1.0 0.0000000000 0.7572153434 0.5865355237
$END
$VB2000
#! VB(4)/STO-3G PRINTALL

Water

0 1
8 0.0000000000 0.0000000000 0.0000000000
1 0.0000000000 -0.7572153434 0.5865355237
1 0.0000000000 0.7572153434 0.5865355237

$END

```

One major advantage of running VB2000 in GAMESS is that the properties of VB wave function can be computed. In the log file of GAMESS-VB run, the VB properties, including the dipole moment, are printed after the following line:

PROPERTIES FOR THE VB WAVEFUNCTION

The dipole moment is printed under the ELECTROSTATIC MOMENTS.

4.7 Tips and tricks

The following advices may help you to run VB2000 more effectively for your systems.

1. If you want to run a multi-group VB calculation, specify the VB groups explicitly using the “dot” notation, such as VB(8).CASVB(6,6).VB(8). You can specify the same kind of calculation by using GPF(ng) and other controls, but the “dot” notation is much easier to use and intuitive.
2. You can do a VB calculation of a molecule at one geometry and do another VB calculation on the same molecule at a different geometry by reading the VB orbitals of a previous calculation. This is not only a good way to speed up convergence of a new calculation, but also a necessary step for generating initial VB orbitals for a bond breaking calculation in some cases.
3. If a VB group has many resonance structures and you have difficulty to determine which ones make major contributions, you can try a CASVB calculation first and then check the biggest contributors.
4. Try to avoid running CASVB or SC for groups with large number of electrons ($N > 8$ for CASVB and $N > 10$ for SCVB). If you want to run SC(12), you can run a VB(12) for orbital optimization with one or a few structures, and then do a SC(12) as a restart calculation without further optimization of VB orbitals (using key word CIONLY).
5. The spin-density calculation can be very time consuming for group with more than 12 electrons.
6. If VBSCF for VB orbital optimization becomes unstable, you can add a constant to Hessian diagonal elements by using \$HESSCONST, starting from 0.1. A larger value can make the iteration more stable, but it can slow down the convergence. Use **PRINTALL** option to watch closely the iterations and adjust the parameter so that you can get the converged results without slowing down the iteration too much.
7. Using the **DIIS** option can reduce the chance of symmetry broken solutions if you wish to keep certain symmetry. See Chapter 6 for more details about key word **DIIS**. However, the **DIIS** option sometimes can lead to false convergence.
8. If the macro iteration is unstable (total energy becomes oscillating, you can set a damping factor for rigid rotation. See Chapter 6 for control **\$DAMPROT**.

We will post more tips for VB calculations on VB2000 website. Comments and suggestions from users are highly appreciated. Contact us at vb@scinetec.com.

5 Terminology

The following definitions and related concepts are frequently used in describing the technical details of VB2000. By getting familiar with these terms, it will be easier to understand the inputs and outputs of VB2000.

5.1 Electron group

A group of electrons moving in a particular region of a molecule. For instance, the electrons of the benzene molecule fall into three groups: the pi-electrons, the sigma-bond electrons and the core electrons. Even though electrons are indistinguishable, it is convenient to regard a number of electrons, described using a particular subspaces of basis functions, as an "electron group".

5.2 Group function

An antisymmetrized wave function describing a group of electrons in the mean field of all other groups.

5.3 System wave function

A generalized production function ([GPF](#)) describing the entire molecular system. Such a wave function can be constructed as an antisymmetrized product of group functions of all electron groups in the molecule.

5.4 Macro iteration

A round of group-by-group optimizations of group functions, followed by a 'rigid rotation' of the system wave function. Usually such a procedure is repeated until the system wave function converges. The group wave functions are optimized within their own subspace, and the rigid rotation is performed for the global optimization of the system wave function. In VB2000, the global optimization algorithm is very general and [applies no matter](#) how each group function is constructed.

5.5 Rigid rotation of a system wave function

Let us assume that the system wave function of a molecule is expressed in terms of M orthogonal orbitals (e.g. a set of Löwdin-orthogonalized orbitals). A 'rigid rotation' of the system wave function is one which all the parameters in the wave function, expressed in the orthogonal basis, are held constant (thus, the orbital coefficients, the VB structure coefficients, the one- and two-electron density matrices defined in the orthogonal basis are left unchanged) while the orthogonal basis are rotated in the M -dimensional space.

5.6 VB, VBSCF, SC, BOVB, CASVB and CASSCF

There is much confusion in the literature concerning VB terminology. In this section we try to clarify the one used in our present work. Most of the differences among VB computations are purely technical. The term 'VB calculation' generally means the wave function is constructed in VB form by defining VB orbitals and their spin-pairing patterns. In modern VB calculations the VB orbitals are usually optimized in some self-consistent way and this gives rise to the term 'VBSCF method'.

In simple cases (e.g. for a saturated molecule) a single VB structure can often provide a fairly good description of the electronic structure; but cases where the chemical bonds cannot be plausibly assigned in any unique way (as in reactions, where bonds may be continually formed or broken) a single structure is inadequate. No single pairing scheme serves to give a good description of the whole structure or process and a multi-structure VB function becomes necessary. One such a choice is the so-called Spin-Coupled wavefunction in VB form (SC), which uses one set of VB orbitals but all linearly independent spin-pairing patterns—without any change of orbital occupation numbers (i.e. without 'excitations').

Another approach used in describing bond-breaking/forming processes also employs a multi-structure VB function, but allows different sets of orbitals for different structures: this is the Breathing Orbital VB (or BOVB) method[17], in which the orbitals are allowed to expand or contract (i.e. 'breathe') on changing from one structure to another. The BOVB method can efficiently incorporate dynamic correlation in a VB wave function while keeping the compact form of the function.

A CASVB calculation is mathematically equivalent to CASSCF: it involves a set of VB orbitals which define the Complete Active Space and includes all linearly independent spin-coupled functions that can be constructed from them. CASVB is not designed as an efficient alternative to CASSCF, but it has some advantages that CASSCF does not provide. In particular, the orbitals have no orthogonality constraints and so can be highly localized like atomic orbitals. Since the CAS wave function is invariant under orbital transformations in the active space, we can require maximum localization of the VBOs, without constraints, in any function of CASVB type – even with a limited number of structures. The CASVB wave function is then also highly compact, a few structures providing almost all significant contributions. One should note the CASVB method implemented in VB2000 is different from Cooper's CASVB approach, in which the CASSCF wave function is transformed into VB form of a subset of CASSCF space.

VB2000 provides a very general implementation of VB theory and is capable of performing all the above types of calculation. The SC and CASVB keywords in VB2000 allow one to perform the specified computation without knowing any of the details involved in constructing all the VB structures.

5.7 Group Function Theory (GFT) vs. ORMAS

To the best of our knowledge, no other electronic structure program has a general implementation of group function theory which can perform full optimization of a wave function which is expressed as a generalized product of group function. GVB and MCSCF can be considered as two special cases of GFT. Quite recently, the so-called Occupation Restricted Multiple Active Space Configuration Interaction (ORMAS) method has been introduced in GAMESS(US)[18,19]. The general philosophy of ORMAS has a certain similarity to GFT in terms of multiple groups. But the two methods are actually totally different both technically and theoretically. It would be interesting to compare the two methods. The ORMAS method involves two steps: first the active space (a set of active orbitals) is partitioned into an unrestricted number of orbitals, and secondly occupation restrictions are specified in the form of the minimal and maximal number of electrons allowed for each group. ORMAS can be considered the most general implementation of restricted active space SCF method, thus all other variations such CASSCF and the 3-group RASSCF can be recovered by appropriate specifications of the restriction parameters. ORMAS efficiently eliminates ineffective configurations in CI computation while keeping the flexibility to account the electron correlations among groups. One should be aware that the flexibility for group-group electron correlation comes at the price the number of exponentially increasing of configurations - even though such an increase is considerably reduced by the restrictions.

The group function method approaches the electron correlation of large systems in a different way. It also involves two steps. The first step is similar to that of ORMAS, i.e. a molecule system can be partitioned into an unrestricted number of electron groups. The second step is quite different from ORMAS: 1) the number of electrons in each group is fixed, i.e. electrons are not hopping between different groups; 2) the wave function is expressed as the antisymmetrized product of group functions; 3) each group function is optimized in the mean field of the other groups thus the small inter-group correlation is neglected. A major advantage of this approach is that the computational cost increases only linearly with the number of groups instead of exponentially as in ORMAS. In principle, any correlation method can be used for the group function of a GF group. For instance, a GF group can be further divided into multiple subgroups and the ORMAS method can be applied to these subgroups.

6 Input Descriptions

VB2000 is designed to balance the flexibility of controls and the convenience of usability. Most of the controls and options needed are automated as defaults, which in most cases work well. This automation reduces the input to a minimum. On the other hand, many controls can be customized and will overwrite the defaults with explicit input, thus providing the maximum flexibility. One should also understand that VB2000 is not designed as a black box. Input backed by chemical knowledge is essential to obtain insights into interesting chemical systems. VB2000 is designed to provide convenient controls to convert sophisticated theory into practice so as to test, verify and refine theoretical ideas about the nature of chemical bonds. This chapter illustrates how to specify different kinds of calculations in the input files.

6.1 General considerations

The controls in the program are divided into three Levels, as follows.

Level 1: Defaults.

When controls are not set explicitly, the defaults take effects automatically, being preset according to the type of calculation.

Level 2: Global controls.

Most of the default settings can be overwritten by providing explicit inputs. There are two kinds of the global controls: (1) controls that concern the optimization of the total wave functions (including the number of subgroups, number of macro iterations, etc.); (2) controls that relate to a specific method. All groups treated by the same method can be dealt with using the same method-specific global controls. A global control can be either a key word or a dollar '\$' flag string.

Level 3: Controls for individual groups.

The optimization of the group wave function of any specific group can be controlled directly by a level 3 control. The general expression of a level 3 control is as follows:

*###CONTROL-NAME
CONTROL-CONTENT*

The control consists of two parts, i.e. the control flag and the control-content. The control flag always starts with a dollar ('\$') sign, followed by a two-digit number and the name of the control (no space in between). We use upper-case letters for all control names. The two digit number (here represented by ##) is the number of the group that the control is

applied to. The number starts from 01 up to 99. In most common cases, the first group is a Hartree-Fock group, and the second one a VB group. The control content can be either empty or a multiple line input.

6.2 Basic controls and the input of a molecule

The input for a calculation starts with ‘#!’ (in order to distinguish it from ‘#’ used in the Gaussian package) and followed by basic controls and molecule specification as shown below:

```
#! METHOD/BASIS_SET [OPTIONS]
    [Blank]
Title or comments (can be multiple lines)
    [Blank]
Charge, Multiplicity
Atomic#1, X1, Y1, Z1
Atomic#2, X2, Y2, Z2
.....
    [Blank]
[Other controls]
```

The first input can be continued and terminated by a blank line. Multiple lines of comments can be inserted before providing the numbers of charge and multiplicity. The comments are also terminated with a blank line. For each atom, the atomic number and its Cartesian coordinates should be provided. All these numbers can be given in free format. The atom input is also terminated by a blank line. After the atomic input, additional controls can be provided in the input. All these controls are described in the following sections.

6.2.1 Method

In the above, *METHOD* can be any of the key words explained below.

HF

If the *METHOD* is set to HF, then Hartree-Fock method will be used for the whole molecule. This is essentially a regular molecular orbital calculation.

VB(*m*)

VB method will be used for a group of *m* electrons. By default, only one VB structure will be included in the calculation. One can explicitly specify multiple VB structures by using an additional control flag \$##VBSTR (see section 6.4.3)

SC(*m*)

Spin-Coupled method is used for a set of *m* electrons in *m* orbitals. All linear

independent spin-configuration of m electrons in m orbitals are included in an automatic way. Therefore, there is not essential difference between $SC(m)$ and $VB(m)$ with multiple VB structures. $SC(m)$ provides a convenient shortcut for a special multiple-VB structure calculation that a user don't have to write all linear independent spin configurations explicitly. The keyword **SCVB(m)** can also be used.

SC(m,n)

We can ask ourselves: what would be the most natural way to extend Spin-Coupled method to systems where the number of electrons is not equal to the number of orbitals? In VB2000, we define it as the all spin-coupled states from all possible electron-orbital configurations which have the maximum number of occupied orbitals. For instance, in the case of $SC(m,m)$, there only one such a configuration, that is all orbitals are occupied with one electron on each, and all spin-coupled states are from the spin-coupling schemes from this configuration. Thus $SC(m,m) = SC(m)$. An essential advantage of SC method is that the wave function and optimized orbitals are independent of the way to choose the Lewis structures of the SC space, therefore, the results is not biased to any personal preference.

One can make special use of $SC(m,n)$ when $m=2n$, i.e. all orbitals are double-occupied. In this case, SC option turns into non-orthogonal Hartree-Fock method. In VB2000, we use this trick to do non-orthogonal localized Hartree-Fock molecular orbital calculations. Even though for regular VB calculation the maximum number of electrons in each VB group is currently limited to 16, the non-orthogonal Hartree-Fock MO calculations does not have such a limitation. The limitation for the number of non-orthogonal molecular orbitals comes from the limit of total number of basis functions which can be handled by the program and the computational time. It should be noted that the non orthogonal Hartree-Fock MO computation is much more expensive than the regular Hartree-Fock method.

CASVB(m,n)

Complete Active Space VB (CASVB) is used for a set of m electrons in n orbitals. It is mathematically equal to a CASSCF(m,n) method, i.e. the wave functions produced by the two methods have the same energy and properties. The CASVB method has some unique advantages that the orbitals in CASVB wave functions are nonorthogonal and can be maximally localized.

GPF(n_g)

This an old way to specify a multi-group VB calculation, where n_g is the number of electron groups including the Hartree-Fock core. Additional controls are required for this method. This notation will become obsolete in the future release. The most convenient way to specify a multiple-group VB calculation is by the dot notation which is described in the next section.

Dot (•) notation

The dot notation has been introduced since version 1.8 for specifying a multi-group VB calculation. This notation is a more convenient and intuitive than the GPF notation. For

example, **VB(8)·CASVB(6,6)·VB(8)** specify a 3 VB group calculation. One should note that if the molecule is a open-shell system, then the last VB group must include the open-shell electrons. Only one VB group can have unpaired spins.

6.2.2 Basis set

BASIS_SET is the name of basis set, which can be any one of the following:

STO-3G
STO-4G
STO-6G
D95 (Dunning/Huzinaga full double-zeta)
MIDIX (Minnesota MIDI! basis set)
3-21G
3-21G*
6-31G
6-31G*
6-31+G*
6-31G**
6-31++G**
cc-pVDZ
AUG-cc-pVDZ
TZVP
GEN (general basis set provided by user with a basis set file)
GENNG (alternative to GEN for minimum basis sets)

The first 15 are built-in basis sets. **GEN** is a flag for a user defined basis set: when specified, the program will expect additional input (after the atomic input) as follows.

&basis_set_file

where *basis_set_file* is the file name of the user-defined basis set, which should be put in the directory **BASET**. You can also provide the basis set file with the full path if the file is not in the **BASET** directory, for instance,

&/usr/people/jli/VB20000/GEN/basis_set_file

This file should be prepared in the appropriate format (see the format used for the standard basis set files in **BASET**).

GENNG is identical to **GEN** in all respects except that it is restricted to minimum basis sets, and it uses the STO-nG VBOLIB, while **GEN** does not use any VBOLIB. A program, *get_gen_sto6g.f*, is available to create STO-6G basis sets with general STO exponents. This is in the **TOOLS** directory below the **VB2000** directory in the main release and in **\$GMSPATH/vb2000/SRC** directory in the **GAMESS** release. Details of the input is given in comments at the top of the file.

6.2.3 Options

[OPTIONS] are key words that control the overall calculation.

PRINTALL

With this option, a lot more messages will be printed during the computation; for example, the initial VB orbitals, Mulliken population of LMOs and VBSCF iterations

RESTART

This option is originally designed for restarting a computation which is terminated premature. However, we find a more useful usage of RESTART is to read VB orbitals from a different calculation as the initial VB orbitals for a new calculation. A typical scenario of such a usage is generating the initial VB orbitals for a bond breaking process. In the dissociation limit, the Hartree-Fock calculation breaks down, and therefore the LMOs based on Hartree-Fock molecular orbitals becomes less meaningful. As a consequence, the method for generating initial VB orbitals from LMOs becomes problematic. The RESTART option can be used to solve the problem: do a VB calculation of the molecule at the normal bonding geometry, and then do same type VB calculation at the bond breaking geometry using the RESTART option. Also see the \$RESTARTFILE control. A VB restart file of a molecule can be even used for another VB calculation of the same molecule but with different number of electrons and geometry. Also see \$RESTARTMAPPING for more details.

CIONLY

This option disable VB orbital optimization during the VBSCF iteration. With this option, a multi-structure VB calculation becomes essentially a VBCI. However, this option does not disable VB orbital optimization completely. The overall wave function optimization involved the in-group optimization (VBSCF) and rigid rotation for group-group optimization. The CIONLY option has no effects on the rigid rotation. To disable rigid rotation, you need use the keyword: NOROT.

NOROT

There are situations that one needs to disable the rigid rotation between groups. The keyword NOROT is designed for this purpose. However, both CIONLY and NOROT does not disable the optimization of Hartree-Fock orbitals. To disable the Hartree-Fock orbital optimization, the following keyword is required.

FROZEN

Freeze all orbitals in the Hartree-Fock group

UNITS=BOHR

This keyword is required if the unit of the the 3D coordinates is Bohr. The default unit is Å

SPDEN

This keyword is required for the computation of spin density of open shell system. For close shell systems, this keyword has no effects. In the current implementation, the spin density calculation can be significantly slow down for VB groups with more than 12 electrons.

DIIS

This option can be used to stabilize the VBSCF iteration based on Direct Inversion of the Iterative Subspace (DIIS) method. The Newton-Raphson (NR) method implemented in VB2000 usually has very good performance for orbital optimization. However, there are cases that the VBSCF iteration becomes unstable. The DIIS option can stabilize the pure NR iteration. It is quite useful for BOVB type of calculations where multiple structures are included and different orbitals sets are used for different structures. We have also found that the DIIS option helps to maintain symmetry.

SPHER

This option is used to enforce the use of spherical harmonics basis sets. GAMESS and the stand-alone VB2000 use cartesian functions for basis sets internally. Some basis sets, such as cc-pVnZ, are officially defined as spherical harmonics. To avoid confusion as well as to keep consistency from one program to another, it is required to stick to the same definition for these basis sets. GAMESS enforces the use of spherical harmonics basis set by the option ISPHER=1, but the VB2000 plugin will still use cartesian functions, unless SPHER is used in the command line of the \$VB2000 block. In VB2000, the SPHER keyword is used for the same purpose.

TEST

This option is usually used to check the LMOs before doing a complete VB calculation. With this option, the program does the Hartree-Fock molecular orbital calculation, orbital localization and then stops. By checking the order and the characteristics of the LMOs one can decide whether it is necessary to use more controls or not to adjust the LMO assignment. TEST runs can use the various visualization routines, such as \$CUBE. The data produced allows you to visualise the LMOs and thus help in checking them.

6.3 Group function control flags

6.3.1 \$GENCTL

This is the main control for group function specification. The input format is as follows:

```
$GENCTL  
(MELE(i),i=1,NG)  
(METHOD(i),i=1,NG)
```

Where NG is the number of electrons groups (including the Hartree-Fock group). $MELE(i)$ is the number of electrons of group i , and $METHOD(i)$ is the method for obtaining wave function of group i . The following values of method are possible:

- 1 for HF
- 2 for VB (may contain multiple VB structures)
- 3 for SC
- 4 for CASVB

\$GENCTL is usually combined with the key work **GPF(n)** in the input. If **GPF(n)** is specified and the control flag **\$GENCTL** is missing, then the molecule is treated with (n-1) GVB pairs. Please note that by default, the first electron group is a Hartree-Fock group. Therefore, if a system is divided into n groups, only n-1 groups are expected to be treated with VB methods. If no specification is provided for the n-1 groups, then they are assumed to be GVB pairs.

6.3.2 \$GRPDIM

This flag specifies the dimension of the subspace for each electron group. The format of this control is as follows:

```
$GRPDIM  
(NDIM(i),i=1,NG)
```

where $NDIM(i)$ is the dimension of the basis for group i . In most cases, the default values for all groups will be generated automatically by the program. For instance, with a (close shell) Hartree-Fock group, the dimension should be equal to half the total number of the electrons in the Hartree-Fock group. If the explicit input number is different from that, an assertion error will be detected and the computation will be aborted.

6.3.3 \$VBGA (or \$VBGROUPASSIGNMENT)

Two \$ flags are introduced for this option: **\$VBGROUPASSIGNMENT** or a short notation as **\$VBGA**. This option provides an intuitive electron group assignment for valence electrons. The following example shows how the control is applied.

Example 6.1. VB calculation of H₂O with two C-H bonds.

