

Logiciels de chimie quantique

Décembre 2008

Laboratoire de Chimie et Physique Quantiques

Logiciels de chimie quantique toulousains

I. Présentation générale	2
II. SCF, CASSCF, interactions de configurations et méthodes perturbatives.....	2
III. Utilitaires	4
IV. Exemples de performances	4
V. Exemple de script	5
VI. Applications	7
A. Etude du tridequenal : Transition $n \rightarrow \pi^*$	7
Manuels d'utilisation.....	12
I. Molcost.....	13
A. What does the program?.....	13
B. Data.....	13
C. Aspect of the Info file (example of ethylene molecule) ("prefix"Info).....	13
D. How to read the unformatted Mono file ("prefix"Mono).....	14
II. DOLO (DO Local Orbitals) : programme de localisation à priori	15
A. Preliminaires.....	15
B. Data.....	15
C. How it works	16
D. Possible Data	17
E. Details of Data	22
III. Proj_scf et Schudort	22
IV. Noscfc	24
V. Casdi.....	25
A. Fichiers d'entree et de sortie ou Comment se raccrocher au reste du monde... ..	27
B. Fichiers de donnees	28
C. Vecteurs d'essai	33
D. Details techniques.....	34
E. Calculs sc2	35
VI. Ijkloc	36
VII. Casdiloc	37
VIII. Naturalt et locnats.....	40
IX. Internat	41
X. Orb2q5.....	43
XI. Q52molden.....	43

Logiciels de chimie quantique toulousains

Depuis longtemps impliqué dans le développement de méthodes et de codes ab initio, le laboratoire de Chimie et Physique Quantiques de Toulouse a acquis un certain savoir-faire dans plusieurs domaines tels que les interactions de configurations, les méthodes perturbatives, de Monte Carlo Quantiques, etc... Depuis quelques années, les méthodes et codes développés permettent l'utilisation d'orbitales localisées afin de profiter des avantages d'une description localisée, aussi bien à la compréhension de la physique qu'à la réduction très importante des coûts calculatoires.

Un grand nombre de codes a donc été développé. Certains sont brièvement présentés ci-dessous. L'ensemble de ces programmes est rattaché au Q5COST.

I. Présentation générale

Le répertoire **cost** contient :

- le fichier *install* avec les instructions nécessaires pour une installation de l'ensemble des logiciels liés aux interactions de configurations.
- les sous-répertoires :
 - manual contient le mode d'emploi des différents codes
 - bin contient tous les exécutables
 - source, lib, etc...

II. SCF, CASSCF, interactions de configurations et méthodes perturbatives

- **DOLO** (DO Local Orbitals) : permet de créer des *orbitales locales*
 - de liaison,
 - atomiques,
 - correspondant à des doublets,
 - ou encore des orbitales diffuses.

Un lancement en interactif (`idolo` ou `dolo3d.py`) offre la possibilité d'une interface graphique (`dolo3d.py`) qui permet de visualiser les orbitales au cours de leur définition.

Bien que de plus en plus simple, les données sont encore un peu difficiles à générer. L'utilisation du manuel (dolo_emploi) reste indispensable pendant un certain temps.

- **LOC-CASSCF**: développé dans le cadre du projet PICS (Développement et applications de méthodes ab initio de Chimie Quantique) en collaboration avec l'équipe de Ferrare (Italie), ce code permet d'optimiser le jeu d'orbitales localisées (ou délocalisées) pour arriver à la solution SCF ou CASSCF.
- **NOSCF (Natural Orbital SCF)** permet d'optimiser le jeu d'orbitales localisées (ou délocalisées) pour arriver à la solution SCF ou quasi-CASSCF. Il utilise la transformation des intégrales atomiques en moléculaires de motra de la chaîne molcas. Il est couramment employé depuis longtemps.
- **CASDI**: Programme d'interactions de configurations multiréférentielles (MR ou CAS) des simples et doubles dont l'originalité est de pouvoir utiliser les méthodes DDCI (Difference Dedicated CI) et DDCI2 particulièrement adaptées à l'étude de systèmes magnétiques mais également de proposer les corrections de l'erreur de size-consistence (SC)², MR-ACPF et MR-AQCC. Ce programme permet la diagonalisation d'une centaine de millions de déterminants.
- **EXSCI (Excitation Selection CI, souvent appelé CASDILOC)**: Programme d'interactions de configurations multiréférentielles (MR ou CAS) des simples et doubles. Tout comme *CASDI* ce programme propose les méthodes DDCI et une correction de l'erreur de size-consistence CD-MRCI (Class-Dressed). Cependant, sa grande originalité est de tirer avantage de la localité de la corrélation électronique afin de réduire la taille de l'espace de déterminants à diagonaliser sans pour autant perdre au niveau de la physique. Ce type de méthode est avant tout destiné à l'étude de systèmes de grande taille qui ne pourrait être menée par un MRCI classique. Réduire la dimension de l'espace d'IC se fait -généralement mais pas uniquement- à l'aide d'un seuil sur la valeur de l'intégrale d'échange.