```
#! VB(4)/D95 PRINTALL

TEST H2O

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582

$VBGA
1-2 => 2
1-3 => 2

$02VBORB
1-2
1-3
```

In this example, the \$VBGA input has two entries. The first one, 1-2 => 2, means that the electrons of the σ bond LMO between atom 1 and 2 are assigned to (=>) electron group 2 (here group 2 is the VB group). The second entry means that the σ bond between atom 1 and 3 is assigned to group 2. One should note that for such a simple molecule, the \$VBGA and \$02VBORB inputs can be omitted, and the LMO groups assignment and VBOs are generated automatically. The next example shows how to explicitly specify the group assignment and how to generate VBOs. For multiple bonds, for instance the triple bond in C₂H₂, one can assign all the three bonds to the same electron group with one entry as follows:

Example 6.2. VB calculation of the triple bond in C₂H₂

```
#! VB(6)/D95

TEST C2H2 TRIPLE BOND (SIGMA+PI)

0 1
6 .000000 .000000 .610009
6 .000000 .000000 -.610009
1 .000000 .000000 1.690006
1 .000000 .000000 -1.690006

$VBGA
1#2 => 2
```

the \$VBGA input is equivalent to the following:

```
$VBGA
1-2(1) => 2
1-2(2) => 2
1-2(3) => 2
```

The first entry means the first (σ) bond is assigned to group 2. The second entry means that the second (i.e. the first π bond) is assigned to group 2. The last entry assigns the third bond (the second π bond) to group 2.

6.3.4 \$LMOGRPMODIFY

The **\$VBGA** (or **\$VBGROUPOASSIGNMENT**) control is based on the automatic LMO classification, which assigns each LMO as a bonding orbital between two atoms or a lone pair (including the inner core) orbital on an atom. However, there are cases where such an automatic LMO assignment does not work well. For instance in the multi-center bonds and some conjugated systems, the LMOs are not necessarily localized on one or two atoms. Therefore the **\$VBGA** logic does not apply for such LMOs. To deal with cases of this kind, a less intuitive but more flexible control has been implemented in VB2000. This control explicitly assigns each LMO to a specific electron group. The general control format is as follows:

```
$LMOGRPMODIFY
NLMO
N1, G1
N2, G2
.....
```

Where *NLMO* is the number of LMOs, followed by *NLMO* pairs of integers (N_i, G_i). N_i is the LMO index, and G_i is the electron group that the N_i -th LMO belongs to.

6.3.5 \$MACROITER

The default maximum number of macro iterations is 12. If this default number needs to be changed, then **\$MACROITER** flag can be used as follows:

```
$MACROITER
MITER
```

where *MITER* is the new number of macro iterations.

6.3.6 \$ECONV

The default energy threshold for the total energy of a molecule is 10^{-6} a.u. This can also be overwritten by this control flag as follows:

\$ECONV
n

and the new energy threshold will be 10^{-n} a.u.

6.3.7 \$ROTATION

The optimization of a molecule wave function expressed as a generalized product function involves the optimization of each group function within its corresponding subspace and the Jacobi rotations between the basis functions of different groups. By default, only one rotation is performed for each pair during each macro iteration. However, this can be changed by using this control flag:

\$ROTATION
nRot

where *nRot* is the number of Jacobi rotations for each macro iteration. The default value is 1. In some cases, by increasing this number (say 2-5), the macro iteration can be made more stable. However, a large value of *nRot* also means a much more expensive computation.

6.3.8 \$NOTROT

There are some cases where we need to disable the Jacobi rotation for basis functions of two groups, such as the σ and π groups in a conjugated system. This can be done in the following way:

\$NOTROT
NR
NA₁,NB₁
NA₂,NB₂
.....

Where *NR* is the number of group pairs to be disabled, and followed by the *NR* pairs. The first pair is group *NA₁* and group *NB₁*, and so on. This option may make the program more efficient and converge faster. To give an example, benzene can be divided into two electron groups, the σ -electron group and the π -electron group. A set of virtual orbitals can be used to define a third group, containing no electrons. The optimization of the system wave function involves the inter-group rotations of the subspaces of all groups. Clearly, however, the rotation between σ and π orbitals should be avoided for benzene; and this can be achieved by turning off the inter-group rotation between the σ and π

groups. This option is used whenever a molecule has such kind of symmetry, thus avoiding inter-group rotations that would break the symmetry. For more details, please check the example inputs.

6.3.9 \$FULLHESS

The Jacobi rotation involves the evaluation of a Hessian matrix, which is usually an expensive procedure. To improve the efficiency, our experience shows where only evaluating the block diagonal Hessian matrix elements can be as good as the full Hessian matrix in most cases. Therefore, by default, only block diagonal matrix elements are evaluated, and any matrix elements involving rotations of more than two groups are ignored. However, there are cases that the full evaluation of Hessian matrix is essential, especially for systems with unusual bonding, as in chemical reactions. To enable the full Hessian evaluation, just put the above control flag in the input.

6.3.10 \$DAMPROT

Occasionally, one may see the macro iteration becomes unstable. This may happen for BOVB type calculations. If the total energy of the wave function oscillates, then a damping factor for rigid rotation will most likely solve the problem. The damping factor d should be in the range ($0 < d < 1.0$). A small factor means a strong damping effect, and can also significantly slow down convergence. A good starting point is 0.5. For instance, you can add the following control parameter in the input:

```
$DAMPROT  
0.5
```

6.4 VB Orbital optimization controls

6.4.1 \$VBSCF

This is a global SCF control for VB groups. The general formation of the input is as follows:

```
$VBSCF  
EPS, MaxIter
```

Where *EPS* is the threshold for the maximum change of orbital coefficients of VB orbitals, and *MaxIter* is the maximum number of VBSCF iterations. This control applies to all VB groups, unless the specific values are specified for a particular VB group. The default values of *EPS* and *MaxIter* are 0.0001 and 15, respectively. Also see the next control for a specific VB group.

6.4.2 $###VBSCF$

This control is very similar to the above control, but it applies only to a specific group as denoted by the number. The general format of this control is:

$###VBSCF$
EPS, MaxIter

Where $##$ is a two digit number of a group index. The default values are specified by $VBSCF$.

6.4.3 $###VBSTR$

This control is used for VB structure specification of a group. The general format of this control is:

$###VBSTR$
NSTR
N_{1 1} N_{1 2} N_{1 N}
N_{2 1} N_{2 2} N_{2 N}
.....
N_{i 1} N_{i 2} N_{i N}
.....
N_{NSTR 1} N_{NSTR 2} N_{NSTR N}

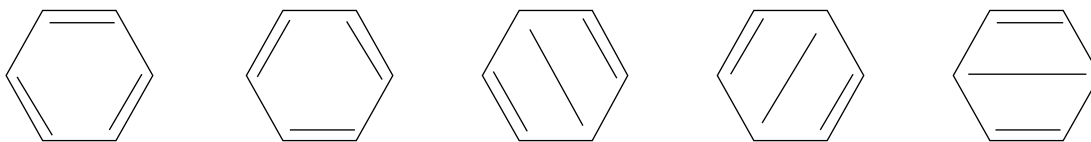
NSTR is the number of VB structures for a VB group, *N* is the number of electrons in this VB group. Each line specifies a VB structure, Each structure is specified by a set of orbital indexes. In VB2000, the following convention is used for the pairing pattern of each VB structure: If the total spin of the electrons of the current group is *S*, and the number of electrons is *N*, then there are $(N-2S)/2$ pairs, and $2S$ unpaired electrons. The pairing pattern of the first structure can be interpreted in the following way:

N_{1 1} - N_{1 2}
N_{1 3} - N_{1 4}
.....
N_{1 N-2S-1} - N_{1 N-2S}
N_{1 N-2S+1}
N_{1 N-2S+2}
.....
N_{1 N}

Note: The indexes of orbitals in the symbolic VB structure specification refer to the *index of the orbitals of the group*. For instance, if an index is 1, it refers to the first orbital in this group. Here the double sharp ($##$) stands for a two digit number. For instance, $02VBSTR$ stands for the VB structure of group 2. Note: the program expects a two-digit

number even it is less than 10.

Let's consider an example. The two Kekule structures and three Dewar structures of benzene can be represented as:



The 5 VB structures can be given as follows:

```
$02VBSTR
5
1 2 3 4 5 6
2 3 4 5 6 1
1 4 2 3 5 6
2 5 3 4 1 6
3 6 1 2 4 5
```

The first two are Kekule structures, and the last 3 are Dewar structures.

6.4.4 \$HESSCONST

This control is used to increase the stability of VB orbital optimization by adding a constant to Hessian diagonal elements. The formation of this input is:

```
$HESSCONST
D
```

where D is a constant, suggested value is 0.1. A larger value can make the iteration more stable, but also slow down the convergence. For difficult cases, one can try to divide the computation into multiple restart runs, and watch carefully the convergence behavior by using **PRINTALL** option. Once the convergence becomes stable but slow, one can reduce the constant D , or even set it to zero to speed up the convergence.

6.4.5 \$BRILLMASK

This control is used for VB orbital optimization under strict localization constraints. This control specifies a set of orbital pairs that are *not* allowed to be mixed during orbital optimization. Generally, only orbitals on the same atom are allowed to be mixed during orbital optimization. The resulting VB orbitals are strictly localized. To make this working, the initial VB orbitals must be strictly localized. One can start from a set of AOs as initial VB orbitals, and then each VB orbital is optimized within the basis functions of the atom what the initial VBO belongs to. The general format of this control is as follows:

```
$BRILLMASK
M
NA1, NB1
NA2, NB2
.....
NAM, NBM
```

Where *M* is the number of orbital pairs that the orbital mixing between the two is to be avoided. For instance, orbital *NA_i* does not have any contribution from orbital *NB_i* during orbital optimization.

6.4.6 \$##LENHANCE

This is a localization enhancement control. For some applications, the strict localization is a too strong constraint imposed on the orbital optimization and can lead to much higher energy of wave functions than that from free optimization. It is also technically cumbersome to set up the calculation and can only be applied in very limited systems. Therefore a more flexible and general way to obtain localization enhanced VB orbitals is needed. In VB2000 2.0, a new algorithm is implemented. In this new algorithm, a penalty function of delocalization of a set of VB orbitals is added to the total energy of the VB wavefunction. The delocalization of a VB orbital is defined as the Mulliken population of the VB orbital on atoms which are not in the localization target of the orbital. The VB wavefunction is optimized by minimizing the following objective function:

$$P = E(VB) + W * (\sum_{i=1}^N P_i)$$

where P_i is the delocalization penalty of VB orbital ϕ_i defined as follows:

$$P_i = \langle \phi_i | \overline{S}_x | \phi_i \rangle / \langle \phi_i | S | \phi_i \rangle$$

S is the overlap matrix of basis functions, ϕ_i is the VB orbital localized on atom X_i (or a set of atoms X_i), and \overline{S}_x is the delocalization overlap matrix defined as follows:

$$\left[\overline{S}_{x_i}\right]_{kl} = S_{kl} \text{ (if } k \vee l \notin X_i, \text{ otherwise } 0)$$

i.e. the delocalization overlap matrix \overline{S}_{X_i} is from the original overlap matrix S by setting the rows and columns of basis functions of X_i to zero.

Such a penalty can be scaled by a weighting parameter. The general format of this control is as follows:

```

$##LENHANCE
M
VBO1, N1, A1, A2, ... AN1
VBO2, N2, B1, B2, .... BN2
.....

```

where M is the number of VBOs to be localized. By default, the delocalization penalty weight W is set to 0.1, and can be modified by **\$DPWEIGHT** input (see the next section). The following lines specify how each VBO will be localized. For instance, for VBO_1 , it is localized on N_1 atoms for the atom list $A_1, A_2, \dots A_{N1}$. There are M lines to specify the localization of M orbitals.

This control for obtaining more localized VB orbitals has the following advantages.

- It is very general. It does not require and special orientation of a molecule and pre-classification of atomic basis functions.
- It is very flexible. The penalty weight control how much localization is needed. By setting a very large weight, the optimized VB orbitals are close to the strictly localization. We recommend **0.1** as a starting point. It is also recommended that the same weight is used for a comparison study of a number of molecules. One can also use the total energy change and the delocalization coefficients to make good judgment for a reasonable weight.
- The input is intuitive and relatively easy to set up. One only needs to specify which atoms a VB orbital should be localized on. Such a control does not need the details of the basis set used in the calculation.
- By increasing the delocalization penalty weight to an extremely large value, say 10^6 , one can get strictly localized VB orbitals. See the example input: TESTINP/h2bovb.inp. However, one should note that it is possible to get strictly localized VB orbitals for only one VB group, unless the multiple VB groups are separated by molecule symmetry (such as σ - π systems). Due to strong orthogonality constraints, it is generally impossible to keep orthogonality between groups and strict localization of orbitals at the same time.

6.4.7 \$DPWEIGHT

This controls the delocalization penalty weight for localization enhanced VB orbital optimization. The format is as follows:

\$DPWEIGHT
W

where *W* is the delocalization penalty weight defined in the previous section. Without this input, the default value is 0.1 for localization enhancement option.

6.4.8 \$CMAXCUT

This controls the Hessian matrix evaluation of VBSCF iterations. If the maximum coefficient change is less than the threshold specified by \$CMAXCUT, then the Hessian matrix of previous iteration is used instead of recomputed. This is especially important for VB groups with more than 10 electrons. However, if the region described by the VB wave function is non-classical (i.e. the Lewis structures are non-classical), one should use a more accurate Hessian by using a smaller threshold. To force full Hessian matrix evaluation for each iteration, one can set the threshold as 0.0. The default value is 0.025.

6.4.9 \$NOLOCVBO

By default, the maximum VBO localization will be performed at the last macro iteration of a CASVB calculation. This is designed to eliminate the intrinsic invariance of VBOs of CASVB space. In most cases, the CASVB calculation leads to well localized AO-like VBOs even without maximum localization. However, one can turn off the maximum localization of CASVB by adding the above control flag in the input. No extra data is needed for this control.

6.5 Initial VB orbital generation

6.5.1 \$##VBORB

This control flag provides a very flexible and intuitive way to specify a set of initial VB orbitals for a specific group. The ## is a two digit number of the group index. The general format of this control is as follows:

\$##VBORB
VBO-Specification-1
VBO-Specification-2
.....

Each VBO specification line specifies one or more VB orbitals, several lines can be used and the control is ended with a blank line. Each VBO specification can take one of the following forms:

- a) $NA \rightarrow NB$
- b) $NA-NB$
- c) $NA:(n)$
- d) $NA^\wedge(n)$
- e) $NA=NB$
- f) $NA\#NB$
- g) *Expression*

Where NA and NB are atom indexes. $NA \rightarrow NB$ stands for a sigma orbital on atom NA pointing to atom NB . $NA-NB$ stands for two orbitals forming a single bond between atom NA and NB . $NA:(n)$ stands for the n -th lone pair on atom NA . $NA^\wedge(n)$ stands for the n -th π orbital on atom NA . $NA=NB$ and $NA\#NB$ stands for 4 and 6 orbitals for double and triple bonds, respectively. For more details, please go to Chapter 7. The last case stands for general expression for initial VB orbitals. See more details in the following section.

6.5.2 General expression

The most general way to specify an initial VB orbital is to express it in a linear combination of atomic orbitals. VB2000 version 2.1 provides a simple and intuitive way for this purpose. The general expression of an orbital is shown as follows:

```
###VBORB
c1 AO n1 + c2 AO n2 + ... ci AO ni + ...
.....
```

where c_i is a coefficient for atomic orbital n_i , denoted as $AO n_i$. Generally speaking, there is no limitation for the number of terms in the expression, however, in practice, all expression must fit into one line for each VB orbital. A good example can be found in TESTINP/StressTest/h2cc6.inp:

```
$01VBORB
0.05 AO 1 +0.1 AO 2 +0.22 AO 3 +0.40 AO 4 +0.30 AO 5 +0.07 AO 18
0.05 AO 76 +0.1 AO 77 +0.22 AO 78 +0.40 AO 79 +0.30 AO 80 -0.07 AO 93
....
```

The order of the terms is not important. A possible scenario of using such expressions is that one already has a good initial guess of VB orbitals using whatever methods, and want to use this guess as an input. Since the expression is limited one orbital per line, you have to truncate the expression to only a few terms with the largest coefficients.

6.5.3 \$##PIVBO

Even though the control flag \$##VBORB can be used to specify a set of π orbitals, a more convenient way to specify a set of π orbitals is the PIVBO flag. The general format is as follows:

\$##PIVBO
nOrbitals
n1, n2, n3

where ## stands for a two digit number of the group index, *nOrbitals* is the number of π orbitals, and *n1, n2..* are the atom indexes on which the π orbitals are located. It is assumed that the atoms specified in this control form a conjugated plane (or approximately plane), and the orbital orientation of each π orbital is perpendicular to the plane of the atom and its connected conjugate atoms.

6.5.4 \$LMODISTORTION

This control specifies the distortion constant for splitting bonding LMOs. The general input format is as follows:

\$LMODISTORTION
SplitPara

Here *SplitPara* a splitting parameter (range 1.0 – 0.0, and the default value 1.0 gives the maximum split). Assume a bonding LMO localized between atom A and B, can be expressed as following

$$LMO(A-B) = \begin{pmatrix} CA1 \\ CA2 \\ \dots \\ CB1 \\ CB2 \end{pmatrix}$$

Where CA1, CA2 ... are the orbital coefficients on atom A and CB1, CB2 ... on atom B. With the default value, the LMO can be split into two VBOs, one is localized on atom A and the other on atom B as follows:-

$$VBO(A \rightarrow B) = \begin{pmatrix} CA1 \\ CA2 \\ \dots \\ 0 \\ 0 \end{pmatrix}$$

and

$$VBO(B \rightarrow A) = \begin{pmatrix} 0 \\ 0 \\ \dots \\ CB1 \\ CB2 \\ \dots \end{pmatrix}$$

Experiences show that the default value works fine in almost all cases. A small value of the splitting parameter means that the two generated VB orbitals are very close. A zero value means two VB orbitals are identical, and it may cause the VBSCF unstable.

6.5.5 \$WRITEGUESS

This control specifies a file name for writing an initial guess of the orbitals into a file. The input format is:

\$WRITEGUESS
FileName

where *FileName* is the file name for the generated initial guess of VB orbitals. The file has the following format:

Nbasis, Norbitals
((orbital(i,j),i=1,Nbasis), j=1,Norbitals)

where the first two integers are the number of basis functions and the number of orbitals, respectively, *orbital(i,j)* are AO coefficients of orbitals. The file is in free text format, and can be manually edited. The file can be read for the initial guess by VB2000. This can be useful in that one can modify some of the VBOs manually, and restart the computation by reading the file for the initial orbitals by using another control flag \$READGUESS. Note: all occupied orbitals (including both HF orbitals and VB orbitals) are exported.

6.5.6 \$READGUESS

This control can be considered as a complementary of \$WRITEGUESS. It controls the initial guess to be created from a disk file generated by a previous computation with control \$WRITEGUESS. Input format:

\$READGUESS
FileName

Here *FileName* is the file name of the initial guess of the orbitals. The file is a free format text file. Even though the automatically generated initial orbitals are normally quite robust, there are some cases where user-defined orbitals may be required. This option provides a convenient way of loading the initial guess, which may have been generated in an independent program, or even created manually.

6.5.7 \$RESTARTFILE

Each VB2000 calculation will generate a binary restart file, which stores all necessary information for a restart calculation. The file contains the 1D density matrix for each electron group and the VB orbitals. This can be useful in case of a not fully converged computation to start a new computation for the same molecule based on a previous computation. If the key word “RESTART” is specified in the input, then the program is expecting a restart file, and by default this restart file is the input file name with extension *.V84*. To restart the computation from a different file, one needs the control flag to specify the restart file in the following way:

\$RESTARTFILE
RestartFileName

Please note that the restart file can be used as a way to provide initial VB orbitals for a different calculation of the same molecule but with a different geometry and even different electrons and methods.

6.5.8 \$RESTARTMAPPING

The restart file not only allows a user to restart a computation with the same electron group partitioning, Version 1.8 also allows a restart of a computation with different electron group partitioning. To make this to work, another control, which tells how to correlate the electron group partitioning of a previous computation and the current one, is needed. The control is \$RESTARTMAPPING. Two or more VB groups in a previous calculation can be combined into one VB group in the second calculation, and the LMOs in a Hartree-Fock group can be re-assigned to a VB group in the second calculation. The general format is as following:

\$RESTARTMAPPING
NHFG(i), (i=1.... NHFX)
NGP(j), (j=1 NGPX)

where *NHFX* is the number of HF orbitals of previous calculation, and *NGPX* is the number of groups of previous calculation. *NHFG(i)* is the group assignment of the *i*-th HF orbital of previous calculation, and *NGP(j)* is the group assignment of the *j*-th group of the previous calculation. Please note that if *NGP(1)* is Hartree-Fock group, then all of these orbital assignments are controlled by *NHFG(1)*.

This can be explained by a two-step calculation on ClO₂. The first step is a GPF(3) calculation, with one Hartree-Fock group and two VB groups:

Example 6.3. Group function description of σ - π system of ClO_2

```

#! GPF(3)/D95 PRINTALL

TEST ClO2

0 2
17 .000000 .000000 .37
8 .0 1.260000 -.39
8 .0 -1.260000 -.39

$GENCTL
24,4,5
1, 2,4

$GRPDIM
12,4,3

$LMOGRPMODIFY
17
1,1
2,1
3,1
4,1
5,1
6,1
7,1
8,1
9,1
10,1
11,3
12,1
13,1
14,3
15,3
16,2
17,2

$03VBORB
1: (2)
2: (3)
3: (3)

$02VBORB
1->2
2->1
1->3
3->1

```

Group 2 has 4 electrons in 4 VB orbitals, corresponding to the two Cl-O bonds. Group 3 has 5 electrons in 3 VBOs. Each O has three LPs. The first LP on O is opposite to the Cl-O bond, the second LP is perpendicular to the Cl-O bond and in the O-Cl-O plane, and

the third LP is perpendicular to the O-Cl-O plane (also see the rules in Chapter 7). For group 3, the CASVB method (method No. 4) is used in order to keep the localization of VBOs as tight as possible. It is usually more efficient and more stable to divide a system into a number of subsystems.

In the next step, we want to perform a VB calculation by combining groups 2 and 3 of the previous calculation into one VB group with two more LPs from the two oxygen atoms. To do this, one needs to specify explicitly how to map the VB groups and LMOs of the previous calculation into the groups of a new calculation. This can be controlled by \$RESTARTMAPPING:

Example 6.4. Restart calculation for σ - π system of ClO_2

```

#! VB(13)/D95 RESTART CIONLY PRINTALL

TEST ClO2

0 2
17 .000000 .000000 .37
8 .0 1.260000 -.39
8 .0 -1.260000 -.39

$GENCTL
20,13
1, 2

$GRPDIM
10,9

$02VBSTR
9
1 1 2 2 3 4 5 6 8 8 9 9 7
1 1 2 2 3 4 5 6 7 7 8 8 9
1 1 2 2 3 4 5 6 7 7 9 9 8
1 1 2 6 3 4 5 5 8 8 9 9 7
2 2 1 4 5 6 3 3 8 8 9 9 7
1 1 2 6 3 4 5 5 7 7 8 8 9
2 2 1 4 5 6 3 3 7 7 9 9 8
1 4 2 6 3 3 5 5 8 8 9 9 7
1 2 4 6 3 3 5 5 8 8 9 9 7

$RESTARTMAPPING
1,1,1,1,1,1,1,1,1,1,2,2
1,2,2

$MACROITERATION
1

```

The first line specifies the new group assignment of the 12 LMOs of the Hartree-Fock group of the previous calculation. The first 10 still belong to group 1 (i.e. the H-F group),

and the last two are assigned to group 2 (i.e. the VB group). The second line is the group-wise assignment. Thus group 1 of the previous calculation is still assigned to the new group 1 (modified by the first line specification), and groups 2 and 3 are combined into the new group 2. A question is: how can we know that the orbital 11 and 12 of the first line are the ones we want to included in the VB group (the second group)? The orbitals 11 and 12 in the final printout of the previous calculation are identified to be the second LPs of the two oxygens by comparing the similarity of the starting LPs and the resulting LPs.

6.5.9 \$AOGROUP

Even though the optimal system wave function can be obtained from a reasonably good initial guess, there are cases where a classification of atomic orbitals can be helpful. A typical situation is the σ - π separation in a conjugated system. In such a case, the system can be divided into a σ group and a π group, and one may wish to avoid σ - π mixing during the optimization. To achieve this goal, a control is used to assign atomic orbitals into different groups, and group functions are then optimized within their corresponding subspaces. The input format is as follows:

```
$AOGROUP
NAO
NA1, NA2, ..... NANAO
GA1, GA2, ..... GANAO
```

where *NAO* is the number of atomic orbitals to be classified. The next line is the atomic orbital indexes, and following the atomic indexes are the group indexes *GA* of their corresponding atomic orbitals. With this control, the pure atomic orbitals are used as the initial orbitals for the group wave function.

6.6 Control for a chemical reaction

6.6.1 \$REACTION

This control is used to specify the energy profile calculation along a chemical reaction path. It is assumed that the reaction path has been obtained, or a reasonable guess has been made. The general input format is shown as follows:

```
$REACTION
IPOINT, NGEOM, NATOMS
GEOM1
atomic#1      x11      y11      z11
atomic#2      x12      y12      z12
.....
atomic#n      x1n      y1n      z1n
GEOM2
atomic#1      x21      y21      z21
atomic#2      x22      y22      z22
.....
atomic#n      x2n      y2n      z2n
.....
```

where *NGEOM* is the number of geometries of the reacting system along the reaction path, and *NATOMS* is the number of atoms in the reacting system. The atomic numbers are the nuclear numbers of atoms, *x*, *y* and *z* are the coordinates of atoms. *IPOINT* is the number of geometric interpolation points between each two consecutive structures along the reaction path. In a typical calculation, three geometries are used, include the geometries of reactants, the transition state, and the products.

6.7 Advanced controls

6.7.1 \$##ROOT

The number of the CI root for an excited state. In most cases, the ground state is calculated, thus this input can be omitted. The general format of this control is:

```
$##ROOT
NROOT
```

Where ## stands for the two digit number of an electron group, *NROOT* is the number of the CI root for an excited state, in ascending order. For instance, if *NROOT* is 2, the first excited state of group ## will be optimized. This is only meaningful when the electron group is described with multiple structure VB wave function (for instance, a CASVB

wave function).

This method only works well for the case where the excited state and the group state have different symmetry (i.e. symmetry vs. anti-symmetry) so that the excited state and the group state are always orthogonal, otherwise, the excited state may collapse into the ground state during the optimization.

6.7.2 \$\$\$STRSYMM

This control specifies the symmetry constraints on VB structure coefficients. The general format of the control is:

```
$$$SYMSYMM
NP
IA1, F1, IB1
IA2, F2, IB2
.....
```

where *NP* is the number of constraints, followed by the *NP* entries of the constraint specification. Each line specifies a symmetry related equality constraint as following:

<i>IA_i</i>	Index of structure coefficient
<i>F_i</i>	The symmetry factor, which can be either 1 or -1.
<i>IB_i</i>	Index of structure coefficient symmetrically related to IA

Assume the structure coefficients are denoted as *CSTR(i)*, where *i* is the index of the structure, then the constraint is

$$CSTR(IA_i) = F_i \times CSTR(IB_i)$$

6.7.3 \$MEMORY

This control is used to specify the size of dynamically allocated memory for the calculation. Following this \$ flag is the memory size in double-precision. For instance,

```
$MEMORY
10000000
```

The control will allow the program to allocate 10 MW memory during the computation. This is very necessary for the integral transformation of large basis sets where a large memory will speed up the transformation, otherwise the program will switch to the disk file mode for the integral transformation, and this will significantly increase the elapsed time of the computation. One can tell from the output file of a computation if the integral transformation is in the disk mode or not. If the disk mode is used, the output file also

tells how much more memory is needed for the in-core integral transformation. The default size of the scratch memory is $6.5MW$

In the GAMESS-VB2000 version, the vb2000 code checks the unused memory from that set in GAMESS. It then uses wither that or the value specified by the \$MEMORY flag, which ever is largest. Thus if more memory is required, this can be done by increasing the value set in the GAMESS \$SYSTEM block, or that specified by \$MEMORY.

6.7.4 \$VBOLIBGEN

This control is used for generating a VBO library for a new basis set; therefore, it may require a number of trial and error attempts. The general strategy of building a VBO library is to choose a set of simple molecules containing different types of bonds and atoms, and perform simple VB calculations which do not rely on a VBO library, and then store the VB orbitals from the calculation. The general format of the control is:

```
$VBOLIBGEN  
VBO-NAME  
NA NB  
NVBO  
VX VY VZ
```

where *VBO-NAME* is a string for the VBO name, and N_A and N_B are the two atoms defining the VBO orientation. The VBO is located on atom N_A and pointing to atom N_B . N_{VBO} is the orbital index of the VBO in the output from the VB calculation. For lone pairs or π VBOs, N_B is zero, and the orientation of the VBO is defined by a vector with components (V_X , V_Y and V_Z). Usually case, one does not know the VBO index before the calculation. Therefore, for creating a new VBO record, one has to do two calculations. Once the first calculation is done, one needs to inspect the VBOs in the final output, and find out the VBO index, then add the \$VBOLIBGEN control with the correct VBO index. If N_B is not zero, the VBO orientation is defined by the vector from atom N_A atom N_B , and the vector line is not required. Multiple VBO records can be created under this control. One can repeat the above inputs for each VBO record without any lines in between and the input ends with a blank line.

The VBO-label consists of two parts: a prefix and an extension. The search for a VBO in a library file is by the prefix, and the same VBOs for different basis sets are stored in different VBO library files. Each VBO library file is a collection of VBOs of the same basis set. Currently, the following VBO-Label prefixes are supported:

σ Orbitals

```
$SIGMA-H-  
$SIGMA-LI  
$SIGMA-BE  
$SIGMA-B-
```

\$SIGMA-N-
\$SIGMA-O-
\$SIGMA-F-
\$SIGMA-NA
\$SIGMA-MG
\$SIGMA-AL
\$SIGMA-SI
\$SIGMA-P-
\$SIGMA-S-
\$SIGMA-CL
\$SIGMA-BR
\$SIGMA-I-

π Orbitals

\$PI-H-IN-HH
\$PI-LI-IN-LIH
\$PI-BE-IN-H2BE
\$PI-B-IN-BH3
\$PI-C-IN-C2H4
\$PI-N-IN-CH2NH
\$PI-O-IN-CH2O
\$PI-F-IN-HF
\$PI-NA-IN-NAH
\$PI-MG-IN-MGH2
\$PI-AL-IN-ALH3
\$PI-SI-IN-SIH3
\$PI-P-IN-PH3
\$PI-S-IN-CH2S
\$PI-CL-IN-HCL
\$PI-BR-IN-HBR
\$PI-I-IN-HI
\$PI-N-IN-PYRROLE
\$PI-O-IN-FURAN
\$PI-S-IN-THIOPHENE

Lone Pairs

\$LP1-N-
\$LP1-O-
\$LP2-O-
\$LP1-F-
\$LP2-F-
\$LP1-P-


```
$LP1-S-  
$LP2-S-  
$LP1-CL  
$LP2-CL  
$LP1-BR  
$LP2-BR  
$LP1-I-  
$LP2-I-
```

To initialize a VBO by using a VBO library, the program searches for the exact match first. If no exact match can be found, then it looks for the most the similar one. For each VBO library, the minimal VBO entries are those for H, C, O and S for 1-3 period elements. The molecules and the utility scripts for creating the minimal VBO library files are released in this package in the subdirectory VBOLIBMOL.

6.7.5 Controls for canonicalization of LMOs

The VBO library is used in two ways: the canonicalization of LMOs and the generation of initial VB orbitals. If the library file exists, then one can use canonicalization option for triple bonds and atoms with multiple lone pairs so that the triple bond LMOs and lone pairs have the standard orientation. To enable canonicalization, you can use the following control flags.

```
$CANONLP  
$CANONPI
```

The first one is used for the canonicalization of lone pairs, and the second one is for the two π LMOs of a triple bond. For a triple bond such as that in C_2H_2 , the MO localization procedure does not guarantee the resulting two π LMOs are in the standard orientations.

6.7.6 \$PRINTHS

If you need VB matrix elements, you can add control flag \$PRINTHS in the input. With this control appearing in the input, both overlap and Hamiltonian matrix elements of VB structures will be printed. The normalization constants for VB structures are also printed. However, this option only prints matrix elements with four decimal digits. You need modify the code if you wish to get higher precision numbers.

6.8 Visualization of VB orbitals and molecules

Since version 1.8, we have introduced a number of routines to prepare data for a number of molecular visualization programs. In some cases there are different procedures in the three different programs - the stand-alone VB2000, the G98 version and the GAMESS(US) version. This is particularly true where the basis set has to be obtained and output. The options for visualization are as follows.

6.8.1 \$MOLPLT

This option is for the GAMESS(US) graphics program MOLPLT. The data created by this option can be used to display the molecule in an X window or to create an image in a postscript file for printing. In all three programs, inclusion of the keyword:-

\$MOLPLT

in the VB input data will create a file with the '.mol' extension to the jobname. In GAMESS(US), the MOLPLT data can be appended to the PUNCH file (jobname.dat) by adding 'MOLPLT=.TRUE.' to the \$CONTRL group. The inclusion of '\$MOLPLT' in the VB data, overrides this so the MOLPLT data is only in the jobname.mol file. In GAMESS(US) you might want to add a line in rungmts to move this file from the scratch directory to your current working directory. However, the file produced by GAMESS is more detailed than the file produced by VB2000, so the best advice is to use this directive in the stand-alone VB2000 program and use the 'MOLPLT=.TRUE.' command in \$CONTRL in the GAMESS(US)/VB2000 program. In G98, the file is also left on the scratch directory. To use the molplt program, you will need the manual, which is 'molplt.man' in the gamess/graphics directory (NB, this is not a man page; it is a straight text file).

6.8.2 \$PLTORB

This option is for the GAMESS(US) graphics program PLTORB. The data created by this option is used to display individual orbitals as contour plots in a specified plane. It requires two input files - jobname.orb and jobname.vec. In the GAMESS version, these can be created very easily. The VB2000 directive **\$PLTORB** in the VB data creates two files, jobname.orb and jobname.vec, and works in an identical way in the stand-alone, GAMESS(US) and G98 versions. The jobname.orb file is entirely different from the data, obtained by adding 'PLTORB=.TRUE.' to the GAMESS (US) \$CONTRL group, appended to the PUNCH file. The jobname.vec file contains the coefficients of the VB orbitals in the format that will be familiar to anyone who has used the PUNCH file and the MOREAD keyword in the \$GUESS group in Gamess(US). Unlike the pltorb output in the PUNCH file, which does require some editing to add the plane for printing the orbitals and so on, the one generated by VB2000 is ready to run, but may be improved by hand editing. To do that, you will need the manual, which is 'pltorb.man' in the

gamess/graphics directory (NB, this is not a man page; it is a straight text file). All distances are assumed to be Angstroms.

The data is:-

\$PLTORB

No. of bonds, followed by that number of atom index pairs, e.g. 2 2 1 3 1

No of VBOS to be plotted, followed by a list of their numbers, e.g. 4 4 5 6 7

SYMBOLIC or 3ATOMS - only these two methods are supported.

If SYMBOLIC, one of the strings XY, YX, XZ, ZX, YZ, ZY to choose the orientation of the plotting axes, or if 3ATOMS a list of three atom indices to define the plotting plane. The first will be set as the origin. E.g "YZ" if SYMBOLIC or "1 3 1" if 3ATOMS.

And offset in Angstroms for the plotting plane relative to the defined plane, e. g. 0.0

The x and y limits for the plotting areas, e. g. -2.0 2.0 -2.0 2.0

A title up to 40 characters for each VBO plot. E.g. for 4 plots:-

Water VBO 4

Water VBO 5

Water VBO 6

Water VBO 7

6.8.3 \$CUBE

Orbitals can be displayed from a standard Gaussian cube file, a format that can be read by several visualization programs, including VMD and Molekel.

The data block:-

\$CUBE

M

(N(i), i=1, M)

Title

NP

will produce *M* output files, called jobname- $\{N(i)\}$.cube. *M* is the number of orbitals to be created and *N(i)* is an array of the orbital numbers. E.g.

\$CUBE

3

15 16 17

Molecule X

64

will generate 3 files, called jobname-015.cube, jobname-016.cube and jobname-017.cube

containing the cube data for the VB orbitals, 15, 16 and 17. *NP* is the number of grid points in each Cartesian coordinates. *NP* has a maximum of 100 and defaults to 80 if given as 0.

Note that cube files generated by Gaussian itself contain many orbitals, but not all visualization codes can read them. Some can only read one orbital, so VB orbitals are created in separate cube files. Note also that the current code is not very efficient and can be slow. Do not create cube files unless you really need them. The \$CUBE directive can be used in all three versions of VB2000.

6.8.4 \$GENGRID or \$GRID

The grid files can be generated in very similar way as above. The input is almost identical except the flag \$CUBE is replaced by \$GENGRID or \$GRID. Note that \$GRID can not be used in GAMESS(US)/VB2000 as it conflicts with a GAMESS data directive. The grid files generated as .grd as the file extension. The grid files can be visualized by DS Visualizer, which is free from Accelrys. One can generate both cube and grid files by using a shortcut as follows:

\$CUBE\$GENGRID

M

(N(i), i=1,M)

Title

NP

\$GENGRID\$CUBE

can be used as an alternative. The prohibition of \$GRID in GAMESS-VB2000 still holds here. *\$CUBE\$GRID* or *\$GRID\$CUBE* can be used likewise.

6.8.5 \$XYZFILE

To create an XYZ file for the molecule one can use the above directive in the input. A file with file name jobname.xyz will be created. The xyz file can be read by many visualization packages, such as VMD:

VMD (Visual Molecular Dynamics) <http://www.ks.uiuc.edu/Research/vmd/>.

VMD is a freely available and very powerful toolkit for visualization and analysis of molecular dynamics (MD) simulation data. Although it was initially developed with focus on MD, the newer versions contain many features that make it a very attractive choice for visualizing results of VB2000. One can find the tutorial of VMD at

<http://www.theochem.ruhr-uni-bochum.de/~axel.kohlmeyer/cpmd-vmd/index.html>

Here are a few tips for how to use VMD for displaying a VB orbital:

1. Load the molecule (.xyz file) into the UI.
2. Add the cube file of the molecule orbital to the molecule.
3. Use the Graphics-> Representations menu from the VMD main Windows and choose Volume as the coloring method and isosurface as the drawing method. You'll just have to choose the appropriate isovalue for the orbital you want to see. You can also make orbital transparent.

One can also use another free software **Molekel** <http://www.cscs.ch/molekel/> to display a Gaussian cube file. Molekel accepts the so-called .mkl file for molecule structure. You may need to use of the directive:-

6.8.6 \$MOLEKEL

The directive in the VB data block generates a file called jobname.mkl (in the GAMESS and G98 cases in the scratch area). This is the molecule input file for **Molekel**.

Tested with version 5.4 - Ugo Varetto, MOLEKEL 5.4; Swiss National Supercomputing Centre: Manno (Switzerland).

6.8.7 \$MOLDEN

The directive in the VB data block generates a file called jobname.molf (in the GAMESS and G98 cases in the scratch area). This is the molecule input file for **Molden**.

6.8.8 Using the visualization directives in TEST runs.

The TEST directive on the command line terminates the program after generation of the initial orbitals. The program now calls the visualisation routines at this point prior to termination of the run, but uses the localised MOs rather than the VB orbitals when the visualization routine requires orbitals. All the above visualization directives can be used. Note that the numbering of the LMOs will be different from the numbering of the VBOs that you will finally want to use to obtain CUBE files.

6.8.9 3D pictures of two VB orbitals of a O-H bond

Here are two orbitals forming an O-H bond in the water molecule. The pictures are created with VMD [20].

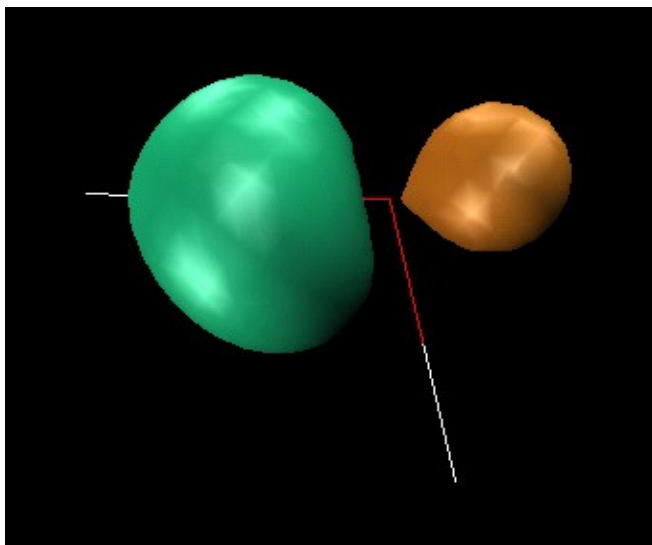


Figure 1: VB orbital localized on oxygen atom.

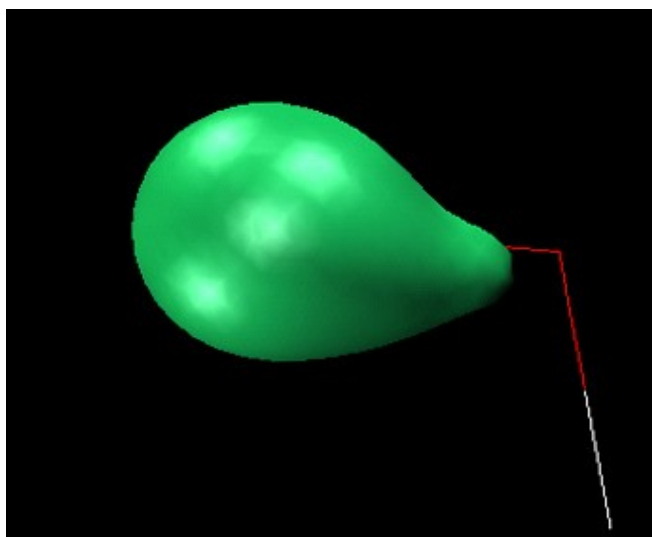


Figure 2: VB orbital localized on hydrogen atom.

7 Construction of Initial Wave Functions

The initial guess of the system wave function of a molecule in the framework of GF theory is essential for the GFT calculation. It involves the partitioning of electron groups and the construction of initial orbitals for each group function. A good initial guess of VB orbitals is essential for VB group function optimization. Construction of high quality initial VB orbitals according to a user's intention is especially challenging. To solve the problem, two technologies have been introduced in VB2000 version 1.8. One involves the standard notation for LMOs and VBOs and the other depends on the standard VBO libraries. This chapter describes the standard notations for different types of LMOs and VBOs, and how to use those notations to specify the electron group partitioning and the construction of initial VB orbitals.

7.1 Notations for localized molecular orbitals

Owing to the locality and transferability of localized molecular orbitals and valence bond orbitals, the orbitals preserve certain characteristics in different molecular environments. Therefore it should be possible to create a set of standard notations for different types of orbitals. From a Hartree-Fock wave function, the molecular orbital localization method generates the following types of LMOs: the inner core orbitals, lone pairs and the bonding orbitals. The following notations are used to label different LMOs.

7.1.1 Lone pair

A lone pair orbital can be denoted by

$$A: (m)$$

where A is the atom index in the input, m is the index of the lone pair, which roughly follows the order of orbital energies. For instance, the lowest lone pair of the valence shell is labeled $A:(1)$, and the higher energy lone pairs (if any) are labeled $A:(2)$ and $A:(3)$. Follow a similar convention, all inner core orbitals on atom A are thus labeled as

$$A: (0)$$

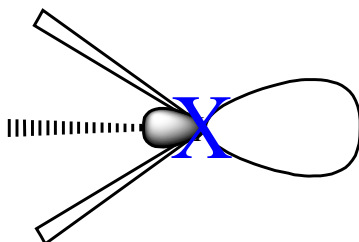
Let's use H₂O as an example. Assume the atoms are in the order as O, H and H in the input. The first lone pair (LP) on oxygen (which should be the sigma lone-pair and is in the plane of the molecule) can be denoted by

$$1: (1)$$

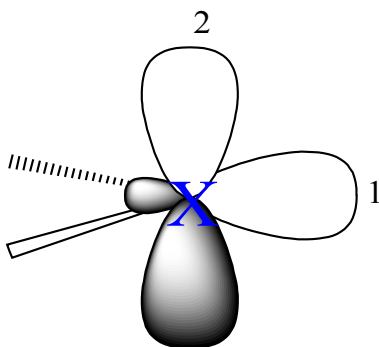
The second LP, which is actually the lone-pair perpendicular to the molecule plane, can

be denoted as **1: (2)**. For multiple LPs on an atom, rules are used to label different LMOs.

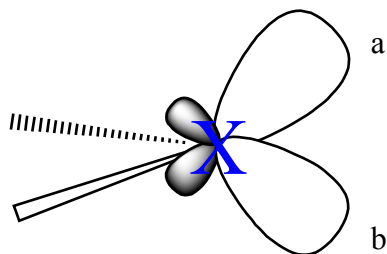
Case 1. XR_3 Type, for example, NH_3 . Only one LP exists. The LP is shown as follows:



Case 2. XR_2 Type. Example, H_2O . There are two LPs. The first one is defined as lying in the molecular plane, but pointing symmetrically away from the two bonds; the second LP direction is perpendicular to the R-X-R plane, as shown below



As shown in the above scheme, LP1 has σ symmetry, and LP2 has π symmetry. Often one can use another way of describing the two LPs, taking them as 'equivalent orbitals', shown as follows:

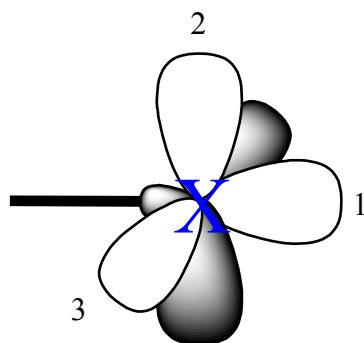


In VB2000, only the primitive σ - π LPs are defined automatically. However, there is a simple way to construct the two equivalent LPs, using the following linear combination of the primitive LPs:

$$\begin{aligned}LP(a) &= LP(1) + LP(2) \\LP(b) &= LP(1) - LP(2)\end{aligned}$$

The normalization coefficients are ignored. It is assumed that the LP(1) and LP(2) are normalized. The current version of VB2000 provides a simple way to construct initial orbitals for the above combinations.

Case 3. *XR* Type. For example, Ph-F.



The first LP is defined as the one opposite to the X-R bond. The remaining two LP directions are perpendicular to the X-R bond direction. If R is a conjugated atom and has a well defined conjugate plane, then the second LP is parallel to the conjugate plane, and the third one is perpendicular to the conjugate plane. For instance in the case of Ph-F, the first LP is the one opposite to the Ph-F bond, the second one is parallel to the phenyl ring, and the third one is perpendicular to the phenyl ring.

7.1.2 Two-center bond LMO

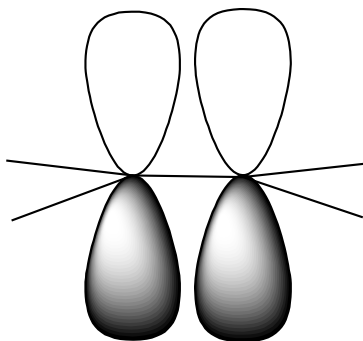
Each local bond(between two atoms) can be denoted by

$$\mathbf{A-B}(k)$$

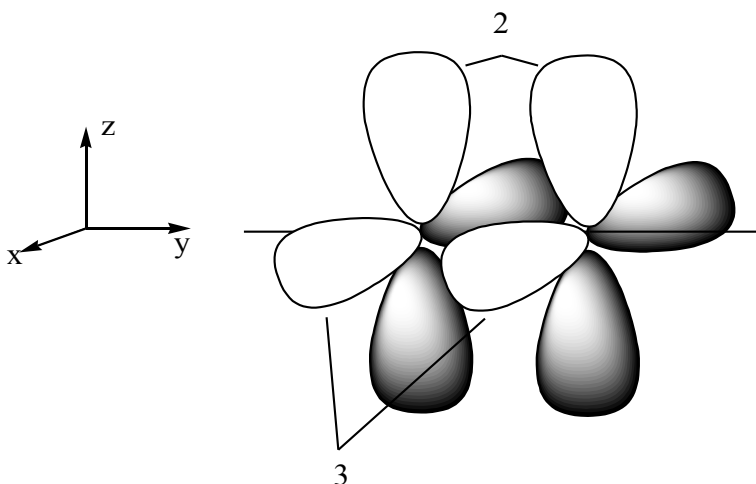
where **A** and **B** are two bonded atom, and **k** is the index of the bond. The first bond **A-B(1)**, or simplified as **A-B**, is the sigma bond as shown in the following scheme:



The second bond (if any) is the first π bond. For a double bond, there is no ambiguity for the π bond as shown in the following scheme:



For a triple bond, the third bonding LMO is the second π MO. There are two cases: if the triple bond atoms are co-linear with a conjugate neighbor, then the second LMO is a π orbital and is defined with direction perpendicular to the neighbor conjugate plane. If there is no such plane, then the π orbital orientation is not well defined. If we put the co-linear atoms on the Y -axis (as shown below), and take π orbital on the Z -axis as the second LMO, then the π orbital on the X -axis is the third LMO. If the triple bond atoms are not co-linear with a third neighbor atom, the π orbitals perpendicular to the plane (defined by the three non-co-linear atoms) is the second LMO, and the other one is the third:



For instance, if we have the molecule C_2H_2 on the Y -axis as shown above, then the first bond is the σ bond (not shown in the above scheme), and the second bond (i.e. the first π bond) is in P_z direction, and the third bond (i.e. the second π bond) is the P_x direction. If we align the two triple bonded atoms on the Z -axis, the first and the second π orbitals are in the X - and Y -axis, respectively. The third case is that the two atoms are on the X -axis, and the first and the second π orbitals are on the Y - and Z -axis, respectively. For an arbitrary orientation of a triple bond, we need define three new axes in the molecule frameworks, and label the three axes according the maximum overlap (absolute values) of the orientation of each axis with the direction of the original axes. Once the axes in the molecule framework are so labeled, define the first and second π orbital orientations in the same way as above. However, if the triple bond is connected to a conjugate system, for instance, Ph-CN, the first π orbital direction is chosen to be the that perpendicular to the conjugate plane. In Ph-CN, the first π is perpendicular to the phenyl ring.

There are special cases that the triple bond atoms are not co-linear, for instance, R-Si#Si-R triple bond. The four atoms R-Si-Si-R are not co-linear. In such a case, the P_z is defined as the direction perpendicular to the R-Si-Si plane.

7.1.3 Multi-center bond LMO

A multi-center (MC) bond LMO can be denoted by

$$MC(k)$$

where k is the index of the MC bond. Thus the first MC bond LMO (i.e the LMO with the lowest energy) is marked as $MC(1)$, the second as $MC(2)$, etc.

7.2 Notations for VB Orbitals

7.2.1 σ VB orbital

A σ VB orbital can be denoted as



where A is the atom that the VBO belongs to, and B is the atom that this VBO bonds to. In other words, this is a VBO on atom A pointing to atom B , and can be schematically represented as following:



Since bonding VB orbitals always appear in pairs, we can use $A-B$ to represent two

bonding VBOs forming a single bond between atom *A* and *B*:



7.2.2 π VB orbital

Since a π VBO is not necessary pointing to a particular atom, the following notation is introduced:

$$n^{\wedge}(k)$$

where *n* is the VBO atom, the hat ' \wedge ' indicates a π orbital, and *k* is the index of the π VBO. For instance, in C_2H_2 , there are two π VBOs on each carbon. Special rules are used to define the first- and second- π bond directions. See previous sections for more details.

In most simple calculations, the initial VB orbitals are generated automatically by splitting the corresponding bonding LMOs. However, there are cases where you need to explicitly specify a set of VB orbitals for your calculation. The following example explains how the above notations and rules can be used to specify the VB orbitals for a CASVB(6,5) calculation, which involved 5 VB orbitals for 6 electrons.

Example 7.1. Explicit specification of VB orbitals in H_2O