III. Utilitaires

Il existe également un certain nombre d'utilitaires, dont la description complète se trouve dans le répertoire « manuel »:

- *OLDORB* ou *NEWORB*, sans données, ces 2 codes permettent de transformer les orbitales du nouveau format *MOLCAS* à l'ancien et vice-versa. Ces deux codes sont maintenant intégrés dans les autres. Ils ne devraient plus être utilisés directement.
- *FAIANO*: permet de développer n'importe quelle base pour lui donner une forme d'ANO.
- *LOCNATS*: diagonalisation par bloc de la matrice densité permettant de moyennner des orbitales (par exemple de différente symétrie d'espace et/ou de spin) ou d'autres types d'opérations à définir. C'est ce programme qui est appelé dans les codes *LOC-CASSCF* et *NOSCF* définis précédemment.
- *Q52MOLDEN*: permet de créer un fichier d'orbitales pour molden. Nécessite l'utilisation d'un fichier Q5FILE.

IV. Exemples de performances

CASDI: Calcul DDCI en symétrie D_{2h} de 139 millions de déterminants en 8000 secondes par itération sur une machine 64G mémoire, (sv8). A titre indicatif, le calcul converge en 11 itérations. Ce calcul est le plus « gros » par le nombre de déterminants de l'espace de diagonalisation réalisé à ce jour avec ce code.

EXSCI: Ce programme relativement récent n'est pas encore très performant et sera encore optimisé. Cependant, il permet d'ores et déjà l'étude de systèmes dont les dimensions excèdent les capacités d'un code d'interaction de configurations plus classique tel que **CASDI** ou les MRCI de **MOLPRO** et **MOLCAS**. Un calcul CAS-DDCI sur un système métallique sans symétrie avec 332 fonctions de base et un CAS de 5 électrons dans 5 orbitales génère 184 millions de déterminants. L'application d'un seuil de 0.003 u.a. sur les intégrales d'échange réduit ce nombre à moins de 12 millions, et le calcul converge en 7 itérations, avec 7140 secondes par itération sur une machine (sv11, moins efficace que sv8).

V. Exemple de script

Le système étudié est le tridequenal, voir dans la partie applications.

```
#PBS -S /bin/bash
#PBS -j oe
#PBS -l ncpus=1
#PBS -l mem=500mb
#PBS -l cput=1:30:00
#PBS -q molcas

. $HOME/.bashrc

export MOLCAS=/usr/local/molcas67
export WorkDir=/tmp/nadia/Tridequenal
export CurrDir=$PWD          # Interactif
export CurrDir=$PBS_O_WORKDIR # En queue
mkdir /tmp/nadia
mkdir $WorkDir
cd $WorkDir

PGM=/home/daniel/cost2/bin
export PATH=$PATH:.$PGM

export Project=tridequenal
export MOLCASMEM=1600
export MOLCASDISK=0          # limite de 200 Gb avec =0
alias molcas=$MOLCAS/shell/molcas.sh

touch core
chmod -w core
```

**MOLCAS = seward (intégrales atomiques)
scf**

```
echo "Workdir="$WorkDir
echo "CurrDir="$CurrDir

molcas seward $CurrDir/input.molcas > $CurrDir/out.molcas
cp $CurrDir/Orb-deloc-SCF INPORB
molcas scf $CurrDir/input.molcas >> $CurrDir/out.molcas
cp $Project.ScfOrb $Project.ScfOrbv
```