```
#! CASVB(6,5)/D95 PRINTALL

TEST H2O

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582

$02VBORB
1->2
2->1
1->3
3->1
1: (2)
```

In this example, 5 VB orbitals are explicitly specified in input group \$02VBORB. Each line specifies one VBO. The input is terminated by a blank line. The first 4 orbitals are used for forming two O-H σ -bonds, while the last one is the second LP (perpendicular to

the HOH plane).

Not only the above primitive (σ and π) orbitals can be specified, any orbitals which can be expressed as a linear combination of those primitives can also be specified in a quite intuitive format. See section 7.3.3 for more details.

7.2.3 Multiple VBOs

Since each bond involves two VBOs, it is intuitive to use a bond notation to specify two VBOs. Thus

$n-m$ (single bond)

is equal to

$n \rightarrow m$
 $m \rightarrow n$

and

$n=m$ (double bond)

is equal to

$n \rightarrow m$
 $m \rightarrow n$
 n^{\wedge}
 m^{\wedge}

and

$n\#m$ (triple bond)

is equal to

$n \rightarrow m$
 $m \rightarrow n$
 $n^{\wedge}(1)$
 $m^{\wedge}(1)$
 $n^{\wedge}(2)$
 $m^{\wedge}(2)$

7.3 Construction of initial VB orbitals

7.3.1 Initial VBOs by splitting LMOs

This is the default method for generating initial VB orbitals if no VBO directives are provided in the input. The technique is based on a very simple and intuitive idea: chemical structures in the sense of qualitative valence bond theory are associated with alternative allocations of electron-pair bonds, which are easily translated into the language of modern VB theory. Each 'structure' becomes a spin-coupling scheme, involving pairs of electrons occupying rather localized, strongly overlapping, orbitals. The overlapping orbitals of any bond are expected to bear some similarity to the AOs, or hybrids, used in constructing a localized MO. A simple and natural way to generate such an orbital pair is to generate a set of LMOs (e.g. by transformation of a simple MO-type wave function) and then to cut each LMO into two pieces (also see section 6.5.2), as described very briefly in reference[1]. In this way, one can construct initial VB orbitals for all groups; and by orthogonalization, using Löwdin's method, it is easy to set up active subspaces satisfying the strong orthogonality constraints of the GF method. This first approximation is used for the sole purpose of starting the calculation and the group functions are allowed to be optimized in the full basis function space within the strong orthogonality constraints. Our experience shows that this method works very well for simple covalent bonds and the generated initial VB wave function has quite good convergence behavior. Moreover, the method is very general, it can be used for any kind of basis set and any simple covalent bonds which can be represented by a bonding LMO.

7.3.2 Initial VBOs from VBO libraries

For most simple calculations, the above default method works very well. However, there are cases in which the above method does not work well, especially for special bonding situations. Examples are molecules with conjugated systems or multi-center bonds; or special bond types (e.g. 'banana bonds'). In VB2000 version 1.8, the standard notation is used to specify the initial VBOs. Example 7.1 in section 7.2.2 explains how the standard VBO notations are used in the input file. The VBO directives in the input file are used by the program to load the standard VBOs from a library file provided in the package. In VB2000 version 1.8, 14 VBO library files are included, each for one or more basis sets. A user can generate a VBO library file for a user-specified basis set by using a utility provided in the package. See Chapter 6 for more details.

7.3.3 Linear combination of standard VBOs

In version 1.8, a user can use a linear combination of library VBOs to form a new VBO with specific shape and orientation. The formation of this linear combination is very simple and intuitive as shown in the following example.

Example 7.2. Specification of an arbitrary combination of the primitive σ and π orbitals

```

#! VB(6)/D95 PRINTALL

TEST C2H2 TRIPLE BOND (3 BANANA BONDS)

0 1
6   .528275  0.305  0.0
6  -.528275 -0.305  0.0
1  1.463583  0.845  0.0
1 -1.463583 -0.845  0.0

$02VBORB
(1->2) + 1^
(2->1) + 2^
(1->2) - 0.5 (1^) + 0.866025 (1^(2))
(2->1) - 0.5 (2^) + 0.866025 (2^(2))
(1->2) - 0.5 (1^) - 0.866025 (1^(2))
(2->1) - 0.5 (2^) - 0.866025 (2^(2))

```

In this example, we want to perform a three banana bond calculation for the CC triple bond. Instead of using one σ and two π VBOs on each carbon, one uses three symmetrically equivalent SP^3 hybrid orbitals. Since the program uses the primitive σ and π VBOs, we need to recombine σ and π VBOs to obtain 3 symmetry-equal VBOs. Each of the first VBOs has two components, and the last 4 VBOs have 3 components for each. The coefficients are chosen in such a way that all the three VBOs on each carbon are symmetrically equivalent.

7.3.4 A method for construction of the CASSCF space

Although both VBO library and the LMO-splitting methods are used to generate initial VB orbitals, the methods can be used for more general purposes such as the construction for the active space for a CASSCF wave function. It is often a big challenge to construct a proper active space for a CASSCF calculation. The tools in VB2000 will make the CAS-type calculation more flexible and easy to control.

8 Test Cases and Output Descriptions

In this chapter, input and output files are extensively listed for all the examples discussed. For the sake of clarity, only the most relevant parts of the output files are shown; but the full output from each test calculation can be found online. The URL for each file is given. For the electronic version, one can open the log file with a double-click on the appropriate hyper-link.

8.1 Simple VB calculation on H₂O

Input:

```
#! VB(4)/cc-pVDZ PRINTALL

TEST H2O

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582
```

For illustration purpose, the full text of the output file of Example 1 is given below:

```
*****
*
*                               V B 2 0 0 0
*
*   An ab initio Valence-Bond Program Based on the
*   Generalized Product Function Method and
*   the Algebrant Algorithm
*
*   Version 2.5   September   2010
*
*   Jiabo Li*, Brian Duke** and Roy McWeeny***
*
*   * SciNet Technologies, 9943 Fieldthorn St., San Diego
*   CA 92127, USA
*
*   ** Monash University Institute of Pharmaceutical Sciences,
*   381 Royal Pde, Parkville, Victoria, 3052, Australia
*
*   *** Department of Chemistry, University of Pisa,
*   56100 Pisa, Italy
*
*   Reference for the software:
*   Jiabo Li, Brian Duke, and Roy McWeeny, VB2000 Version
*
```



```

*      2.5, SciNet Technologies, San Diego, CA, 2010      *
*      URL: http://www.scinetec.com or http://www.vb2000.net *
*
*      Reference for the theory:
*      Jiabo Li, and Roy McWeeny, "VB2000: Pushing Valence
*      Bond Theory to New Limits", Int. J. Quantum Chem.,
*      89(2002)208-216.
*
*      Copyright (C) 2000-2010 by SciNet Technologies
*****

```

JOB NAME IS h2o

VB2000 DIRECTORY IS: /home/jli/VB2000
 MAXIMUM DYNAMIC MEMORY IS: 6500000

Input of molecule

=====

#! VB(4)/CC-PVDZ SPHER PRINTALL

Title: TEST H2O

Charge = 0 Multiplicity = 1

BASIS SET FILE IS: /home/jli/VB2000/BASET/cc-pVDZ

Cartesian coordinates of atoms in Angstrom

```

-----
              X              Y              Z
O      8.0      0.000000      0.000000      0.000000
H      1.0      0.801842      0.000000      0.555582
H      1.0     -0.801842      0.000000      0.555582
-----

```

Total number of two-electron integrals = 25620
 Time for integral evaluation 0.203

SKIPPED INTEGRALS= 23683
 Number of iter = 20, DMAX = 0.000008547425
 Final Hartree-Fock Energy = -76.024351732389

Molecular orbitals by HF method

```

=====
MO#              1              2              3              4              5              6
E(a.u.)         -20.551      -1.323      -0.700      -0.551      -0.490      0.181
-----
1  1  O      s      -1.00098  -0.00802   0.00000  -0.00238   0.00000  -0.05679
2  1  O      s      -0.00206   0.44344   0.00000   0.14454   0.00000   0.08478
3  1  O      s       0.00114   0.38746   0.00000   0.34076   0.00000   1.06437
4  1  O      x       0.00000   0.00000   0.48446  -0.00000   0.00000   0.00000
5  1  O      y      -0.00000  -0.00000  -0.00000   0.00000   0.63106  -0.00000
6  1  O      z      -0.00154   0.07079  -0.00000  -0.55171   0.00000   0.18904
7  1  O      x       0.00000   0.00000   0.21715  -0.00000   0.00000   0.00000
8  1  O      y      -0.00000  -0.00000  -0.00000   0.00000   0.49762   0.00000
9  1  O      z       0.00044  -0.00877  -0.00000  -0.38237   0.00000   0.33395
10 1  O     xx       0.00108   0.00578   0.00000  -0.00300   0.00000  -0.02660
11 1  O     yy       0.00119  -0.00026   0.00000   0.00596   0.00000  -0.04026

```

12	1	O	zz	0.00116	0.00354	0.00000	-0.02384	-0.00000	-0.02445
13	1	O	xy	-0.00000	-0.00000	-0.00000	-0.00000	-0.00000	-0.00000
14	1	O	xz	0.00000	0.00000	0.02503	-0.00000	0.00000	0.00000
15	1	O	yz	-0.00000	-0.00000	-0.00000	-0.00000	0.01730	-0.00000
16	2	H	s	0.00009	0.18307	0.32775	-0.19665	0.00000	-0.07225
17	2	H	s	-0.00022	0.01186	0.08897	-0.04666	0.00000	-0.83420
18	2	H	x	-0.00038	-0.03671	-0.02612	0.02896	-0.00000	0.01232
19	2	H	y	0.00000	0.00000	0.00000	0.00000	0.02997	0.00000
20	2	H	z	-0.00029	-0.01731	-0.03100	-0.00989	-0.00000	0.01179
21	3	H	s	0.00009	0.18307	-0.32775	-0.19665	-0.00000	-0.07225
22	3	H	s	-0.00022	0.01186	-0.08897	-0.04666	-0.00000	-0.83420
23	3	H	x	0.00038	0.03671	-0.02612	-0.02896	-0.00000	-0.01232
24	3	H	y	0.00000	0.00000	-0.00000	0.00000	0.02997	0.00000
25	3	H	z	-0.00029	-0.01731	0.03100	-0.00989	0.00000	0.01179

MO#	7	8	9	10
E(a.u.)	0.253	0.798	0.810	1.159

1	1	O	s	0.00000	0.00000	-0.05691	-0.01233
2	1	O	s	0.00000	0.00000	-0.23839	-0.09812
3	1	O	s	-0.00000	-0.00000	0.23123	-0.57172
4	1	O	x	0.28554	-0.29362	-0.00000	-0.00000
5	1	O	y	-0.00000	-0.00000	-0.00000	-0.00000
6	1	O	z	-0.00000	0.00000	-0.27959	0.78898
7	1	O	x	0.67781	-0.41880	-0.00000	0.00000
8	1	O	y	0.00000	0.00000	0.00000	0.00000
9	1	O	z	-0.00000	0.00000	-0.06517	-1.21139
10	1	O	xx	-0.00000	-0.00000	0.09947	-0.04330
11	1	O	yy	0.00000	0.00000	-0.08383	-0.03242
12	1	O	zz	-0.00000	0.00000	-0.01043	-0.05273
13	1	O	xy	-0.00000	-0.00000	-0.00000	-0.00000
14	1	O	xz	0.02022	0.11036	0.00000	0.00000
15	1	O	yz	-0.00000	-0.00000	-0.00000	-0.00000
16	2	H	s	-0.02568	0.93900	0.83714	0.44132
17	2	H	s	-1.38415	-0.69185	-0.55338	0.11867
18	2	H	x	0.02058	0.07197	0.26542	-0.12181
19	2	H	y	0.00000	0.00000	0.00000	0.00000
20	2	H	z	0.01550	0.16306	0.02719	-0.24133
21	3	H	s	0.02568	-0.93900	0.83714	0.44132
22	3	H	s	1.38415	0.69185	-0.55338	0.11867
23	3	H	x	0.02058	0.07197	-0.26542	0.12181
24	3	H	y	0.00000	0.00000	-0.00000	0.00000
25	3	H	z	-0.01550	-0.16306	0.02719	-0.24133

VBO LIB IS: /home/jli/VB2000/VBOLIB/VBOLIBCCPVDZ

Localized molecular orbitals and the bond labels

LMO#	1	2	3	4	5
E(a.u.)	-19.408	-1.909	-0.490	-0.905	-0.905
Bond Label	1:(0)	1:(1)	1:(2)	3-1(1)	2-1(1)

1	1	O	s	-0.97313	0.22947	0.00001	0.03465	0.03465
2	1	O	s	0.10175	0.36557	-0.00001	0.19176	0.19176
3	1	O	s	0.12585	0.48820	-0.00001	0.07767	0.07768
4	1	O	x	-0.00000	-0.00000	0.00000	-0.34257	0.34257
5	1	O	y	0.00001	0.00001	0.63106	0.00000	0.00000
6	1	O	z	-0.07631	-0.39555	0.00000	0.27121	0.27121

7	1	O	x	-0.00000	-0.00000	0.00000	-0.15355	0.15355
8	1	O	y	0.00001	0.00001	0.49762	0.00000	0.00000
9	1	O	z	-0.06196	-0.30721	0.00000	0.15503	0.15503
10	1	O	xx	0.00162	0.00064	-0.00000	0.00450	0.00450
11	1	O	yy	0.00205	0.00428	-0.00000	-0.00268	-0.00268
12	1	O	zz	-0.00202	-0.01711	0.00000	0.01195	0.01195
13	1	O	xy	-0.00000	-0.00000	-0.00000	0.00000	-0.00000
14	1	O	xz	-0.00000	-0.00000	0.00000	-0.01770	0.01770
15	1	O	yz	0.00000	0.00000	0.01730	0.00000	0.00000
16	2	H	s	0.00218	-0.05184	0.00000	-0.04535	0.41816
17	2	H	s	-0.00547	-0.03011	0.00000	-0.03663	0.08919
18	2	H	x	-0.00245	0.00220	0.00000	-0.01451	-0.05145
19	2	H	y	0.00000	0.00000	0.02997	0.00000	0.00000
20	2	H	z	-0.00501	-0.01754	0.00000	0.01623	-0.02761
21	3	H	s	0.00218	-0.05184	0.00000	0.41816	-0.04535
22	3	H	s	-0.00547	-0.03011	0.00000	0.08919	-0.03663
23	3	H	x	0.00245	-0.00220	-0.00000	0.05145	0.01451
24	3	H	y	0.00000	0.00000	0.02997	0.00000	0.00000
25	3	H	z	-0.00501	-0.01754	0.00000	-0.02761	0.01623

Mulliken population of localized molecular orbitals

LMO#	1	2	3	4	5
E (a.u.)	-19.408	-1.909	-0.490	-0.905	-0.905
Bond Label	1:(0)	1:(1)	1:(2)	3-1(1)	2-1(1)

1	O	2.00065	2.02871	1.95912	1.17571	1.17571
2	H	-0.00032	-0.01435	0.02044	-0.00722	0.83151
3	H	-0.00032	-0.01435	0.02044	0.83151	-0.00722

PARTITIONING OF LMOs INTO GROUPS (\$LMOGRP)

LMO#	1	2	3	4	5
Group#	1	1	1	2	2
Split	0	0	0	1	1

WARNING: Initial orbital 8 undefined from DISLMO
Initial orbital

Initial guess of orbitals

ORBITAL#	1	2	3	4	5	6			
Bond Label	1:(0)	1:(1)	1:(2)	3-1(1)	3-1(1)	2-1(1)			
1	1	O	s	-0.97306	0.22901	0.00001	0.07646	-0.03150	0.07646
2	1	O	s	0.10223	0.36201	-0.00001	0.34122	-0.05369	0.34122
3	1	O	s	0.12662	0.48258	-0.00001	0.17471	-0.07155	0.17471
4	1	O	x	-0.00000	-0.00000	0.00000	-0.54231	0.00000	0.54231
5	1	O	y	0.00001	0.00001	0.63106	0.00000	0.00000	0.00000
6	1	O	z	-0.07637	-0.39512	0.00000	0.39441	0.05852	0.39441
7	1	O	x	-0.00000	-0.00000	0.00000	-0.24308	0.00000	0.24308
8	1	O	y	0.00001	0.00001	0.49762	0.00000	0.00000	0.00000
9	1	O	z	-0.06192	-0.30750	0.00000	0.21934	0.04555	0.21934
10	1	O	xx	0.00104	0.00492	-0.00000	-0.00010	-0.00073	-0.00010
11	1	O	yy	0.00152	0.00816	-0.00000	-0.01049	-0.00121	-0.01049

12	1	O	zz	-0.00257	-0.01308	0.00000	0.01059	0.00194	0.01059
13	1	O	xy	-0.00000	-0.00000	-0.00000	0.00000	0.00000	-0.00000
14	1	O	xz	-0.00000	-0.00000	0.00000	-0.02802	0.00000	0.02802
15	1	O	yz	0.00000	0.00000	0.01730	0.00000	0.00000	0.00000
16	2	H	s	0.00221	-0.05212	0.00000	-0.00402	0.00769	-0.00402
17	2	H	s	-0.00557	-0.02935	0.00000	-0.00389	0.00435	-0.00389
18	2	H	x	-0.00252	0.00276	0.00000	-0.00075	-0.00040	-0.00075
19	2	H	y	0.00000	0.00000	0.02997	-0.00000	0.00000	-0.00000
20	2	H	z	-0.00507	-0.01708	0.00000	-0.00230	0.00253	-0.00230
21	3	H	s	0.00221	-0.05212	0.00000	-0.00402	0.87539	-0.00402
22	3	H	s	-0.00557	-0.02935	0.00000	-0.00389	0.18942	-0.00389
23	3	H	x	0.00252	-0.00276	-0.00000	0.00075	0.10716	0.00075
24	3	H	y	0.00000	0.00000	0.02997	-0.00000	0.00000	-0.00000
25	3	H	z	-0.00507	-0.01708	0.00000	-0.00230	-0.05477	-0.00230

ORBITAL#		7	8
Bond Label		2-1 (1)	

1	1	O	s	-0.03150	-0.16945
2	1	O	s	-0.05369	-1.32291
3	1	O	s	-0.07155	-2.08434
4	1	O	x	0.00000	0.00000
5	1	O	y	0.00000	-0.00000
6	1	O	z	0.05852	0.15963
7	1	O	x	0.00000	-0.00000
8	1	O	y	0.00000	-0.00000
9	1	O	z	0.04555	-0.10632
10	1	O	xx	-0.00073	1.58788
11	1	O	yy	-0.00121	1.44053
12	1	O	zz	0.00194	1.49352
13	1	O	xy	0.00000	-0.00000
14	1	O	xz	0.00000	-0.00000
15	1	O	yz	0.00000	0.00000
16	2	H	s	0.87539	-0.10395
17	2	H	s	0.18942	0.28012
18	2	H	x	-0.10716	0.20557
19	2	H	y	0.00000	0.00000
20	2	H	z	-0.05477	0.17180
21	3	H	s	0.00769	-0.10395
22	3	H	s	0.00435	0.28012
23	3	H	x	0.00040	-0.20557
24	3	H	y	0.00000	0.00000
25	3	H	z	0.00253	0.17180

GENERAL CONTROLS (\$GENCTL)

```

=====
Number of electron groups      =      3
Maximum macro-iterations      =     12
Energy threshold               = 10**(- 6)
Restart calculation(0/1/2)    =      0
Group#                         1  2  3
Num. of electrons             6  4  0
Num. of spins                 0  0  0
Num. of orbitals              3  4  1
Method#                       1  2  99

```

SYMBOLIC VB STRUCTURE(S) OF GROUP 2
STR. NO. RUMER PATTERN

```

1          1  2  3  4

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.150000000000000
GROUP =  2  ITER =  2 CMAX = 0.10039609465417
GROUP =  2  ITER =  3 CMAX = 0.05477863461951
GROUP =  2  ITER =  4 CMAX = 0.02156047016666
GROUP =  2  ITER =  5 CMAX = 0.00478066388963
GROUP =  2  ITER =  6 CMAX = 0.00207064621826
GROUP =  2  ITER =  7 CMAX = 0.00087610675346

ENERGY AND DIFF OF MACROITER    1 =    -76.05664155    -76.05664155

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.01550916155391
GROUP =  2  ITER =  2 CMAX = 0.01286002764750
GROUP =  2  ITER =  3 CMAX = 0.01160428524617
GROUP =  2  ITER =  4 CMAX = 0.00955608995148
GROUP =  2  ITER =  5 CMAX = 0.00405275365966
GROUP =  2  ITER =  6 CMAX = 0.00160204273130
GROUP =  2  ITER =  7 CMAX = 0.00056354539312

ENERGY AND DIFF OF MACROITER    2 =    -76.06610450    -0.00946295

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00437808468152
GROUP =  2  ITER =  2 CMAX = 0.01638344197307
GROUP =  2  ITER =  3 CMAX = 0.00235152475684
GROUP =  2  ITER =  4 CMAX = 0.00002904648874

ENERGY AND DIFF OF MACROITER    3 =    -76.06657692    -0.00047242

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00269200649555
GROUP =  2  ITER =  2 CMAX = 0.00683835744882
GROUP =  2  ITER =  3 CMAX = 0.00034897666516

ENERGY AND DIFF OF MACROITER    4 =    -76.06676597    -0.00018906

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00165077600286
GROUP =  2  ITER =  2 CMAX = 0.00417035367474
GROUP =  2  ITER =  3 CMAX = 0.00013307957815

ENERGY AND DIFF OF MACROITER    5 =    -76.06684682    -0.00008085

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00090095722574

ENERGY AND DIFF OF MACROITER    6 =    -76.06686931    -0.00002249

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00069435175485

ENERGY AND DIFF OF MACROITER    7 =    -76.06687834    -0.00000903

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1 CMAX = 0.00066439786865

```

ENERGY AND DIFF OF MACROITER 8 = -76.06688329 -0.00000494

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX = 0.00055247302183

ENERGY AND DIFF OF MACROITER 9 = -76.06688576 -0.00000248

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX = 0.00047279866987

ENERGY AND DIFF OF MACROITER 10 = -76.06688704 -0.00000127

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX = 0.00038107654793

ENERGY AND DIFF OF MACROITER 11 = -76.06688767 -0.00000064

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX = 0.00031636144456

OVERLAP MATRIX OF VB ORBITALS FOR GROUP 2

	1	2	3	4
1	1.00			
2	0.81	1.00		
3	0.19	0.07	1.00	
4	0.07	-0.03	0.81	1.00

ENERGY AND DIFF OF MACROITER 12 = -76.06688800 -0.00000033

MULLIKEN CHARGES ON ATOMS

ATOM	CHARGE
1	-0.30493
2	0.15247
3	0.15247

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

Orbital#	1	2	3	4	5	6
Group#	1	1	1	2	2	2
AO# LABELS						
1 1 O s	0.99265	-0.12834	-0.00001	0.03101	-0.02724	0.03101
2 1 O s	-0.05896	-0.43353	0.00000	0.23596	-0.00152	0.23596
3 1 O s	-0.07267	-0.50981	0.00001	0.05226	-0.08646	0.05226
4 1 O x	0.00000	0.00000	-0.00000	-0.47302	-0.15502	0.47302
5 1 O y	0.00001	0.00001	0.63113	-0.00000	-0.00000	-0.00000
6 1 O z	0.03853	0.28304	-0.00000	0.50234	0.16380	0.50234
7 1 O x	0.00000	0.00000	-0.00000	-0.15851	-0.10756	0.15851
8 1 O y	0.00001	0.00000	0.49797	-0.00000	-0.00000	-0.00000
9 1 O z	0.03119	0.23982	-0.00000	0.21820	0.18232	0.21820
10 1 O xx	-0.00048	-0.00443	0.00000	0.00105	-0.00299	0.00105

11	1	O	yy	-0.00080	-0.00550	0.00000	-0.01750	0.00049	-0.01750
12	1	O	zz	0.00127	0.00993	-0.00000	0.01645	0.00250	0.01645
13	1	O	xy	0.00000	0.00000	0.00000	0.00000	-0.00000	-0.00000
14	1	O	xz	0.00000	0.00000	-0.00000	-0.03331	0.00029	0.03331
15	1	O	yz	0.00000	0.00000	0.01717	-0.00000	-0.00000	-0.00000
16	2	H	s	-0.00378	-0.01838	0.00000	-0.00425	-0.04051	0.11931
17	2	H	s	0.00303	0.02009	-0.00000	-0.01383	-0.03947	0.03854
18	2	H	x	0.00219	0.01074	-0.00000	-0.01860	-0.01154	-0.03303
19	2	H	y	0.00000	0.00000	0.02945	-0.00000	-0.00000	0.00000
20	2	H	z	0.00304	0.01897	-0.00000	0.01461	0.01864	-0.01891
21	3	H	s	-0.00378	-0.01838	0.00000	0.11931	0.70111	-0.00425
22	3	H	s	0.00303	0.02009	-0.00000	0.03854	0.13744	-0.01383
23	3	H	x	-0.00219	-0.01074	0.00000	0.03303	0.05810	0.01860
24	3	H	y	0.00000	0.00000	0.02945	-0.00000	-0.00000	-0.00000
25	3	H	z	0.00304	0.01897	-0.00000	-0.01891	-0.02089	0.01461

```
=====
Orbital#           7           8
Group#             2           3
-----
```

```
AO# LABELS
 1  1  O      s  -0.02724  -0.16945
 2  1  O      s  -0.00152  -1.32291
 3  1  O      s  -0.08646  -2.08434
 4  1  O      x   0.15502   0.00000
 5  1  O      y  -0.00000  -0.00000
 6  1  O      z   0.16380   0.15963
 7  1  O      x   0.10756  -0.00000
 8  1  O      y  -0.00000  -0.00000
 9  1  O      z   0.18232  -0.10632
10  1  O     xx  -0.00299   1.58788
11  1  O     yy   0.00049   1.44053
12  1  O     zz   0.00250   1.49352
13  1  O     xy   0.00000  -0.00000
14  1  O     xz  -0.00029  -0.00000
15  1  O     yz   0.00000   0.00000
16  2  H      s   0.70111  -0.10395
17  2  H      s   0.13744   0.28012
18  2  H      x  -0.05810   0.20557
19  2  H      y  -0.00000   0.00000
20  2  H      z  -0.02089   0.17180
21  3  H      s  -0.04051  -0.10395
22  3  H      s  -0.03947   0.28012
23  3  H      x   0.01154  -0.20557
24  3  H      y  -0.00000   0.00000
25  3  H      z   0.01864   0.17180
```

WORDS OF DYNAMIC MEMORY USED IN VB2000 CODE ONLY: 71125

```
STATISTICS OF CPU TIMES (SECONDS)
CPU TIME FOR INITIALIZATION      0.140
CPU TIME FOR MACROITERATION      4.079
TOTAL CPU TIME                   4.219
```

VB2000 output consists of many parts, and some parts may or may not be printed according to the calculation type and the print control. Example 1 is a typical case of VB calculation. It consists of the following parts:

1. The banner

The output starts with the banner of the program, which includes the version number, the build date, the suggested citations for this program and the reference paper and the copyright information.

2. Key words and coordinates

The keywords of the controls and the atom coordinates are echoed in the output.

3. Hartree-Fock molecular orbital coefficients

The Hartree-Fock molecular orbitals and the orbitals energies E(a.u.) will be printed if the keyword **PRINTALL** is included in the control. The molecular orbital coefficients are printed by columns. The first column of each block is the atomic orbital indexes, the second column is the atom indexes, and the third column is the atomic symbol, and the fourth column is the atomic orbital symbol. Starting from the fifth column are the MO coefficients. Each block a maximum of 6 orbitals are printed. If the molecule has more than 6 orbitals, multiple blocks of MO coefficients will be printed.

4. Localized molecular orbitals and the corresponding labels

The orbital coefficients of LMOs are printed in a very similar format as the Hartree-Fock MOs. Each LMO has a bonding label. The LMOs are sorted first by the bonding type and then by orbital energies. The inner core orbitals and lone pairs come first, then the bonding LMOs. For the same bonding type of LMOs, the lower energy LMOs come first. In H₂O, there are three non bonding LMOs, one 1s core of the oxygen atom, labeled as 1:(0), and the two lone-pairs, labeled as 1:(1) and 1:(2), respectively. The two bonding LMOs, which correspond to two O-H bonds, are labeled as 1-2(1) and 1-3(1), respectively.

5. Mulliken charges of LMOs

The atomic charges from Mulliken population analysis of each individual LMO are also printed. It is much easier to check the localization of each LMO from its Mulliken charge distribution. For the inner core and the two lone-pair orbitals, the Mulliken electron charge is almost localized on one atom. For the two bonding LMOs, the Mulliken charge is roughly equal distributed on two bonding atom.

6. General controls

This gives a summary of the GFT controls.

7. Partitioning of LMOs

This table gives the group assignments of all LMOs. The splitting flag of each LMO is also printed. For H₂O, the orbital 1,2 and 3, which are the core, and two lone pairs of O atom, are assigned to group 1, and orbital 4 and 5, which are the two bonding orbitals, are assigned to group 2 (i.e. the VB group). The orbital 1,2 and 3 are not split, and orbital 4 and 5 are split into 4 VB orbitals.

8. Initial guess of orbitals

The initial guess of orbitals of all groups are printed in a very similar format as LMOs. The orbitals and their corresponding orbital labels are listed in the consequence of the electron groups that they belong to. The first three orbitals belong to the first group, i.e. the Hartree-Fock group, and the last four orbitals belong to the VB group. Orbital 4 and 5 are derived from the bonding LMO of atom 3 and 1, and orbital 6 and 7 are derived from the bonding LMO of atom 2 and 1.

9. Macro iterations

The energy of each macro iteration and the energy change from previous macro iteration are printed. The maximum orbital coefficient change of VB orbital optimization are also printed.

10. Overlap of VB orbitals

The overlap matrix of VB orbitals are printed. Since the matrix is symmetric, only the lower half of the matrix is printed.

11. Final orbitals

The final optimized orbitals of all groups are printed in the same format as above. The group indexes of the orbitals are also printed.

8.2 A CASVB(4,4) calculation of H₂O

Input file:

```
#! CASVB(4,4)/D95 PRINTALL

TEST H2O CASVB(4,4)

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582
```

Output:

CASVB calculation has more output as compare to simple VB calculation of Example 1.
The automatically generated VB structures are printed:

```
SYMBOLIC VB STRUCTURE(S) OF GROUP 2
STR. NO.      RUMER PATTERN
  1           1  2  3  4
  2           1  3  2  4
  3           1  1  2  3
  4           1  1  3  4
  5           1  1  2  4
  6           2  2  3  4
  7           2  2  4  1
  8           2  2  3  1
  9           3  3  4  1
 10           3  3  1  2
 11           3  3  4  2
 12           4  4  1  2
 13           4  4  2  3
 14           4  4  1  3
 15           1  1  2  2
 16           2  2  3  3
 17           3  3  4  4
 18           1  1  3  3
 19           1  1  4  4
 20           2  2  4  4
```

The final energy and the structure weights computed from Mulliken, Löwdin and Hiberty methods are shown as follows:

```
.....
Normalized structure coefficients
-0.4184 -0.0270  0.0246 -0.2841  0.0208 -0.0658 -0.0120 -0.0128  0.0247 -0.2840
 0.0209 -0.0659 -0.0121 -0.0129  0.0061 -0.0912  0.0061 -0.1417 -0.0913  0.0067

====Mulliken Weight====
 0.3593 -0.0142  0.0029  0.2080  0.0021  0.0316  0.0009  0.0012  0.0029  0.2079
 0.0021  0.0317  0.0009  0.0012  0.0000  0.0400  0.0000  0.0831  0.0401 -0.0017

====Löwdin Weight====
 0.2333  0.0220  0.0029  0.1920  0.0021  0.0506  0.0009  0.0012  0.0029  0.1918
```

```

0.0021  0.0507  0.0009  0.0012  0.0000  0.0580  0.0000  0.1217  0.0581  0.0077
====Hiberty Weight====
0.4542  0.0019  0.0016  0.2095  0.0011  0.0112  0.0004  0.0004  0.0016  0.2093
0.0011  0.0113  0.0004  0.0004  0.0001  0.0216  0.0001  0.0521  0.0216  0.0001

```

```

ENERGY AND DIFF OF MACROITER 12 = -76.06287735 -0.00000047

```

```

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

```

```

=====
Orbital#          1          2          3          4          5          6
Group#           1          1          1          2          2          2
-----
AO# LABELS
1  1  O      s    0.59144    0.07222   -0.00001   -0.05097    0.01688   -0.05098
2  1  O      s    0.47755    0.13539   -0.00001   -0.07247    0.02649   -0.07248
3  1  O      s   -0.05350   -0.52084    0.00001    0.26916   -0.19009    0.26927
4  1  O      s   -0.05175   -0.52396    0.00001    0.06214   -0.02634    0.06217
5  1  O      x    0.00000    0.00000    0.00000    0.64259   -0.08054   -0.64270
6  1  O      y   -0.00001   -0.00001   -0.72798    0.00000    0.00000    0.00000
7  1  O      z    0.02957    0.28037    0.00000    0.62089   -0.08892    0.62097
8  1  O      x    0.00000    0.00000    0.00000    0.05365    0.14212   -0.05368
9  1  O      y    0.00000    0.00000   -0.40993    0.00000    0.00000    0.00000
10 1  O      z    0.01669    0.17087    0.00000    0.10586    0.27942    0.10568
11 2  H      s   -0.00265   -0.02427    0.00000    0.00324    0.65436    0.00752
12 2  H      s    0.00340    0.03568    0.00000   -0.00382    0.30913   -0.00896
13 3  H      s   -0.00265   -0.02427    0.00000    0.00776   -0.03963    0.00310
14 3  H      s    0.00340    0.03568    0.00000   -0.00886   -0.06449   -0.00387
=====

```

```

=====
Orbital#          7
Group#           2
-----
AO# LABELS
1  1  O      s    0.01687
2  1  O      s    0.02648
3  1  O      s   -0.19006
4  1  O      s   -0.02636
5  1  O      x    0.08051
6  1  O      y    0.00000
7  1  O      z   -0.08890
8  1  O      x   -0.14211
9  1  O      y    0.00000
10 1  O      z    0.27944
11 2  H      s   -0.03962
12 2  H      s   -0.06448
13 3  H      s    0.65435
14 3  H      s    0.30913

```

```

STATISTICS OF CPU TIMES (SECONDS)
CPU TIME FOR INITIALIZATION      0.010
CPU TIME FOR MACROITERATION      0.701
TOTAL CPU TIME                   0.711

```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example02.txt>

8.3 A CASVB(8,6) calculation of H₂O

Input file:

```

#! CASVB(8,6)/D95

TEST H2O CASVB(8,6)

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582

$VBGA
1-2 => 2
1-3 => 2
1:(1) => 2
1:(2) => 2

$02VBORB
1-2
1-3
1:(1)
1:(2)

```

Output:

```

.....
Normalized structure coefficients
-0.4319 -0.0243 -0.0428 -0.0373 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0129 0.0002
0.0427 -0.0051 0.0115 0.0116 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0386 -0.2916 0.0268 0.0079 0.0004 0.0022 -0.0005 0.0000 0.0000 0.0000
0.0000 0.0001 0.0075 0.0000 0.0000 0.0000 0.0000 -0.0016 0.0000 0.0075
0.0000 0.0089 -0.0152 0.0060 0.0022 0.0079 0.0004 0.0000 0.0000 -0.0005
-0.0692 -0.0149 -0.0243 0.0386 -0.2916 0.0268 -0.0692 -0.0149 -0.0243 0.0286
0.0287 -0.0139 0.0089 0.0060 -0.0152 0.0000 0.0000 0.0001 0.0000 0.0000
0.0010 0.0000 -0.0016 0.0000 0.0000 0.0010 0.0000 0.0005 -0.0005 0.0005
0.0042 0.0070 -0.0004 0.0055 0.0011 0.0008 -0.0953 0.0042 0.0070 0.0055
-0.1423 -0.0953 0.0008 0.0011 0.0070

====Mulliken Weight====
0.3741 -0.0129 -0.0046 -0.0033 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0022 0.0000
-0.0073 -0.0005 -0.0011 -0.0011 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0019 0.2129 0.0013 -0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001
0.0000 -0.0010 -0.0021 0.0002 0.0000 -0.0001 0.0000 0.0000 0.0000 0.0000
0.0349 0.0005 0.0008 0.0019 0.2129 0.0013 0.0349 0.0005 0.0008 -0.0037
-0.0037 0.0004 -0.0010 0.0002 -0.0021 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0000 0.0001
0.0000 0.0000 0.0000 0.0001 0.0000 0.0000 0.0427 0.0000 0.0000 0.0001
0.0835 0.0427 0.0000 0.0000 -0.0019

====Lowdin Weight====

```

0.2316	0.0223	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0008	0.0004
0.0001	0.0002	0.0003	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0019	0.1882	0.0015	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
0.0000	0.0002	0.0005	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
0.0538	0.0005	0.0007	0.0019	0.1884	0.0015	0.0539	0.0005	0.0007	0.0000
0.0000	0.0001	0.0001	0.0000	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0004
0.0000	0.0001	0.0000	0.0001	0.0000	0.0000	0.0593	0.0000	0.0001	0.0001
0.1200	0.0594	0.0000	0.0000	0.0091					

====Hiberty Weight====

0.4444	0.0014	0.0044	0.0033	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000
0.0043	0.0001	0.0003	0.0003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0036	0.0205	0.0017	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
0.0000	0.0002	0.0005	0.0001	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
0.0114	0.0005	0.0014	0.0035	0.2026	0.0017	0.0114	0.0005	0.0014	0.0020
0.0020	0.0005	0.0002	0.0001	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0001	0.0000	0.0001	0.0000	0.0000	0.0216	0.0000	0.0001	0.0001
0.0483	0.0216	0.0000	0.0000	0.0001					

ENERGY AND DIFF OF MACROITER 12 = -76.06499679 -0.00000025

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

=====									
Orbital#									
Group#									
		1	2	3	4	5	6		
		1	2	2	2	2	2		

AO#	LABELS								
1	1 O s	0.58750	-0.08500	0.00000	-0.05986	-0.00501	-0.05985		
2	1 O s	0.47082	-0.13640	0.00000	-0.08675	-0.00816	-0.08673		
3	1 O s	-0.02954	0.44491	-0.00001	0.31762	-0.07866	0.31757		
4	1 O s	-0.02750	0.50145	-0.00001	0.10885	0.09491	0.10881		
5	1 O x	0.00000	-0.00007	0.00000	0.63821	-0.07100	-0.63825		
6	1 O y	0.00000	-0.00001	-0.72885	0.00000	0.00000	0.00000		
7	1 O z	0.01636	-0.43365	0.00001	0.59171	-0.15847	0.59172		
8	1 O x	0.00000	0.00000	0.00000	0.05826	0.12428	-0.05826		
9	1 O y	0.00000	-0.00001	-0.40888	0.00000	0.00000	0.00000		
10	1 O z	0.00864	-0.18936	0.00000	0.08773	0.24933	0.08774		
11	2 H s	-0.00157	0.02721	0.00000	0.00649	0.63996	0.00870		
12	2 H s	0.00176	-0.03248	0.00000	-0.00781	0.29336	-0.01022		
13	3 H s	-0.00157	0.02720	0.00000	0.00870	0.00322	0.00648		
14	3 H s	0.00176	-0.03249	0.00000	-0.01023	-0.05788	-0.00781		

=====									
Orbital#									
Group#									
		7							
		2							

AO#	LABELS								
1	1 O s	-0.00501							
2	1 O s	-0.00816							
3	1 O s	-0.07868							
4	1 O s	0.09490							
5	1 O x	0.07099							
6	1 O y	0.00000							

7	1	O	z	-0.15852
8	1	O	x	-0.12428
9	1	O	y	0.00000
10	1	O	z	0.24933
11	2	H	s	0.00323
12	2	H	s	-0.05787
13	3	H	s	0.63996
14	3	H	s	0.29337

STATISTICS OF CPU TIMES (SECONDS)
CPU TIME FOR INITIALIZATION 0.000
CPU TIME FOR MACROITERATION 43.382
TOTAL CPU TIME 43.382

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example03.txt>

8.4 VB(6) calculation of C₆H₆

Input file:

```

#! VB(6)/D95 UNITS=BOHR

TEST BENZENE

0 1
6 .000000 2.637000 .000000
1 .000000 4.688000 .000000
6 2.28370898978 1.318500 .000000
1 4.05992709294 2.344000 .000000
6 2.28370898978 -1.318500 .000000
1 4.05992709294 -2.344000 .000000
6 .000000 -2.637000 .000000
1 .000000 -4.688000 .000000
6 -2.28370898978 -1.318500 .000000
1 -4.05992709294 -2.344000 .000000
6 -2.28370898978 1.318500 .000000
1 -4.05992709294 2.344000 .000000