MOLCAS2Q5 ET Q52CASDI

```
cp $Project.ScfOrbv INPORB
cp $Project.ScfOrbv $Project.ScfOrb
molcas $CurrDir/input.q5 > $CurrDir/out.q5
mv Q5FILE $Project.q5
q52casdi < $CurrDir/input.q5 >> $CurrDir/out.q5
```

← crée les fichiers
tridequenal.Info et
tridequenal.Mono

LOCALISATION

1) construire les orbitales locales

Fichiers en entrée = DOLOIN, INPORB, tridequenal.Info et tridequenal.Mono
En interactif = idolo ou dolo3d.py (graphique) : exporter Project,
CurrDir, WorkDir et PATH=\$PATH:/chemin_des_programmes/bin

```
cp $CurrDir/DOLOIN .
cp $CurrDir/OLD_DATA .
dolo > $CurrDir/out.loc.om.scf
cp $Project.ScfOrbv $Project.ScfOrb
```

Fichiers de sortie = OAO, NONORLOC
Cree les fichiers de données
DOLO_MOLCOSTIN DOLO_MOTRAIN

2) Projeter les orbitales locales sur les OM SCF

Fichier en entree = OAO, NONORLOC, \$Project.ScfOrb (ou \$Project.RasOrb)

```
proj_scf < $CurrDir/input.loc.om.scf >> $CurrDir/out.loc.om.scf
cp NONORLOC_scf NONORLOC
```

Fichier de sortie
= NONORLOC_scf

3) Orthogonaliser le jeu d'OM locales

Fichier en entree = OAO, NONORLOC

```
schmudort < $CurrDir/input.loc.om.scf >> $CurrDir/out.loc.om.scf
cp LOCORB $CurrDir/LOCORB-OM-scf
```

Fichier de
sortie = LOCORB

4) visualiser le jeu d'OM locales

```
ORB2q5 < $CurrDir/in.dolo.orb2q5 > $CurrDir/out.dolo.orb2q5
q52molden < $CurrDir/in > $CurrDir/orb.dolo.molden
```

```
cp $CurrDir/LOCORB-OM-scf INPORB
```

Optimisation des orbitales : ICS -> matrice densité -> nouvelles OM

Itérer les OM

Les OM obtenues sont locales et de qualité CASSCF

```
noscf $CurrDir/input.noscf-fond > $CurrDir/out.noscf.fond
cp LOCORB2 $CurrDir/LOCORB2-noscf-fond
```

Fichier de sortie =
LOCORB2

Transformation des integrales atomiques -> moleculaires

```
cp $CurrDir/LOCORB2-noscf-fond INPORB
neworb
cp OUTORB INPORB
```

Fichier de sortie = OUTORB

```
molcas DOLO_MOTRAIN > $CurrDir/out.ic_loc
```

Motra

```
cp $CurrDir/LOCORB2-noscf-fond INPORB
molcost < $CurrDir/input.casdiloc >> $CurrDir/out.ic_loc
ijkloc < $CurrDir/input.casdiloc >> $CurrDir/out.ic_loc
```

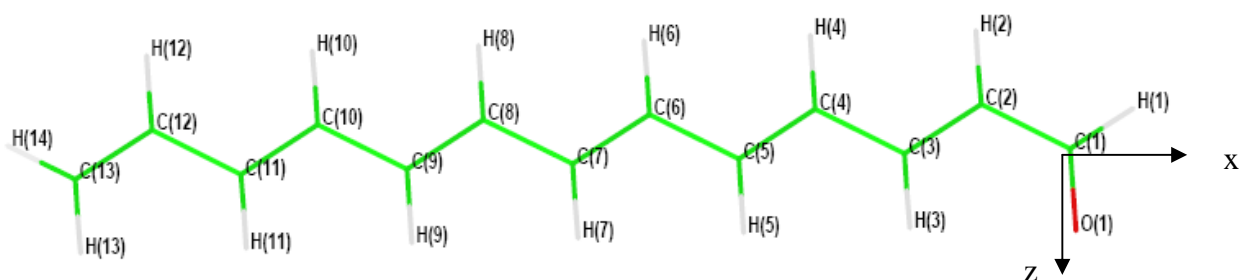
Voir manuels

IC localisee

```
casdiloc < $CurrDir/input.casdiloc > $CurrDir/out.ic_loc-sl=0.003
```