$02PIVBO
6
1,3,5,7,9,11

$02VBSTR
5
1 2 3 4 5 6
2 3 4 5 6 1
1 4 2 3 5 6
2 5 3 4 1 6
3 6 1 2 4 5

```

Output:

```

SYMBOLIC VB STRUCTURE(S) OF GROUP 2
STR. NO.      RUMER PATTERN
  1           1 2 3 4 5 6
  2           2 3 4 5 6 1
  3           1 4 2 3 5 6
  4           2 5 3 4 1 6
  5           3 6 1 2 4 5

ENERGY AND DIFF OF MACROITER  1 =  -230.70991476  -230.70991476
ENERGY AND DIFF OF MACROITER  2 =  -230.71017076  -0.00025600
ENERGY AND DIFF OF MACROITER  3 =  -230.71017316  -0.00000240
ENERGY AND DIFF OF MACROITER  4 =  -230.71017323  -0.00000007

```

OVERLAP OF VB ORBITALS FOR GROUP 2

	1	2	3	4	5	6
1	1.00					
2	0.52	1.00				
3	0.04	0.52	1.00			
4	-0.14	0.04	0.52	1.00		
5	0.04	-0.14	0.04	0.52	1.00	
6	0.52	0.04	-0.14	0.04	0.52	1.00

OVERLAP OF VB STRUCTURES FOR GROUP 2

	1	2	3	4	5
1	1.00				
2	0.67	1.00			
3	-0.83	-0.83	1.00		
4	-0.83	-0.83	0.75	1.00	
5	-0.83	-0.83	0.75	0.75	1.00

Normalized structure coefficients

0.4432 0.4432 -0.0703 -0.0703 -0.0703

====Mulliken Weight====

0.4043 0.4043 0.0638 0.0638 0.0638

====Lowdin Weight====

0.2580 0.2580 0.1613 0.1613 0.1613

====Hiberty Weight====

0.4818 0.4818 0.0121 0.0121 0.0121

ENERGY AND DIFF OF MACROITER 5 = -230.71017324 -0.00000001

5 VB structures (two Kekule and three Dewar) are used in the computation. Apparently, the two Kekule structures have much larger weights than the three Dewar structures.

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example04.txt>

8.5 VB(6) calculation of CC triple bond of C₂H₂

Input file:

```
#! VB(6)/D95

TEST C2H2 TRIPLE BOND (SIGMA+PI)

0 1
6 .000000 .000000 .610009
6 .000000 .000000 -.610009
1 .000000 .000000 1.690006
1 .000000 .000000 -1.690006

$VBGA
1#2 => 2
```

Output:

```
SYMBOLIC VB STRUCTURE(S) OF GROUP 2
STR. NO.      RUMER PATTERN
  1           1  2  3  4  5  6
```

```
ENERGY AND DIFF OF MACROITER    1 =    -76.85662973    -76.85662973
ENERGY AND DIFF OF MACROITER    2 =    -76.85698355    -0.00035382
ENERGY AND DIFF OF MACROITER    3 =    -76.85698490    -0.00000135
ENERGY AND DIFF OF MACROITER    4 =    -76.85698493    -0.00000004
```

```
OVERLAP OF VB ORBITALS FOR GROUP 2
```

```

      1      2      3      4      5      6
=====
1  1.00
2  0.91  1.00
3  0.00  0.00  1.00
4  0.00  0.00  0.69  1.00
5  0.00  0.00  0.00  0.00  1.00
6  0.00  0.00  0.00  0.00  0.69  1.00
```

```
ENERGY AND DIFF OF MACROITER    5 =    -76.85698494    -0.00000001
```

```
ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)
```

```
=====
Orbital#      1      2      3      4      5      6
Group#        1      1      1      1      2      2
-----
AO# LABELS
1  1  C      s    0.60349 -0.00078 -0.00050  0.07152 -0.05455 -0.10847
2  1  C      s    0.44133 -0.00105 -0.00124  0.10968 -0.07100 -0.14322
3  1  C      s   -0.00661  0.00208 -0.00749 -0.34302  0.05067  0.49377
4  1  C      s   -0.01023  0.00242  0.00715 -0.27493  0.08073  0.16942
5  1  C      x    0.00000  0.00000  0.00000  0.00000  0.00010 -0.00012
6  1  C      y    0.00000  0.00000  0.00000  0.00000  0.00011 -0.00012
7  1  C      z   -0.01426 -0.00065  0.00704 -0.35477 -0.17435 -0.60564
8  1  C      x    0.00000  0.00000  0.00000  0.00000 -0.00004  0.00004
```

9	1	C	y	0.00000	0.00000	0.00000	0.00000	-0.00004	0.00004
10	1	C	z	-0.00146	-0.00076	-0.02281	-0.06389	0.02811	-0.00567
11	2	C	s	-0.00078	0.60349	0.07152	-0.00050	-0.10847	-0.05455
12	2	C	s	-0.00105	0.44133	0.10967	-0.00124	-0.14322	-0.07100
13	2	C	s	0.00208	-0.00661	-0.34302	-0.00749	0.49377	0.05067
14	2	C	s	0.00242	-0.01023	-0.27493	0.00715	0.16942	0.08073
15	2	C	x	0.00000	0.00000	0.00000	0.00000	-0.00012	0.00010
16	2	C	y	0.00000	0.00000	0.00000	0.00000	-0.00012	0.00011
17	2	C	z	0.00065	0.01426	0.35477	-0.00704	0.60564	0.17435
18	2	C	x	0.00000	0.00000	0.00000	0.00000	0.00004	-0.00004
19	2	C	y	0.00000	0.00000	0.00000	0.00000	0.00004	-0.00004
20	2	C	z	0.00076	0.00146	0.06389	0.02281	0.00567	-0.02811
21	3	H	s	-0.01028	0.00062	-0.00373	-0.30356	-0.05057	-0.05797
22	3	H	s	-0.00607	0.00059	0.00861	-0.15467	-0.09340	-0.06192
23	4	H	s	0.00062	-0.01028	-0.30356	-0.00373	-0.05797	-0.05057
24	4	H	s	0.00059	-0.00607	-0.15467	0.00861	-0.06192	-0.09340

```
=====
Orbital#           7           8           9           10
Group#             2           2           2           2
=====
```

AO#	LABELS						
1	1	C	s	-0.00004	-0.00004	-0.00004	-0.00004
2	1	C	s	-0.00005	-0.00005	-0.00005	-0.00005
3	1	C	s	0.00012	0.00013	0.00012	0.00013
4	1	C	s	0.00005	0.00007	0.00005	0.00007
5	1	C	x	0.16080	0.75280	-0.00274	-0.01283
6	1	C	y	0.00274	0.01283	0.16080	0.75280
7	1	C	z	-0.00018	-0.00018	-0.00018	-0.00018
8	1	C	x	0.06293	0.24007	-0.00107	-0.00409
9	1	C	y	0.00107	0.00409	0.06293	0.24007
10	1	C	z	0.00001	0.00000	0.00001	0.00000
11	2	C	s	-0.00004	-0.00004	-0.00004	-0.00004
12	2	C	s	-0.00005	-0.00005	-0.00005	-0.00005
13	2	C	s	0.00013	0.00012	0.00013	0.00012
14	2	C	s	0.00007	0.00005	0.00007	0.00005
15	2	C	x	0.75280	0.16080	-0.01283	-0.00274
16	2	C	y	0.01283	0.00274	0.75280	0.16080
17	2	C	z	0.00018	0.00018	0.00018	0.00018
18	2	C	x	0.24007	0.06293	-0.00409	-0.00107
19	2	C	y	0.00409	0.00107	0.24007	0.06293
20	2	C	z	0.00000	-0.00001	0.00000	-0.00001
21	3	H	s	-0.00002	-0.00003	-0.00002	-0.00003
22	3	H	s	-0.00003	-0.00004	-0.00003	-0.00004
23	4	H	s	-0.00003	-0.00002	-0.00003	-0.00002
24	4	H	s	-0.00004	-0.00003	-0.00004	-0.00003

```
STATISTICS OF CPU TIMES (SECONDS)
CPU TIME FOR INITIALIZATION      0.050
CPU TIME FOR MACROITERATION      1.212
TOTAL CPU TIME                   1.262
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example05.txt>

8.6 VB calculation of the 3 banana bonds in C₂H₂

```
#! VB(6)/D95 PRINTALL

TEST C2H2 TRIPLE BOND (3 BANANA BONDS, 6 ELECTRONS, VBGA INPUT)

0 1
6   .528275  0.305   0.0
6  -.528275 -0.305   0.0
1   1.463583  0.845   0.0
1  -1.463583 -0.845   0.0

$VBGA
(1#2) => 2

$02VBORB
(1->2) + 1^
(2->1) + 2^
(1->2) - 0.5 (1^) + 0.866025 (1^(2))
(2->1) - 0.5 (2^) + 0.866025 (2^(2))
(1->2) - 0.5 (1^) - 0.866025 (1^(2))
(2->1) - 0.5 (2^) - 0.866025 (2^(2))
```

Final energy:

```
ENERGY AND DIFF OF MACROITER      8 =      -76.86471892      -0.00000030
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example06.txt>

8.7 VB(10) calculation of C₂H₂

```
#! VB(10)/D95
```

```
TEST C2H2 TRIPLE BOND (SIGMA+PI)
```

```
0 1
```

```
6 .000000 .000000 .610009
```

```
6 .000000 .000000 -.610009
```

```
1 .000000 .000000 1.690006
```

```
1 .000000 .000000 -1.690006
```

Final results:

ENERGY AND DIFF OF MACROITER	1 =	-76.88684004	-76.88684004
ENERGY AND DIFF OF MACROITER	2 =	-76.88753450	-0.00069445
ENERGY AND DIFF OF MACROITER	3 =	-76.88754860	-0.00001410
ENERGY AND DIFF OF MACROITER	4 =	-76.88754891	-0.00000032
ENERGY AND DIFF OF MACROITER	5 =	-76.88754896	-0.00000005

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example07.txt>

8.8 VB(12) calculation of C₂H₄

```
#! VB(12)/D95
```

```
TEST C2H4 DOUBLE BOND. ALL VALENCE ELECTRONS AND BONDS
```

```
0 1
6 .000000 .000000 .000000
6 .000000 .000000 1.319979
1 .935305 .000000 -.539999
1 -.935305 .000000 -.539999
1 -.935305 .000000 1.859978
1 .935305 .000000 1.859978
```

```
$02VBSCF
0.001,5
```

Output:

```
ENERGY AND DIFF OF MACROITER    1 =    -78.11026994    -78.11026994
ENERGY AND DIFF OF MACROITER    2 =    -78.11155896    -0.00128902
ENERGY AND DIFF OF MACROITER    3 =    -78.11158396    -0.00002500
ENERGY AND DIFF OF MACROITER    4 =    -78.11158477    -0.00000082
```

```
OVERLAP OF VB ORBITALS FOR GROUP    2
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.00											
2	0.89	1.00										
3	0.23	0.17	1.00									
4	0.17	0.13	0.84	1.00								
5	0.17	0.23	-0.09	-0.03	1.00							
6	0.13	0.17	-0.03	0.01	0.84	1.00						
7	-0.17	-0.23	-0.12	-0.06	-0.22	-0.15	1.00					
8	-0.13	-0.17	-0.06	-0.01	-0.15	-0.11	0.84	1.00				
9	0.23	0.17	0.22	0.15	0.12	0.06	0.09	0.03	1.00			
10	0.17	0.13	0.15	0.11	0.06	0.01	0.03	-0.01	0.84	1.00		
11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	1.00

```
ENERGY AND DIFF OF MACROITER    5 =    -78.11158494    -0.00000017
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example08.txt>

8.9 VB(14) calculation of C₂H₆

#! VB(14)/D95

TEST C2H6 (7 SIGMA BONDS)

```
0 1
6 .000000 .000000 .000000
6 .000000 .000000 1.540011
1 1.027669 .000000 -.363311
1 -.513834 -.889987 -.363311
1 -.513834 .889987 -.363311
1 -1.027669 .000000 1.903322
1 .513834 .889987 1.903322
1 .513834 -.889987 1.903322
```

\$02VBSCF
0.001,5

Output:

SYMBOLIC VB STRUCTURE(S) OF GROUP 2

STR. NO.	RUMER PATTERN
1	1 2 3 4 5 6 7 8 9 10 11 12 13 14

ENERGY AND DIFF OF MACROITER	1 =	-79.31292996	-79.31292996
ENERGY AND DIFF OF MACROITER	2 =	-79.31603338	-0.00310343
ENERGY AND DIFF OF MACROITER	3 =	-79.31609730	-0.00006392
ENERGY AND DIFF OF MACROITER	4 =	-79.31609901	-0.00000170
ENERGY AND DIFF OF MACROITER	5 =	-79.31609919	-0.00000019

OVERLAP OF VB ORBITALS FOR GROUP 2

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.00											
2	0.84	1.00										
3	-0.15	-0.20	1.00									
4	-0.11	-0.16	0.83	1.00								
5	0.20	0.15	-0.07	-0.04	1.00							
6	0.16	0.11	-0.04	-0.02	0.83	1.00						
7	-0.15	-0.20	0.20	0.16	0.07	0.03	1.00					
8	-0.11	-0.16	0.16	0.13	0.03	0.01	0.83	1.00				
9	0.15	0.20	-0.20	-0.16	0.07	0.04	-0.20	-0.16	1.00			
10	0.11	0.16	-0.16	-0.13	0.04	0.02	-0.16	-0.13	0.83	1.00		
11	0.20	0.15	-0.07	-0.04	0.20	0.16	-0.07	-0.04	-0.07	-0.03	1.00	
12	0.16	0.11	-0.04	-0.02	0.16	0.13	-0.04	-0.02	-0.03	-0.01	0.83	1.00
13	-0.20	-0.15	-0.07	-0.03	-0.20	-0.16	0.07	0.04	-0.07	-0.04	-0.20	-0.16

```

14 -0.16 -0.11 -0.03 -0.01 -0.16 -0.13  0.04  0.02 -0.04 -0.02 -0.16 -0.13

      13      14
=====
13  1.00
14  0.83  1.00

ENERGY AND DIFF OF MACROITER      6 =      -79.31609922      -0.00000003
....

```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example09.txt>

8.10 VB(4) calculation of the triplet state and the spin density of TMM

```

#! VB(4)/D95 UNITS=BOHR SPDEN PRINTALL

TMM TRIPLET

0 3
6      0.0000000000      0.0000000000      0.0000000000
6      0.0000000000      2.6985287117      0.0000000000
6      2.3369944172     -1.3492643558      0.0000000000
6      -2.3369944172     -1.3492643558      0.0000000000
1     -1.7528496640      3.7475875942      0.0000000000
1      1.7528496640      3.7475875942      0.0000000000
1      4.1219308914     -0.3557814591      0.0000000000
1      2.3690812275     -3.3918061351      0.0000000000
1     -4.1219308914     -0.3557814591      0.0000000000
1     -2.3690812275     -3.3918061351      0.0000000000

$GRPDIM
13,8

$02VBSTR
3
  1  2  3  4
  1  3  4  2
  1  4  2  3

$AOGROUP
8
7,17,27,37,10,20,30,40
2,2, 2, 2, 2, 2, 2, 2

$NOTROT
2
1,2
2,3

```

This is a case of σ - π separation. We can divide AO space into two subspaces, one for σ electrons and the other for π electrons. With D95 basis set, there are 8 π AOs, all assigned to Group 2, i.e. the π electron group described by VB method. The remaining AOs belong to Group 1 (Hartree-Fock group) and Group 3 (virtual orbitals). Because of the symmetry of the molecule, the σ and π orbitals should not be mixed, i.e. Group 2 does not mix with Group 1 and Group 3. This is controlled by **\$NOTROT** flag. Even though only 4 VB orbitals are used for the π electrons, the VB orbitals are optimized within the 8 π AO subspace. The core-electrons are optimized within the remaining AO subspace (Group 1 and Group 3). The key word **SPDEN** tells the program to compute spin density.

The triplet state of the TMM has the full symmetry. To describe the state with proper

symmetry, three symmetry related VB structures (also linearly independent) are involved. In each VB structure, the first two orbitals form a single bond between the central carbon atom and one of its neighbor carbon, and the last two orbitals have unpaired (parallel) spins. As shown by the calculation, all three structures have exactly the same weights. The 4 VB orbitals show the same symmetry. The results of the final iteration are shown below:

```
VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP =  2  ITER =  1  CMAX =0.00005457121450
```

```
OVERLAP OF VB ORBITALS FOR GROUP  2
```

	1	2	3	4	5	6	7	8
1	1.00							
2	0.45	1.00						
3	0.45	-0.07	1.00					
4	0.45	-0.07	-0.07	1.00				
5	0.81	0.41	0.41	0.41	1.00			
6	0.51	0.76	0.08	0.08	0.66	1.00		
7	0.51	0.08	0.76	0.08	0.66	0.29	1.00	
8	0.51	0.08	0.08	0.76	0.66	0.29	0.29	1.00

```
OVERLAP OF VB STRUCTURES FOR GROUP  2
```

	1	2	3
1	1.00		
2	0.71	1.00	
3	0.71	0.71	1.00

```
Normalized structure coefficients
-0.3712 -0.3712 -0.3712
```

```
====Mulliken Weight====
0.3333 0.3333 0.3333
```

```
====Lowdin Weight====
0.3333 0.3333 0.3333
```

```
====Hiberty Weight====
0.3333 0.3333 0.3333
```

```
SPIN DENSITY ON ATOMS
```

```
-----
```

ATOM	DENSITY
1	-0.33634
2	0.77878
3	0.77878
4	0.77878
5	0.00000
6	0.00000
7	0.00000
8	0.00000

```
9      0.00000
10     0.00000
```

```
ENERGY AND DIFF OF MACROITER    5 =    -154.86929402    0.00000000
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example10.txt>

8.11 VB(4) calculation of the singlet state of TMM

```
#! VB(4)/D95 UNITS=BOHR
```

```
TEST TMM SINGLET
```

```
0 1
6      0.0000000000      0.0899765479      0.0000000000
6      0.0000000000      2.6789011522      0.0000000000
6      2.4206928037     -1.370278709      0.0000000000
6     -2.4206928037     -1.370278709      0.0000000000
1     -1.7489492060      3.7381024410      0.0000000000
1      1.7489492060      3.7381024410      0.0000000000
1      4.2227041376     -0.4121322875      0.0000000000
1      2.3886356760     -3.4109309960      0.0000000000
1     -4.2227041376     -0.4121322875      0.0000000000
1     -2.3886356760     -3.4109309960      0.0000000000
```

```
$GRPDIM
```

```
13,8
```

```
$AOGROUP
```

```
8
```

```
7,17,27,37,10,20,30,40
```

```
2,2, 2, 2, 2, 2, 2, 2
```

```
$02VBSTR
```

```
2
```

```
1 2 3 4
```

```
1 3 2 4
```

```
$NOTROT
```

```
2
```

```
1,2
```

```
2,3
```

The input of this calculation is very similar to that for Example 10. The only difference is the multiplicity and the choice of VB structures. The singlet state has a lower symmetry. As the results confirm, only one VB structure is needed to describe the singlet state properly. Since the singlet state has no unpaired spin, no spin-density calculation is needed.

Output:

```
.....
```

```
SYMBOLIC VB STRUCTURE(S) OF GROUP 2
```

```
STR. NO.      RUMER PATTERN
```

```
1            1 2 3 4
```

```
2            1 3 2 4
```

ENERGY AND DIFF OF MACROITER	1 =	-154.82743432	-154.82743432
ENERGY AND DIFF OF MACROITER	2 =	-154.83825534	-0.01082102
ENERGY AND DIFF OF MACROITER	3 =	-154.83885161	-0.00059627
ENERGY AND DIFF OF MACROITER	4 =	-154.83889289	-0.00004128
ENERGY AND DIFF OF MACROITER	5 =	-154.83889583	-0.00000293
ENERGY AND DIFF OF MACROITER	6 =	-154.83889609	-0.00000027

OVERLAP OF VB ORBITALS FOR GROUP 2

	1	2	3	4	5	6	7	8
1	1.00							
2	0.63	1.00						
3	0.42	0.13	1.00					
4	0.42	0.13	0.10	1.00				
5	0.78	0.58	0.41	0.41	1.00			
6	0.54	0.77	0.09	0.09	0.68	1.00		
7	0.44	0.24	0.78	0.14	0.63	0.28	1.00	
8	0.44	0.24	0.14	0.78	0.63	0.28	0.26	1.00

OVERLAP OF VB STRUCTURES FOR GROUP 2

	1	2
1	1.00	
2	-0.59	1.00

Normalized structure coefficients
-1.0000 0.0000

====Mulliken Weight====
1.0000 0.0000

====Lowdin Weight====
0.9189 0.0811

====Hiberty Weight====
1.0000 0.0000

ENERGY AND DIFF OF MACROITER	7 =	-154.83889613	-0.00000004
.....			