VI. Applications

A. Etude du tridequenal : Transition $n \rightarrow \pi^*$



1. Molcas

doublet n de l'oxygène = l'orbitale p_x
système π selon l'axe y.

```

&SEWARD &END
Title
  tridequenal
symmetry
  Y
Basis set
C.ANO-L...3s2p.
  C1  0.00000000  0.00000000  0.00000000  Angstrom
  C2 -1.25573684  0.00000000 -0.72500000  Angstrom
  C3 -2.42487113  0.00000000 -0.05000000  Angstrom
  C4 -3.68060797  0.00000000 -0.77500000  Angstrom
  C5 -4.84974226  0.00000000 -0.10000000  Angstrom
  C6 -6.10547910  0.00000000 -0.82500000  Angstrom
  C7 -7.27461339  0.00000000 -0.15000000  Angstrom
  C8 -8.53035023  0.00000000 -0.87500000  Angstrom
  C9 -9.69948452  0.00000000 -0.20000000  Angstrom
  C10 -10.95522136  0.00000000 -0.92500000  Angstrom
  C11 -12.12435565  0.00000000 -0.25000000  Angstrom
  C12 -13.38009249  0.00000000 -0.97500000  Angstrom
  C13 -14.54922678  0.00000000 -0.30000000  Angstrom
End of basis
Basis set
O.ANO-L...3s2p.
  O1  0.0000000  0.00000000  1.220000  Angstrom
End of basis
Basis set
H.ANO-L...2s.
  H1  0.95262794  0.00000000 -0.55000000  Angstrom
  H2 -1.25573684  0.00000000 -1.82500000  Angstrom
  H3 -2.42487113  0.00000000  1.05000000  Angstrom
  H4 -3.68060797  0.00000000 -1.87500000  Angstrom
  H5 -4.84974226  0.00000000  1.00000000  Angstrom
  H6 -6.10547910  0.00000000 -1.92500000  Angstrom
  H7 -7.27461339  0.00000000  0.95000000  Angstrom
  H8 -8.53035023  0.00000000 -1.97500000  Angstrom
  H9 -9.69948452  0.00000000  0.90000000  Angstrom
  H10 -10.95522136  0.00000000 -2.02500000  Angstrom
  H11 -12.12435565  0.00000000  0.85000000  Angstrom
  H12 -13.38009249  0.00000000 -2.07500000  Angstrom
  H13 -14.54922678  0.00000000  0.80000000  Angstrom
  H14 -15.50185473  0.00000000 -0.85000000  Angstrom
End of basis
End of input
  
```



```

&SCF &END
Title
  tridequenal
occupied
  43 7
end of input

```

2. Localisation

a. Dolo

Dolo a besoin du fichier DOLOIN et lit OLD_DATA si il existe. OLD_DATA peut être créé de façon interactive.

<pre> <u>DOLOIN</u> &smufil prefix='tridequenal.' / &smu nprint=0, orb='tridequenal.ScfOrb' / &oao / o1 1s(1) pr=1 o1 1s(2) pr=2 o1 2p(1) pr=2 c* 1s(1) pr=1 c* 1s(2) pr=2 c* 2p(1) pr=2 h* 1s(1) pr=2 fin &orb symm='Y' / </pre>	<pre> 4 namelists : &smufil (préfixe) &smu orbitales utilisées &oao orbitales atomiques de cœur et de valence (les virtuelles ne doivent être données que si l'on veut s'en servir). Pr= définit des priorités, pr=1 pour les OA les plus profondes puis on augmente. Ici pr=2 pour la valence. &orb donne la symétrie TELLE qu'elle est notée dans seward </pre>
<pre> <u>OLD_DATA</u> core_ c* 1s(1) core_ o1 1s(1) sigma_ c* 1s(2) 2p(1) : h* 1s(1) dmax=1.2 ANGSTROM BOND_ CHAIN='O1(LP2)=C1-C2=C3-C4=C5-C6=C7- C8=C9-C10=C11-C12=C13',BAS=14*'1S(2) 2P(1)' > O1 Z Z Z A Z Z A </pre>	<pre> ORBITALES DE CŒUR LIAISONS C-H CHAINE DE LIAISONS SIMPLES (-) ET DOUBLES (=) AINSI QUE LES DOUBLETS DE L'OXYGENE (lp2) ON DEFINIT LES ACTIVES A LA MAIN (EN ARRIVANT D'UN SCF) = grep O1 : les OM contenant O1 sont Actives(A) ou inchangees (Z) </pre>

Dans l'output, quelques contrôles (géométrie, nombre d'électrons...) sont faits : le programme s'arrête s'il y a un problème.