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example11.txt>

8.12 A two-VB group description of CH₃CH₂OH

```
#! VB(8).VB(8)/D95 PRINTALL

TEST C2H5OH (TWO VB GROUPS)

0 1
6 -0.4413 1.9095 -0.1486
1 0.1529 2.7834 -0.4194
1 -1.2624 1.8366 -0.8622
1 -0.8754 2.0884 0.8352
1 -0.2076 -0.2379 0.0887
6 0.4029 0.6280 -0.1666
1 0.8338 0.4543 -1.1531
1 2.0387 1.4420 0.5056
8 1.4477 0.7230 0.7814

$VBGA
1-2 => 2
1-3 => 2
1-4 => 2
1-6 => 2
6-7 => 3
5-6 => 3
6-9 => 3
8-9 => 3
```

This example shows how to divide a molecule into electron groups by grouping chemical bonds. The \$VBGA control flag provides a simple way of doing this. The results of two-VB group calculation are shown as follows:

```
.....

SYMBOLIC VB STRUCTURE(S) OF GROUP 2
STR. NO.      RUMER PATTERN
  1          1  2  3  4  5  6  7  8

GROUP = 2 ITER = 1 CMAX =0.08882622045810
GROUP = 2 ITER = 2 CMAX =0.05757816322642
GROUP = 2 ITER = 3 CMAX =0.02803710368545
GROUP = 2 ITER = 4 CMAX =0.01676545232832
GROUP = 2 ITER = 5 CMAX =0.00469405090304
GROUP = 2 ITER = 6 CMAX =0.00151560044991
GROUP = 2 ITER = 7 CMAX =0.00052262922350

SYMBOLIC VB STRUCTURE(S) OF GROUP 3
STR. NO.      RUMER PATTERN
  1          1  2  3  4  5  6  7  8

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
```

```

GROUP = 3  ITER = 1  CMAX =0.08087718449158
GROUP = 3  ITER = 2  CMAX =0.08180161210420
GROUP = 3  ITER = 3  CMAX =0.06759759162339
GROUP = 3  ITER = 4  CMAX =0.04258653572397
GROUP = 3  ITER = 5  CMAX =0.02983606847750
GROUP = 3  ITER = 6  CMAX =0.01385238643796
GROUP = 3  ITER = 7  CMAX =0.00130270755929
GROUP = 3  ITER = 8  CMAX =0.00022546285446

ENERGY AND DIFF OF MACROITER    1 =    -154.17666306    -154.17666306

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP = 2  ITER = 1  CMAX =0.01056614803143
GROUP = 2  ITER = 2  CMAX =0.02149020096531
GROUP = 2  ITER = 3  CMAX =0.00402693319014
GROUP = 2  ITER = 4  CMAX =0.00600859343232
GROUP = 2  ITER = 5  CMAX =0.00063894048449

VB-ORBITAL OPTIMIZATION OF GROUP  3 ...
GROUP = 3  ITER = 1  CMAX =0.01215903849804
GROUP = 3  ITER = 2  CMAX =0.03214995059146
GROUP = 3  ITER = 3  CMAX =0.00447770282658
GROUP = 3  ITER = 4  CMAX =0.00541737715233
GROUP = 3  ITER = 5  CMAX =0.00060140529393

ENERGY AND DIFF OF MACROITER    2 =    -154.17960904    -0.00294599

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP = 2  ITER = 1  CMAX =0.00160883379901
GROUP = 2  ITER = 2  CMAX =0.00351580500451
GROUP = 2  ITER = 3  CMAX =0.00023900211573

VB-ORBITAL OPTIMIZATION OF GROUP  3 ...
GROUP = 3  ITER = 1  CMAX =0.00354245834644
GROUP = 3  ITER = 2  CMAX =0.00629229916474
GROUP = 3  ITER = 3  CMAX =0.00086025849933

ENERGY AND DIFF OF MACROITER    3 =    -154.17967005    -0.00006100

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP = 2  ITER = 1  CMAX =0.00054290121281

VB-ORBITAL OPTIMIZATION OF GROUP  3 ...
GROUP = 3  ITER = 1  CMAX =0.00178694872413
GROUP = 3  ITER = 2  CMAX =0.00205132105570
GROUP = 3  ITER = 3  CMAX =0.00018713433320

ENERGY AND DIFF OF MACROITER    4 =    -154.17967538    -0.00000534

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP = 2  ITER = 1  CMAX =0.00027585300619

VB-ORBITAL OPTIMIZATION OF GROUP  3 ...
GROUP = 3  ITER = 1  CMAX =0.00065358919596

ENERGY AND DIFF OF MACROITER    5 =    -154.17968158    -0.00000620

VB-ORBITAL OPTIMIZATION OF GROUP  2 ...
GROUP = 2  ITER = 1  CMAX =0.00014750741088

```

```

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
GROUP = 3 ITER = 1 CMAX =0.00094104113875

ENERGY AND DIFF OF MACROITER    6 =   -154.17968611   -0.00000453

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX =0.00010398974776

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
GROUP = 3 ITER = 1 CMAX =0.00061031800461

ENERGY AND DIFF OF MACROITER    7 =   -154.17968876   -0.00000265

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX =0.00007362565759

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
GROUP = 3 ITER = 1 CMAX =0.00055234512828

ENERGY AND DIFF OF MACROITER    8 =   -154.17969011   -0.00000136

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX =0.00005742970662

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
GROUP = 3 ITER = 1 CMAX =0.00037120029862

ENERGY AND DIFF OF MACROITER    9 =   -154.17969080   -0.00000068

ENERGY AND DIFF OF MACROITER    9 =   -154.17969080   -0.00000068

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX =0.00004615902268

```

OVERLAP OF VB ORBITALS FOR GROUP 2

	1	2	3	4	5	6	7	8
1	1.00							
2	0.84	1.00						
3	-0.21	-0.16	1.00					
4	-0.16	-0.12	0.84	1.00				
5	-0.21	-0.16	0.21	0.16	1.00			
6	-0.16	-0.12	0.16	0.13	0.84	1.00		
7	-0.20	-0.15	0.21	0.17	0.21	0.17	1.00	
8	-0.15	-0.11	0.17	0.13	0.17	0.13	0.84	1.00

```

VB-ORBITAL OPTIMIZATION OF GROUP 3 ...
GROUP = 3 ITER = 1 CMAX =0.00030464297301

```

OVERLAP OF VB ORBITALS FOR GROUP 3

	1	2	3	4	5	6	7	8
1	1.00							
2	0.80	1.00						
3	0.23	0.12	1.00					

```

4  0.12  0.02  0.81  1.00
5  0.07  0.05  0.13  0.20  1.00
6  0.04  0.03  0.09  0.14  0.83  1.00
7  0.11  0.09 -0.08 -0.14 -0.20 -0.16  1.00
8  0.06  0.04 -0.06 -0.10 -0.16 -0.12  0.83  1.00

ENERGY AND DIFF OF MACROITER   10 =   -154.17969113      -0.00000033

.....

```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example12.txt>

8.13 All-valence electron VB calculation of C₆H₆

```

#! GPF(14)/STO-3G UNITS=BOHR NOBIAS FROZEN

TEST BENZENE 14 GROUPS (1 FROZEN CORE GROUP + 12 SIGMA
BONDS + 1 PI-GROUP)

0 1
6 .000000 2.637000 .000000
1 .000000 4.688000 .000000
6 2.28370898978 1.318500 .000000
1 4.05992709294 2.344000 .000000
6 2.28370898978 -1.318500 .000000
1 4.05992709294 -2.344000 .000000
6 .000000 -2.637000 .000000
1 .000000 -4.688000 .000000
6 -2.28370898978 -1.318500 .000000
1 -4.05992709294 -2.344000 .000000
6 -2.28370898978 1.318500 .000000
1 -4.05992709294 2.344000 .000000

$GENCTL
12,2,2,2,2,2,2,2,2,2,2,2,2,6
1,2,2,2,2,2,2,2,2,2,2,2,2,2

$VBSCF
0.001,10

$14PIVBO
6
1,3,5,7,9,11

$14VBSTR
5
1 2 3 4 5 6
2 3 4 5 6 1
1 4 2 3 5 6
2 5 3 4 6 1
3 6 4 5 1 2

$NOTROT
13
1,14
2,14
3,14
4,14
5,14
6,14
7,14
8,14
9,14

```

10,14
11,14
12,14
13,14

This example shows how to perform a VB calculation with all valence electrons included in the VB groups. The electrons are divided into 14 groups, including an HF group of 12 carbon core electrons, 12 GVB pairs, and the 6- π electron group. The σ - π mixing is suppressed by using the \$NOTROT control. The results are shown as follows:

.....

GENERAL CONTROLS (\$GENCTL)

=====

Number of electron groups = 14

Maximum macro-iterations = 12

Restart calculation(0/1/2)= 0

Group#	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Num. of electrons	12	2	2	2	2	2	2	2	2	2	2	2	2	6
Num. of spins	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Num. of orbitals	6	2	2	2	2	2	2	2	2	2	2	2	2	6
Method#	1	2	2	2	2	2	2	2	2	2	2	2	2	2

PARTITIONING OF LMOs INTO GROUPS (\$LMOGRP)

=====

LMO#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Group#	1	1	1	1	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14
Split	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LMO# 21
Group# 14
Split 1

SYMBOLIC VB STRUCTURE(S) OF GROUP 2

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 3

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 4

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 5

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 6

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 7

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 8

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 9

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 10

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 11

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 12

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 13

STR. NO.	RUMER PATTERN
1	1 2

SYMBOLIC VB STRUCTURE(S) OF GROUP 14

STR. NO.	RUMER PATTERN
1	1 2 3 4 5 6
2	2 3 4 5 6 1
3	1 4 2 3 5 6
4	2 5 3 4 6 1
5	3 6 4 5 1 2

ENERGY AND DIFF OF MACROITER	1 =	-228.10530547	-228.10530547
ENERGY AND DIFF OF MACROITER	2 =	-228.12683250	-0.02152703
ENERGY AND DIFF OF MACROITER	3 =	-228.12745850	-0.00062600
ENERGY AND DIFF OF MACROITER	4 =	-228.12748721	-0.00002871
ENERGY AND DIFF OF MACROITER	5 =	-228.12748904	-0.00000183
ENERGY AND DIFF OF MACROITER	6 =	-228.12748919	-0.00000015

OVERLAP OF VB ORBITALS FOR GROUP 2

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 3

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 4

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 5

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 6

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 7

```
      1      2
=====
1  1.00
2  0.89  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 8

```
      1      2
=====
1  1.00
2  0.84  1.00
```

OVERLAP OF VB ORBITALS FOR GROUP 9

```
      1      2
=====
```

```

1  1.00
2  0.84  1.00

OVERLAP OF VB ORBITALS FOR GROUP  10

      1      2
=====
1  1.00
2  0.84  1.00

OVERLAP OF VB ORBITALS FOR GROUP  11

      1      2
=====
1  1.00
2  0.84  1.00

OVERLAP OF VB ORBITALS FOR GROUP  12

      1      2
=====
1  1.00
2  0.84  1.00

OVERLAP OF VB ORBITALS FOR GROUP  13

      1      2
=====
1  1.00
2  0.84  1.00

OVERLAP OF VB ORBITALS FOR GROUP  14

      1      2      3      4      5      6
=====
1  1.00
2  0.51  1.00
3  0.06  0.51  1.00
4 -0.11  0.06  0.51  1.00
5  0.06 -0.11  0.06  0.51  1.00
6  0.51  0.06 -0.11  0.06  0.51  1.00

OVERLAP OF VB STRUCTURES FOR GROUP  14

      1      2      3      4      5
=====
1  1.00
2  0.62  1.00
3 -0.80 -0.80  1.00
4 -0.80 -0.80  0.70  1.00
5 -0.80 -0.80  0.70  0.70  1.00

Normalized structure coefficients
-0.4379 -0.4380  0.0797  0.0797  0.0797

```

```
====Mulliken Weight====
 0.3938  0.3939  0.0708  0.0708  0.0708

====Lowdin   Weight====
 0.2597  0.2598  0.1602  0.1602  0.1602

====Hiberty  Weight====
 0.4762  0.4765  0.0158  0.0158  0.0158

ENERGY AND DIFF OF MACROITER      7 =    -228.12748921      -0.00000002
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example13.txt>

8.14 A three-VB group description of a model system of $\text{Si}\equiv\text{Si}$ triple bond in Si_4H_6

```
#! VB(8).CASVB(6,6).VB(8)/6-31G* PRINTALL
```

```
TEST SI4H6 TRIPLE BOND
```

```
0 1
14 -0.8920 -0.5156 .000000
14 0.8920 0.5156 .000000
14 -3.1696 0.1200 .000000
14 3.1696 -0.1200 .000000
1 -3.2926 1.5924 .000000
1 -3.8226 -0.4408 -1.203300
1 -3.8226 -0.4408 1.203300
1 3.2926 -1.5924 .000000
1 3.8226 0.4408 -1.203300
1 3.8226 0.4408 1.203300
```

```
$MEMORY
9000000
```

```
$VBGA
(3-5) => 2
(3-6) => 2
(3-7) => 2
(1-3) => 2
(1#2) => 3
(4-8) => 4
(4-9) => 4
(4-10) => 4
(2-4) => 4
```

```
$03VBORB
(1-2)
1^
2^
1:
2:
```

```
$NOCANONPI
```

```
$CMAXCUT
0.001
```

One prominent difference between the $\text{Si}\equiv\text{Si}$ triple bond and the $\text{C}\equiv\text{C}$ triple bond is that the two terminal groups of the $\text{Si}\equiv\text{Si}$ triple bond are not co-linear as shown in the following figure:

The molecule is divided into three parts, the two -SiH₃ groups and the Si≡Si triple bond. The Si≡Si triple bond is described by CASVB(6,6), and the other two groups are described by VB(8). The CASVB provides the maximum flexibility for the description of the unusual triple bond, while the simple VB method is used for the 4 regular bonds in each -SiH₃ group.

The specification of the initial 6 VB orbitals of the Si≡Si triple bond is a little tricky. Owing to the bend structure, the bonds in the Si-Si link cannot be regarded as forming a normal triple bond. Thus, the method for generating initial VBOs by splitting the corresponding LMOs does not work well here. Therefore, we use VBO library and the **\$02VBORB** control flag to generate the 6 orbitals. In the **\$02VBORB** control, the first two VBOs are forming the Si-Si σ bond, as specified by (1-2). The next two lines specify the two π orbitals on the two central Si atoms. The last two specify the lone pairs on each central Si atom. Even though there are no lone pairs on any Si atom in this molecule, we use a virtual lone pair orbital to generate an initial VBO with proper orientation. VB2000 uses the lone pair orbital of sulfur as an approximation for the silicons. The VBOs make an angle with the central Si-Si link of roughly 120 degree.

This calculation has 88 basis function involved. The memory allocated by default does not allow a full in-core integral transformation. By allocating large memory with the flag **\$MEMORY**, the CPU time can be reduced nearly by half, and the elapsed time can be reduced significantly.

The HF energy and the three-VB group calculation energy result as follows:

```
.....  
Total number of two-electron integrals = 3284766  
Time for integral evaluation           15.651  
  
SKIPPED INTEGRALS= 3909291  
Number of iter = 26, DMAX =           0.000006705579  
Final Hartree-Fock Energy =          -1159.035780128051  
  
.....  
ENERGY AND DIFF OF MACROITER   12 =  -1159.20695689      -0.00000508  
  
.....
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example14.txt>

8.15 VB calculation of transition state of $[\text{Cl}\dots\text{CH}_3\dots\text{Cl}]^-$

```

#! VB(10)/6-31G* PRINTALL

TEST SN2 REACTION MECHANISM: CL- + CH3CL -> CLCH3 + CL-
THIS RUN GENERATE INITIAL GUESS FOR THE TRANSITION STATES AND REACTION

-1 1
6      0.000000      0.000000      0.000000
17     0.000000      0.000000     -2.000000
1      0.000000      1.061099      0.000000
1     -0.918939     -0.530549      0.000000
1      0.918939     -0.530549      0.000000
17     0.000000      0.000000      2.000000

$GRPDIM
17,9

$02VBORB
2->1
6->1
1->2 (PI)
1-3
1-4
1-5

$02VBSTR
2
1 1 2 3 4 5 6 7 8 9
2 2 1 3 4 5 6 7 8 9

$FULLHESS

```

The transition state is symmetric, and so can be easily optimized at any theoretical level. The optimized Cl...CH₃ distance at HF/6-31G* level is 2.38 Å. Since the HF method gives a very poor description for long weak bonds, we first use a much shorter Cl...CH₃ distance (2.0 Å) for the transition state: this gives LMOs from which the electron groups can be properly assigned. The VB orbitals generated in this calculation are used as the initial guess for the final calculation of Example 16. To provide a uniform description of the electronic structure of the reaction system from reactants to products, 10 electrons and 9 orbitals are used. In the transition state, each heavy atom provides an orbital for Cl...C...Cl bonding, and 6 orbitals for three C-H bonds. The central carbon provides a π orbital pointing to both Cl atoms. Two VB structures, which serve to describe the state of reactant and product, provide a smooth coverage of the whole reaction path.

Output:

```

.....

VB-ORBITAL OPTIMIZATION OF GROUP 2 ...
GROUP = 2 ITER = 1 CMAX =0.00020780700704

```

OVERLAP OF VB ORBITALS FOR GROUP 2

	1	2	3	4	5	6	7	8	9
1	1.00								
2	0.06	1.00							
3	-0.59	0.59	1.00						
4	-0.19	-0.19	0.00	1.00					
5	-0.10	-0.10	0.00	0.83	1.00				
6	-0.19	-0.19	0.00	0.24	0.13	1.00			
7	-0.10	-0.10	0.00	0.13	0.05	0.83	1.00		
8	-0.19	-0.19	0.00	0.24	0.13	0.24	0.13	1.00	
9	-0.10	-0.10	0.00	0.13	0.05	0.13	0.05	0.83	1.00

Normalized structure coefficients

0.5371 -0.5371

====Mulliken Weight====

0.5000 0.5000

====Lowdin Weight====

0.5000 0.5000

====Hiberty Weight====

0.5000 0.5000

ENERGY AND DIFF OF MACROITER 7 = -958.61629512 -0.00000014

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example15.txt>

8.16 Energy profile of the S_N2 reaction of Cl⁻ + CH₃Cl → ClCH₃ + Cl⁻

```

#! VB(10)/6-31G* GUESS=READ PRINTALL

TEST SN2 REACTION MECHANISM: CL- + CH3CL -> CLCH3 + CL-
TWO STRUCTURES ARE USED FOR C1-C-CL BONDS: C1-C :C1 + C1: C-CL

-1 1
6          0.000000      0.000000      0.000000
17         0.000000      0.000000     -2.383018
1          0.000000      1.061099      0.000000
1         -0.918939     -0.530549      0.000000
1          0.918939     -0.530549      0.000000
17         0.000000      0.000000      2.383018

$REACTION
10,2,6
GEOM1
6          0.000000      0.000000      0.000000
17         0.000000      0.000000     -2.383018
1          0.000000      1.061099      0.000000
1         -0.918939     -0.530549      0.000000
1          0.918939     -0.530549      0.000000
17         0.000000      0.000000      2.383018
GEOM2
6          0.000000      0.000000     -0.591622
17         0.000000      0.000000     -2.419329
1          0.000000      1.020360     -0.259950
1         -0.883658     -0.510180     -0.259950
1          0.883658     -0.510180     -0.259950
17         0.000000      0.000000      2.674010

$GRPDIM
17,9

$RESTARTFILE
'example15.V84'

$02VBSTR
2
1 1 2 3 4 5 6 7 8 9
2 2 1 3 4 5 6 7 8 9

```

This calculation uses the previous one (Example 15) as a starting point. Since the reaction is symmetric, only half of the reaction path needs to be computed. The computed energy profile is given below:

.....

Energy profile

1	2	3	4	5	6
-958.66836	-958.67141	-958.67366	-958.67655	-958.67988	-958.68347
7	8	9	10	11	12
-958.68715	-958.69072	-958.69398	-958.69667	-958.69850	-958.69911

```
1| *****
2| *****
3| *****
4| *****
5| *****
6| *****
7| *****
8| *****
9| *****
10| ***
11|
12|
```

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example16.txt>

8.17 Visualization of VB orbitals of H₂O

```
#! VB (4) /STO-3G
```

```
Water
```

```
0 1
8 0.0000000000 0.0000000000 0.0000000000
1 0.0000000000 -0.7572153434 0.5865355237
1 0.0000000000 0.7572153434 0.5865355237
```

```
$CUBE
```

```
4
```

```
4 5 6 7
```

```
Water
```

```
50
```

```
$XYZFILE
```

In this example, the electrons are described by 3 Hartree-Fock orbitals and 4 VB orbitals. The orbitals 4-7 are VB orbitals. Four cube files for the VB orbitals are created. An xyz file is also created.

Output:

```
CREATED XYZ FILE OF MOLECULAR COORDINATES
```

```
CUBE FILE HEADERS
```

```
#####
```

```
Water
```

```
SCF Molecular Orbitals
```

```
-3 -6.000000 -7.430930 -6.000000
50 0.244898 0.000000 0.000000
50 0.000000 0.303303 0.000000
50 0.000000 0.000000 0.267518
8 8.000000 0.000000 0.000000 0.000000
1 1.000000 0.000000 -1.430930 1.108391
1 1.000000 0.000000 1.430930 1.108391
1 4
```

```
#####
```

```
CUBE FILE FOR ORBITAL 4 CREATED AS example17-004.cube
```

```
CUBE FILE FOR ORBITAL 5 CREATED AS example17-005.cube
```

```
CUBE FILE FOR ORBITAL 6 CREATED AS example17-006.cube
```

```
CUBE FILE FOR ORBITAL 7 CREATED AS example17-007.cube
```

See Figure 1 and 2 for orbital 4 and 5, respectively, in Chapter 6.

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example17.txt>

8.18 Calculation with dynamic VB orbitals

```
#! VB(3)/D95 SPDEN DIIS PRINTALL

TEST

0 2
1      0.0  0.0  1.0
1      0.0  0.0  0.0
1      0.0  0.0 -1.0

$GRPDIM
4

$01VBORB
(1->2)
(2->1)+0.50(1->2)
(2->3)+0.50(3->2)
(3->2)

$01VBSTR
2
1 2 4
3 4 1
```

In this example, two VB structures are included in the calculation and each structure has a different set of VB orbitals. Such a treatment is quite similar to the BOVB type calculation. However, the original BOVB[17] approach has additional constraints that each orbital has to be “strictly” localized on its parent atom. In this calculation, such constraints are removed. In version 1.8.2, significant algorithm enhancement has been made for the orbital optimization. The optimization can be performed even in the case where the VB orbitals have linear dependency.

Output:

```
Normalized structure coefficients
0.5747 -0.5747

====Mulliken Weight====
0.5000 0.5000

====Lowdin Weight====
0.5000 0.5000

====Hiberty Weight====
0.4999 0.5001
```

SPIN DENSITY ON ATOMS

ATOM	DENSITY
1	0.62723
2	-0.25443
3	0.62720

LINEAR DEPENDENT VECTOR IDENTIFIED

ENERGY AND DIFF OF MACROITER 5 = -1.61408272 -0.00000002

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

Orbital#	1	2	3	4
Group#	1	1	1	1
AO# LABELS				
1 1 H s	0.43527	0.14243	-0.02468	0.00885
2 1 H s	0.60017	0.11722	-0.16583	-0.12214
3 2 H s	0.11269	0.42154	0.42154	0.11270
4 2 H s	0.03829	0.61032	0.61032	0.03829
5 3 H s	0.00881	-0.02469	0.14242	0.43527
6 3 H s	-0.12219	-0.16585	0.11722	0.60015

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example18.txt>

8.19 VB calculation with strictly localized orbitals