Ensuite, les orbitales locales définies sont listées :

Exemple : les liaisons définies par

sigma_c* 1s(2) 2p(1) : h* 1s(1) dmax=1.2 ANGSTROM

Liaisons σ_{CH} et σ^*_{CH}

Generated Basic Data

```
O_C1H1.SG C1 1S(2) 2P(1) : H1 1S(1) (1,1)      d= 1.100000
O_C2H2.SG C2 1S(2) 2P(1) : H2 1S(1) (1,1)      d= 1.100000
O_C3H3.SG C3 1S(2) 2P(1) : H3 1S(1) (1,1)      d= 1.100000
Etc...
```

BOND_CHAIN='O1(LP2)=C1-C2=C3-C4=C5-C6=C7-C8=C9-C10=C11-C12=C13',BAS=14*'1S(2) 2P(1)'

bond_generated data:

```
O_C1O1.SP C1 1S(2) 2PX(1) 2PY(1) 2PZ(1) : O1 1S(2) 2PX(1) 2PY(1) 2PZ(1) (4 2)
O_C1C2.SP C1 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C2 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C2C3.SP C2 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C3 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_C3C4.SP C3 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C4 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C4C5.SP C4 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C5 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_C5C6.SP C5 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C6 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C6C7.SP C6 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C7 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_C7C8.SP C7 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C8 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C8C9.SP C8 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C9 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_C10C9.SP C10 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C9 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C10C11.SP C10 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C11 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_C11C12.SP C11 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C12 1S(2) 2PX(1) 2PY(1) 2PZ(1) (1 1)
O_C12C13.SP C12 1S(2) 2PX(1) 2PY(1) 2PZ(1) : C13 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 2)
O_O1.S.LP O1 1S(2) 2PX(1) 2PY(1) 2PZ(1) (2 0)
```

Le doublet n et l'orbitale π sont déclarées actives (a posteriori car les orbitales lues sont des orbitales SCF). Ce deux orbitales font intervenir l'atome O1. La commande «> O1 » fait un grep sur les orbitales locales et sept d'entre elles contiennent cet atome :

```
> O1 Z Z Z A Z Z A      (z = pas de changement, a = active)
      7 orbitals changed
```

```
-----
C_COR.O1.2 O_C1O1.SG O_C1O1.PI O_C1O1.PI O_C1O1.SG O_O1.S.LP01 O_O1.S.LP02
Z Z Z A Z Z A
LO 45 O_C1O1.PI was ACTIVATED
LO 84 O_O1.S.LP02 was ACTIVATED
```

Les données sont contrôlées à la fin dans un tableau. Il faut avoir le même nombre d'occupées locales que SCF :

SYM		1	2	
	Occ	43	7	
SCF	Act	0	0	Rappel les orbitales SCF délocalisées
	Virt	83	21	
	Occ	42	7	
LO's	Act	1	1	Ce qu'on a définit comme orbitales locales
	Virt	27	6	
	Occ	1	0	
Diff	Act	-1	-1	
Différence entre orbitales SCF et locales				
	Virt	56	15	
Warning: there are non-zero values in occ and/or				
=====				
AO's		70	14	Egalité entre AO's et LO's : les
LO's		70	14	orbitales locales sont générées
(numbers of LO's and AO's must (output et fichier NONORLOC)				
be equal for each symmetry)				

=====

Le programme termine en donnant les données de motra et de molcost dans l'output et crée également 2 fichiers contenant ces infos: DOLO_MOTRAIN et DOLO_MOLCOSTIN.

Ces orbitales sont locales mais ne sont pas orthogonales. Elles sont écrites en clair dans le fichier NONORLOC.

On commence par les projeter sur les orbitales SCF (proj_scf), puis par les orthogonaliser (schmudorb)

b. Proj_scf

Peu de données :

```
&pscf prefix='tridequenal.' /
```

L'output signale s'il y a des recouvrements inférieur à 0.8 et écrit les actives. Les orbitales projetées sont dans le fichier NONORLOC_SCF.

c. schmudort

Très peu de données

```
&ort /
```

Lit le fichier NONORLOC (donc copier le fichier NONORLOC_SCF dans NONORLOC) et écrit les orbitales dans l'output et dans le fichier LOCORB. Ces orbitales sont locales et orthogonales.

3. Optimisation des orbitales locales

Les orbitales obtenues sont de qualité SCF. On peut les optimiser pour décrire le ou les états qui nous intéressent. Ceci se fait par une interaction de configurations limité aux monoexcités (ICS) puis en diagonalisant la matrice densité (de l'état ou la moyenne des matrices densités des différents états) un nouveau jeu d'OM est obtenu. A partir de ce nouveau jeu d'OM, on recommence l'IC des monos etc... jusqu'à convergence.

Cette optimisation est faite par le programme noscf, qui gère l'appel à la transformation des intégrales atomiques en moléculaires(motra / molcost), l'IC (casdi) et la diagonalisation de la matrice densité (locnats)

```
&nofil prefix='tridequenal.' /  
&no data='AUTO',r=' > ',  
nmat=1,netat=1,metat=1,coef=1*1.d0 / (on n'a qu'un seul état)  
&casdi syspin='+',gener='CAS+S'  
nelac=2,iprec=7,
```

Les orbitales obtenues sont écrites dans l'output et dans le fichier LOCORB2. LOCORB2 est également copié dans le CurrDir.

4. Transformation des intégrales atomiques en intégrales moléculaires

Il faut encore executer motra de molcas puis transformer ces intégrales moléculaires dans le format du code d'IC puis –si on fait une IC localisée (casdiloc) executer ijkloc.

Les données de motra et molcost ont été écrites par dolo :

DOLO_MOTRAIN :

```
&MOTRA &END
Frozen
0 0
Delete
0 0
END of Input
```

DOLO_MOLCOSTIN

```
&COST
Prefix='tridequenal.'
Frozen=0,0
FERMI=43, 7
/
```

Données de ijkloc

```
&ijkloc
```

```
prefix='tridequenal.',symm='Y' / Attention à préciser la symétrie telle que
donnée dans le fichier de données de seward.
```

Vérifier dans l'output l' orthogonalite des orbitales :

```
« défaut d'orthogonalite max 7.961681529035586E-015
défaut de normalisation max 2.930988785010413E-014
dsymmétrisation s'est bien passée »
```

5. Interaction de configurations localisée (casdiloc ou exsci)

Données

```
&locci prefix='tridequenal.',
NHAB=1,mhab=3,iterh=10,yhmono=t
```

données relatives à la correction de l'erreur
de size-consistence des IC

```
gener='CAS+DDCI',liens='KIJ',syspin='+', type d'IC, type de sélection (KIJ=  
intégrales d'échange)
```

```
mprint(4)=0,nessai=3,mprint(5)=0,
```

```
s1=0.003,ms2=0,prec=1.d-6,nvec=2 /
```

seuil sur les intégrales d'échange

Manuels d'utilisation

Logiciels de chimie quantique toulousains	2
I. Présentation générale	2
II. SCF, CASSCF, interactions de configurations et méthodes perturbatives.....	2
III. Utilitaires	4
IV. Exemples de performances	4
V. Exemple de script	5
VI. Applications	7
A. Etude du tridequenal : Transition $n \rightarrow \pi^*$	7
Manuels d'utilisation.....	12
I. Molcost	13
A. What does the program?	13
B. Data	13
C. Aspect of the Info file (example of ethylene molecule) ("prefix"Info).....	13
D. How to read the unformatted Mono file ("prefix"Mono)	14
II. DOLO (DO Local Orbitals) : programme de localisation à priori	15
A. Préliminaires	15
B. Data	15
C. How it works	16
D. Possible Data.....	17
E. Details of Data.....	22
III. Proj_scf et Schudort	22
IV. Noscf	24
V. Casdi	25
A. Fichiers d'entree et de sortie ou Comment se raccrocher au reste du monde	27
B. Fichiers de donnees	28
C. Vecteurs d'essai	33
D. Détails techniques	34
E. Calculs sc2.....	35
VI. Ijkloc	36
VII. Casdiloc	37
VIII. Naturalt et locnats.....	40
IX. Internat	41
X. Orb2q5.....	43
XI. Q52molden.....	43

Les modes d'emplois présentés ici ne couvrent pas l'ensemble des manuels mis à la disposition dans le répertoire d'installation du code.