```
#! VB(9)/D95 DIIS PRINTALL

TEST H O

0 2
8      .000000 .000000 .000000
1      1.00000 .000000 .000000

$GRPDIM
12

$AOGROUP
12
1,2,3,4,5,6,7,8,9,10,11,12
1,1,1,1,1,1,1,1,1,1,1,1

$01VBSTR
3
1 1 3 3 5 11 6 6 7
1 1 3 3 5 5 6 6 7
1 1 3 3 11 11 6 6 7

$BRILLMASK
20
1,11
1,12
3,11
3,12
5,11
5,12
6,11
6,12
7,11
7,12
11,1
11,2
11,3
11,4
11,5
11,6
11,7
11,8
11,9
11,10

$01VBSCF
0.00001,30
```

In this example, the two atoms of OH are put on x-axis. Such an orientation makes it more convenient to divide the atomic orbitals into different groups. The 12 atomic orbitals of OH in D95 basis set are shown as follows:

O: 1S, 1S', 2S, 2S', 2Px, 2Py, 2Pz, 2Px', 2Py', 2Pz' (basis function 1-10)

H: 1S, 1S' (basis function 11,12)

Three structures, which corresponds to the following VB structures, are included in the calculation:

1S² 2S² 2Px-1S(H) 2Py² 2Pz¹

1S² 2S² 2Px² 2Py² Pz¹

1S² 2S² 1S²(H)2Py² 2Pz¹

The subscript for each AO specifies the number electrons on the AO. 1S(H) is the 1S orbital of the hydrogen atom. All other orbitals are from O. 2Px-1S(H) specifies the O-H bond formed by 2Px(O) and 1S(H). According to this convention, the first structure corresponds to a covalent OH bond with a negative charge on O. The second structure corresponds to a ionic structure with two negative charges on O, and the third one corresponds to a ionic structure with a negative charge on H. The initial VB orbitals are from atomic orbitals as shown in the above VB structures (the AO on H are explicitly specified in parenthesis), therefore they are all strictly localized on their corresponding atoms.

In a general VB calculation, each VB orbital is optimized within the whole basis function space, therefore the final resulting VB orbitals are somewhat delocalized to neighbor atoms. In this case, we want to preserve the strict localization while still allow each VB orbital to be optimized within the atomic orbitals of their corresponding atoms. This is controlled by \$BRILLMASK (Brillouin Mask). In this case, there are 20 lines of controls, each line specify which orbital will NOT be mixed with which AO. For instance, the first line is

1,11

which means the VB orbital *1* will NOT be mixed with AO *11*, which is *1S(H)*, during orbital optimization.

The final optimized VB orbitals are strictly localized on their corresponding atoms as shown below:

Output:

Normalized structure coefficients

0.6602 0.4025 0.0524

====Mulliken Weight====

0.6300 0.3394 0.0307

====Lowdin Weight====

0.4869 0.3950 0.1181

====Hiberty Weight====

0.7257 0.2698 0.0046

ENERGY AND DIFF OF MACROITER 7 = -75.40523919 -0.00000051

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)

				=====					
Orbital#				1	2	3	4	5	6
Group#				1	1	1	1	1	1

AO#	LABELS								
1	1	O	s	0.59799	0.00000	0.24697	0.00000	-0.03211	0.00000
2	1	O	s	0.49327	1.00000	0.12964	0.00000	-0.02054	0.00000
3	1	O	s	-0.12697	0.00000	0.45371	0.00000	-0.03171	0.00000
4	1	O	s	-0.11131	0.00000	0.40523	1.00000	-0.17957	0.00000
5	1	O	x	0.03636	0.00000	-0.12557	0.00000	0.79241	0.00000
6	1	O	y	0.00000	0.00000	0.00000	0.00000	0.00000	0.74993
7	1	O	z	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
8	1	O	x	0.02270	0.00000	-0.08372	0.00000	0.29449	0.00000
9	1	O	y	0.00000	0.00000	0.00000	0.00000	0.00000	0.38316
10	1	O	z	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
11	2	H	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
12	2	H	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
				=====					
Orbital#				7	8	9	10	11	12
Group#				1	1	1	1	1	1

AO#	LABELS								
1	1	O	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	1	O	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	1	O	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	1	O	s	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	1	O	x	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	1	O	y	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	1	O	z	0.78288	0.00000	0.00000	0.00000	0.00000	0.00000
8	1	O	x	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
9	1	O	y	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000
10	1	O	z	0.34144	0.00000	0.00000	1.00000	0.00000	0.00000
11	2	H	s	0.00000	0.00000	0.00000	0.00000	0.74963	0.00000
12	2	H	s	0.00000	0.00000	0.00000	0.00000	0.32476	1.00000

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example19.txt>

8.20 VB calculation of structure weights with localization enhanced VB orbitals

```
#!/ VB(4)/cc-pVDZ printall

TEST H2O

0 1
8 .000000 .000000 .000000
1 .801842 .000000 .555582
1 -.801842 .000000 .555582

$02VBORB
1-2
1-3

$02LENHANCE
4,0.1
1,1,1
2,1,2
3,1,1
4,1,3

$02VBSTR
5
1 2 3 4
1 1 3 4
3 3 1 2
2 2 3 4
4 4 1 2

$MACROITER
30

$CUBE
4
4 5 6 7
water local
64
```

In this case, 5 structures are included for 4-electron VB calculation of the two OH bonds in H₂O. We want to calculate the structure weights for the covalent structure and the four ionic structures. The delocalization of VB orbitals lead to some ambiguity for the distinction between covalent structures and ionic structures. More over, there is also a mathematical problem for computing the weights of ionic structures. Since the VB orbitals are optimized, the formal contribution of singly excited ionic structures to the fully optimized VBSCF wave function can be zero as shown by Brillouin theorem. Therefore, we need a more convenient way to obtain optimized VB orbitals with

localization enhancement. This will not only solve the mathematical problem for the computation of ionic structure weights but also reduce the ambiguity between covalent and ionic structures in modern VB descriptions of a molecular system. The strict localization method in example 19 is cumbersome to use and is difficult to be applied for more general cases. The localization enhancement control provides a flexible and convenient method for the balance of total wave function energy and localization.

The localization enhancement control is specified by the key word \$02LENHANCE. By default, the 4-electron VB group is group 2, therefore the LENHANCE flag has a number prefix \$02. The 4 VB orbitals as specified by \$02VBORB are

- (1) orbital on O pointing to the first H
- (2) orbital on the first H
- (3) orbital on O pointing to the second H
- (4) the orbital on the second H

In this example, each VB orbital is localized on one parent atom, i.e. the four VB orbitals 1-4 are localized on atom 1,2,1 and 3 respectively.

The delocalization quantities and structure weights from the optimized VB orbitals are shown as follows:

Output:

```
Normalized structure coefficients

DELOCALIZATION FOR VB ORBITALS OF GROUP 2
=====
VBO INDEX = 1 DELOCALIZATION = 0.00613
VBO INDEX = 2 DELOCALIZATION = 0.00960
VBO INDEX = 3 DELOCALIZATION = 0.00613
VBO INDEX = 4 DELOCALIZATION = 0.00960

OVERLAP MATRIX OF VB ORBITALS FOR GROUP 2

      1      2      3      4
=====
1  1.00
2  0.69  1.00
3  0.26  0.18  1.00
4  0.18  0.15  0.69  1.00

Normalized structure coefficients
-0.4476 -0.3170 -0.3170 -0.0067 -0.0068

====Mulliken Weight====
```

```

0.4316  0.2794  0.2794  0.0049  0.0049

====Lowdin   Weight====
0.2487  0.2799  0.2799  0.0958  0.0958

====Hiberty  Weight====
0.4991  0.2503  0.2503  0.0001  0.0001

ENERGY AND DIFF OF MACROITER    15 =      -76.06632554      0.00000708

ORBITALS OF EACH ELECTRON GROUP IN AO BASIS: ORBITAL(GROUP)
=====
Orbital#           1           2           3           4           5           6
Group#            1           1           1           2           2           2
-----
AO# LABELS
1  1  O      s  -0.89422  -0.44963  0.00001  0.04922  -0.04582  0.04921
2  1  O      s   0.16312  -0.30810  0.00000  0.37141  -0.01430  0.37141
3  1  O      s   0.22775  -0.43468  0.00000  0.22621  -0.04715  0.22621
4  1  O      x   0.00000  0.00000  0.00000  0.45537  -0.00568  -0.45537
5  1  O      y   0.00000  0.00001  0.62988  0.00000  0.00000  0.00000
6  1  O      z  -0.19405  0.39397  0.00000  0.38727  0.02190  0.38727
7  1  O      x   0.00000  0.00000  0.00000  0.18788  0.02334  -0.18787
8  1  O      y   0.00000  0.00001  0.49906  0.00000  0.00000  0.00000
9  1  O      z  -0.14783  0.29980  0.00000  0.19622  0.03555  0.19623
10 1  O     xx   0.00130  0.00030  0.00000  0.01070  -0.00386  0.01070
11 1  O     yy   0.00345  -0.00428  0.00000  -0.02443  0.02826  -0.02443
12 1  O     zz  -0.00706  0.01722  0.00000  0.00743  0.01472  0.00743
13 1  O     xy   0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
14 1  O     xz   0.00000  0.00000  0.00000  0.04166  -0.02585  -0.04166
15 1  O     yz   0.00000  0.00000  0.01729  0.00000  0.00000  0.00000
16 2  H      s  -0.02975  0.07202  0.00000  0.07441  0.87485  -0.00894
17 2  H      s  -0.01494  0.03046  0.00000  0.00111  0.17052  -0.00468
18 2  H      x   0.00133  -0.00559  0.00000  -0.01052  -0.09316  -0.01357
19 2  H      y   0.00000  0.00000  0.02974  0.00000  0.00000  0.00000
20 2  H      z  -0.00829  0.01519  0.00000  -0.00392  -0.04626  0.00253
21 3  H      s  -0.02975  0.07202  0.00000  -0.00894  0.01618  0.07441
22 3  H      s  -0.01495  0.03046  0.00000  -0.00467  -0.04281  0.00111
23 3  H      x  -0.00133  0.00559  0.00000  0.01357  0.02338  0.01052
23 3  H      x  -0.00133  0.00559  0.00000  0.01357  0.02338  0.01052
24 3  H      y   0.00000  0.00000  0.02974  0.00000  0.00000  0.00000
25 3  H      z  -0.00829  0.01519  0.00000  0.00253  0.02093  -0.00392
=====
Orbital#           7
Group#            2
-----
AO# LABELS
1  1  O      s  -0.04583
2  1  O      s  -0.01431
3  1  O      s  -0.04715
4  1  O      x   0.00568
5  1  O      y   0.00000
6  1  O      z   0.02190
7  1  O      x  -0.02334
8  1  O      y   0.00000
9  1  O      z   0.03555
10 1  O     xx  -0.00386
11 1  O     yy   0.02826

```

12	1	O	zz	0.01473
13	1	O	xy	0.00000
14	1	O	xz	0.02585
15	1	O	yz	0.00000
16	2	H	s	0.01618
17	2	H	s	-0.04281
18	2	H	x	-0.02338
19	2	H	y	0.00000
20	2	H	z	0.02093
21	3	H	s	0.87485
22	3	H	s	0.17052
23	3	H	x	0.09316
24	3	H	y	0.00000
25	3	H	z	-0.04626

The structure weights of the covalent structure and the four ionic structures are shown as follows:

Table 8.1. Structure weights from localization enhanced VB calculation.

<i>Structure type</i>	<i>VB structure</i>	<i>Weight*</i>
I (covalent)	H–O–H	0.432
II (ionic)	H–O [−] H ⁺	0.279
III (ionic)	H ⁺ O [−] –H	0.279
IV (ionic)	H–O ⁺ H [−]	0.005
V (ionic)	H [−] O ⁺ –H	0.005

* Mulliken analysis

The delocalization of each VB orbital is below 0.01 as shown in the following table:

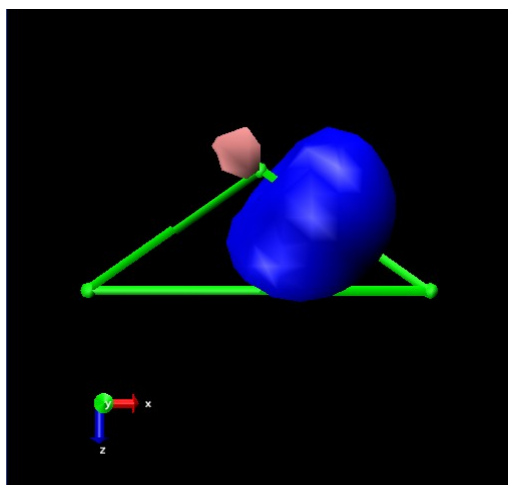
Table 8.2. Delocalization of VB orbitals in localization enhanced calculation.

<i>VB orbital</i>	<i>Location</i>	<i>Delocalization*</i>
1	O(1)	0.0061
2	H(2)	0.0096
3	O(1)	0.0061
4	H(3)	0.0096

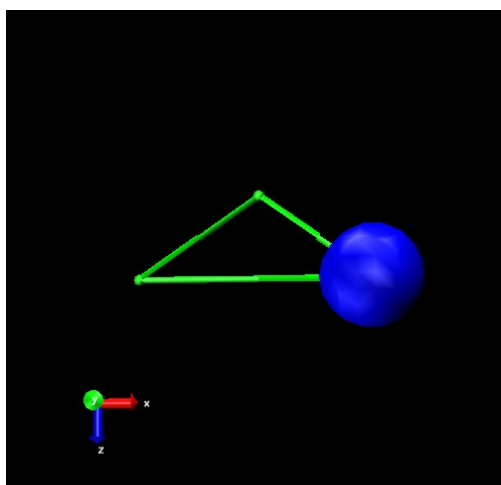
* Mulliken analysis

In a VB structure calculation with only one covalent structure, the resulting VB orbital on hydrogens are considerably delocalized with a delocalization coefficient 0.16. The delocalization is reduced by a factor of 16 with the localization enhanced optimization.

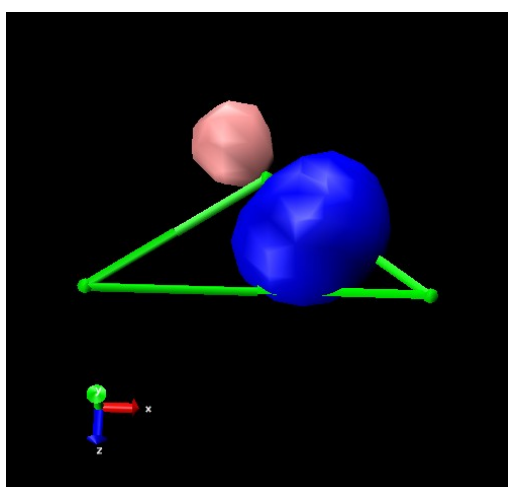
The VB orbitals of localization enhanced optimization and the regular optimization are shown as follows:



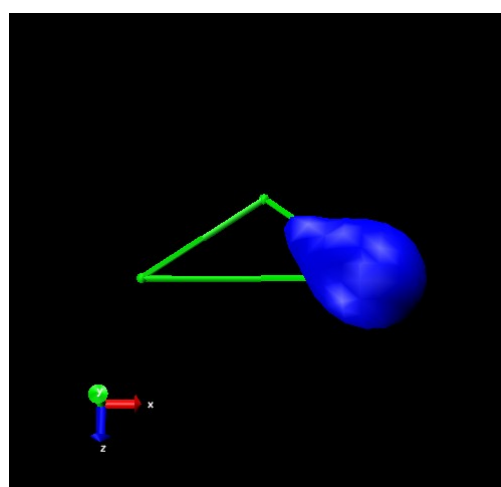
8.1a



8.1b



8.1c



8.1d

Figure 8.1 The optimized VB orbitals from localization enhanced calculation (8.1a, 8.1b) and the VB orbitals from a normal single covalent VB calculation for H₂O.

The log file can be found at <http://www.scinetec.com/~vb/VB2000V18/example20.txt>

9 Acknowledgments

VB2000 is based on the extensive re-engineering of two independent programs AMOY-VB and VB98 (both were not publicly distributed). The former one was originated from the early investigation on group theoretical approach to VB calculations during one of the authors (JL)'s stay at Xiamen University (1990 -1992). He would like to express his thanks to Professor Qianer Zhang for teaching him the spin-free approach to VB theory in early 1990s. He is also very thankful to his former colleagues, Dr. Wei Wu and Dr. Yirong Mo at Xiamen University, for the early development of permanent method and for VB calculations (1991-1992). Since the early days of the program, many other people have been contributed to the development of VB2000 in one way or another. JL is very grateful to Professor Ruben Pauncz for the very inspiring collaboration on developing the algebrant algorithm(1995-1997), which is one of the mathematical foundations of VB2000. Sincere thanks are due to Dr. David L. Cooper for introducing him to the use of the Pipek-Mezey (PM) localization method for electron group partitioning(1995), which is the technical foundation of VB2000. He also thanks for Professor Keiji Morokuma for his hospitality during the author's 6-month stay at Institute of Molecule Science, Okazaki, Japan, 1992-1993. During this period, Professor Morokuma offered complete freedom for doing research on VB algorithm development. JL is grateful to the Alexander von Humboldt Foundation for the award of a fellowship to make his 2-year stay in Germany possible. He is also very grateful to Professor Janos Ladik for providing a position as an AvH Research Fellow at the Institute of Physical and Theoretical Chemistry, University of Erlangen-Nürnberg, Erlangen, Germany, 1993-1995. Thanks are also due to Professor Donald G. Truhlar for reading the manual of the VB program during his stay in the Truhlar-Cramer group (1996-1999).

Since the first release of VB2000 in the new Millennium, a lot of feedbacks have been received from many enthusiastic users. Their excellent comments are a strong motivation for us to develop an even more efficient, more flexible and easy to use VB program. In particular, we would like to thank Professor Thomas M. Klapötke of University of Munich for his creativity in using VB2000: he has provided us with many examples which challenge our program. JL thanks Drs. Heribert Reis, Aggelos Avramopoulos and Professor Manthos G. Papadopoulos for many years collaboration on applying VB2000 to novel materials. We thank Saryu Jindal, Felipe Fleming and Nuno A. G. Bandeira for their feedbacks on the use of the program.

We thank Dr Rika Kobayashi at the National Computational Infrastructure (NCI) hosted at the Australian National University for compiling the Gaussian03/VB2000 and Gaussian09/VB2000 version, and, with particular thanks, for her help and advice with the code changes between Gaussian03 and Gaussian09. We thank Michelle Styles and Harry Quiney, both of the University of Melbourne, for the initial Windows .NET GUI and the first compilation of VB2000 for Windows respectively (with version 1.6). One of the

authors (BD) thanks Professor Jack Linnett for introducing him to Valence Bond theory over 40 years ago and to Dick Harcourt of the University of Melbourne for many recent stimulating discussions.

One of the authors (RMcW) wishes to record his indebtedness to Wei Wu for introducing him to the algorithms under development at Xiamen University, during a fruitful collaboration at the University of Pisa (1993-94).

10 References

VB2000 Algorithms

1. a) Jiabo Li and Roy McWeeny, VB2000: Pushing Valence Bond Theory to New Limits, *Int. J. Quant. Chem.*, 89 (2002) 208. b) Jiabo Li, Brian J. Duke, Thomas M. Klapötke and Roy McWeeny, Spin-Density of Spin-Free Valence Bond Wave Functions and Its Implementation in VB2000, *J. Theor. Comp. Chem.* 7 (2008) 853.
2. Jiabo Li, and Ruben Pauncz, Efficient Evaluation of the Algebrants of VB Wave Functions Using the Successive Expansion Method. I: Spin $S = 0, 1/2$. *Int. J. Quant. Chem.*, 62 (1997) 245.
3. Jiabo Li, Graphical Representation of a New Algorithm for Nonorthogonal Ab Initio Valence Bond Calculations, *Theor. Chim. Acta*, 93 (1996) 35.
4. Jiabo Li, New Algorithm for Nonorthogonal Ab Initio Valence Bond Calculations II: Subgraph-Driven Method, *J. Math. Chem.*, 17 (1995) 295.
5. Jiabo Li, and Wei Wu, New Algorithm for Nonorthogonal Ab Initio Valence Bond Calculations. I: New Strategy and Basis Expressions, *Theor. Chim. Acta*, 89 (1994) 105.

Group Function Theory

6. Roy McWeeny, The Density Matrix in Many-Electron Quantum Mechanics I. Generalized Product Functions. Factorization and Physical Interpretation of the Density Matrices, *Proc. R. Soc. London, Ser A* 253(1959) 242.
7. János G. Ángyán, Chemical Building Blocks in Quantum Chemistry Calculations. Perspective on “The Density Matrix in Many-Electron Quantum Mechanics I. Generalized Product Functions. Factorization and Physical Interpretation of the Density Matrices”, *Theor. Chem. Acc.* 103 (2000) 238.
8. Roy McWeeny, Separability of Quantum Systems: A Density Matrix Approach, *Adv. Quantum Chem.*, 31 (1999) 15.

Applications of VB2000

9. Aggelos Avramopoulos, Herbert Reis, Jiabo Li, and Manthos G. Papadopoulos, The Dipole Moment, Polarizabilities, and the First Hyperpolarizabilities of HArF. A Computational and Comparative Study, *J. Am. Chem. Soc.*, 126 (2004) 6179.
10. Anton Hammerl, Thomas M. Klapötke, Heinrich Nöth, and Markus Warchhold, Synthesis, Structure, Molecular Orbital and Valence Bond Calculations for Tetrazole Azide, CHN₇, *Propellants, Explosives, Pyrotechnics* 28 (2003) 165.
11. Thomas M. Klapötke, Jiabo Li, and Richard D. Harcourt, Ab initio Double- ζ (D95) Valence Bond Calculations for the Ground States of S₂N₂ and S₄²⁺, *J. Phys. Chem.*, 108 (2004) 6527.

12. Thomas M. Klapötke, Richard D. Harcourt, and Jiabo Li, Ab initio Double- ζ (D95) Valence Bond Calculations for the Ground States of NO₂, O₃ and ClO₂, *Inorg. Chim. Acta*, 358(2005)4131.
13. a) Daniel Benker, Thomas M. Klapötke, Gerhard Kuhn, Jiabo Li, and Christian Miller, An Ab Initio Valence Bond (VB) Calculation of the Aromatic Stabilization Energy in Borazine, B₃N₃H₆, *Heteroatom Chem.*, 16(2005)311. b) A. Avramopoulos, L. Serrano-Andrés, J. Li, H. Reis, M. G. Papadopoulos, *J. Chem. Phys.*, 127 (2007) 214102. c) L. Serrano-Andrés, A. Avramopoulos, J. Li, P. Labeguerie, D. Begue, V. Kello and M. G. Papadopoulos, Linear and nonlinear optical properties of a series of Ni-dithiolene derivatives, *J. Chem. Phys.*, 231(2009) 134312.

Structure Weight Analysis

14. B. H. Chirgwin, and C. A. Coulson, *Proc. R. Soc. Lond., Ser. A* 201 (1950) 196.
15. P. O. Löwdin, *J. Mol. Struct. (THEOCHEM)* 229 (1991) 1.
16. P. C. Hiberty, G. Ohanessian, *Int. J. Quantum Chem.*, 27(1985)245.

BOVB Theory

17. Philippe C. Hiberty and Sason Shaik, BOVB - A Modern Valence Bond Method that Includes Dynamic Correlation, *Theor. Chem. Acc.* 108 (2002) 255.

ORMAS (Occupation Restricted Multiple Active Space)

18. Joseph Ivanic, CI and MCSCF Method for Multiple Active Spaces with Variable Occupations. I. Theory. *J. Chem. Phys.* 119 (2003) 9364.
19. Joseph Ivanic, Direct CI and MCSCF Method for Multiple Active Spaces with Variable Occupations. II. Application to oxoMn(salen) and N₂O₄. *J. Chem. Phys.* 119 (2003) 9377.

VMD (Visual Molecular Dynamics)

20. W. Humphrey, A. Dalke and K. Schulten, VMD - Visual Molecular Dynamics, *J. Mol. Graphics*, 14(1996)33.