I. Molcost

Interface between molcas and the toulouse programs

A. What does the program?

1. After seaward MAINTENANT, ON PEUT UTILISER MOLCAS2Q5 et Q52CASDI
 - generates an formatted Info file containing various information about the system studied
 - generates an unformatted Mono file containing one-electron operators
2. After motra
 - generates the same files, with Molecular Orbital information if files TraOne and TraInt exist.

B. Data

a namelist

```
&cost
prefix='Project.'          (required data)
molcas=  version of molcas: 4 or 5 or 54 (default 0: in that case, molcost
          analyzes the OneInt file to get the molcas value)
ycl=      T: generates a formatted file of MO integrals (default F)
yao=      T: reads and interfaces the AO integrals (default T)
ymo=      T: reads and interfaces the OM integrals (default T)
ybas=     T: reads and interfaces the COM file      (default F)
ysym=     T: the one-electron operators in the Mono file are
          square matrices
          F: the one-electron operators in the Mono file are
          triangular matrices                      (default F)
          (see in paragraph IV the form of file Mono)
fermi=    0: fermi level (number of occupied orbitals in each symmetry)
          (required if molcost runs after motra)
```

C. Aspect of the Info file (example of ethylene molecule) ("prefix"Info)

```
=====
File: C2H4.Info
created by molcost
date:
=====
&cost_AO
```

```

nsym=8
isym= 6, 6, 3, 3, 1, 1, 0, 0,
norb= 20
its=
  1,2,3,4,5,6,7,8,
  2,1,4,3,6,5,8,7,
  3,4,1,2,7,8,5,6,
  4,3,2,1,8,7,6,5,
  5,6,7,8,1,2,3,4,
  6,5,8,7,2,1,4,3,
  7,8,5,6,3,4,1,2,
  8,7,6,5,4,3,2,1,
enuc= 33.382397244601
natom= 2
coor=
  1.2651715494, 0.0000000000, 0.0000000000,
  2.2159663935, 1.8187044168, 0.0000000000,
atom=
'C ', 'H ',
label=
'C 1s ', 'C 1s ', 'C 1s ', 'C 2px ', 'H 1s ',
'H 1s ', 'C 1s ', 'C 1s ', 'C 1s ', 'C 2px ',
'H 1s ', 'H 1s ', 'C 2py ', 'H 1s ', 'H 1s ',
'C 2py ', 'H 1s ', 'H 1s ', 'C 2pz ', 'C 2pz ',
/

```

if Info is created after motra, a second namelist appears:

```

&cost_MO
nsym=8,norb= 20,noc= 8
itsym=
1,1,1,2,2,3,4,5,1,1,1,2,2,2,2,3,3,4,4,6,
isym= 6, 6, 3, 3, 1, 1, 0, 0,
frozen= 0, 0, 0, 0, 0, 0, 0, 0, 0,
enuc= 0.333823972446E+02
ycl=F
&end

```

D. How to read the unformatted Mono file ("prefix"Mono)

```
open(1,file=...)
```

1. ysym=F (one-eletron operators in triangular form)

a. seek the label:

```

- One-electrons integrals: label= '==ONEINT MATRIX (TOTAL)'
- Overlap matrix:          label= '==OVERLAP MATRIX (SYM)'
```

```
character*80 aa
```

```
do
```

```

  read(1) aa
  if(trim(aa).eq.label) then
```

b.

```

  read the operator
  - read the length of the operator, the number of symmetries,
    the number of orbitals by symmetry
  read(1) len,nsym,(isym(k),k=1,nsym)
  - read the operator
  read(1) hmono(1:len)
```

```
endif
enddo
```

other one electron operators must be added...

II. DOLO (DO Local Orbitals) : programme de localisation à priori

Three programs : a) dolo : batch version
b) idolo : interactive version
c) dolo3d.py : for interactive AND graphical version

Files to be read : - molcost files
- an orbital file (RasOrb, ScfOrb...)

output files : - NONORLOC
(Local Orbitals: they are not orthogonal, and are written in the basis of the OAO (Orthogonalized Atomic Orbitals))

A. Preliminaries

design the molecule, with atom names

B. Data

1. A DOLOIN file (see "DOLOIN" part)

```
example of DOLOIN file:
&smufil prefix='test.' /
&smu nprint=0, orb='test.RasOrb' /
&oao /
Fe 1s(1) pr=1
Fe 2p(1) pr=1
Fe 3d(1) pr=2
C* 1s(1) pr=2
N* 1s(1) pr=2
H* 1s(1) pr=2
S 1s(1) pr=2
C* 2p(1) pr=2
N* 2p(1) pr=2
H* 2p(1) pr=2
S* 2p(1) pr=2
fin
&orb symm='XY' /
```

2. The data to generate local Orbitals (LO) can be given

- interactively
- or can be prepared in the file 'OLD_DATA'

C. How it works

a. Interactive run

The program uses the 2 molcost files (Info, Mono) and an orbital file (ScfOrb, RasOrb, or other). These files are in workdir, as given in namelist &smu

If there is no access to workdir:

(for example if workdir is on another node), a solution is to:

- give workdir= current directory
- in the script, after running molcost, add:


```
cp ...Info $CurrDir
cp ...Mono $CurrDir
cp ...Orb $CurrDir
```

- enter: dolo

The program starts, and ask for a data to generate one or several LO's (see below list of commands)

- In the same time, the program generates a new file OLD_DATA_1, or OLD_DATA_2, or OLD_DATA_n+1, if OLD_DATA_n is the last existing one
In this file, it writes the data you just entered

If the program crashes, or if you stop it, data already entered are in OLD_DATA_x. Copy OLD_DATA_x to OLD_DATA, verify if everything is OK, and re-enter "dolo"

b. background run

Once dolo runs correctly, copy the last OLD_DATA_n into OLD_DATA

The script is something like:

- molcost < INPUT
- cp \$CurrDir/DOLOIN .
- cp \$CurrDir/OLD_DATA .
- dolo < INPUT_DOLO (the INPUT_DOLO file is one line long (one char): X)
- (proj_scf < INPUT ; cp NONORLOC_scf NONORLOC)
- schmudort < INPUT

D. Possible Data

Many possibilities exist to enter a data line which will generate one or several LO's.

a. Basic generation of one or several LO

The "basic data" correspond to those read by the program
it is easier to use "elaborated data" (see below)
The program transforms the "elaborated data" into "basic data"

example:

```
O_C2H1 C2 1S(2) 2P(1) : H1 1S(1) (1,1)
(not case sensitive)
```

explanation:

O_C2H1 is the label of the LO('s).
- The first letter (here: O), can be
A: active. C: core. G: frozen. O: other
- _ character number 2: compulsory
- characters 3 to 12. Name of the LO, chosen by the user

```
C2 1S(2) 2P(1) , H1 1S(1)
atoms and AO's that describe the LO. Here, the LO is described by
atoms C2 and H1. The AO's for C2 are 1S(2) (i.e. the second S in the
ANO basis set, corresponding to the valence shell), and for H1 it is
1S(1)
```

the atoms are separated by ":"
There is in theory no limitation in the number of atoms/AO.

(1,1): first number: 1 bonding LO
second number: 1 anti-bonding LO
for a double bond, we would have (2,2)
for a core orbital, there would be only one atom, and (1,0)
Warning! For a SINGLE bond, appearing TWICE for symmetry reasons
we would have (2,2)!

Elaborated data

The "elaborated data" are easier to use. In the code, they generate basic data

There are two main "elaborated data". Their names are "bond_" and "deloc_"

Almost everything can be done with these two data

There are other kinds of data (core_, sigma_ etc...)
They may produce errors, if there is symmetry. Be careful, and verify that you really obtained what you wanted.

b. An elaborated data: BOND_

a subset of the Lewis structure is given by:

```
chain='AT1-AT2-AT3...'
```

where:

ATi are labels of atoms

"-" can be -, =, ~ for single, double and triple bonds

An information may be added to ATi:

ATi(LPn) or ATi(SYn) (lone pairs, bonds invariant under a symmetry transformation) (see below the examples)

```
.....  
Give basis set for atoms  
.....
```

```
bond_ basref='...'
```

These is a preliminary data, which defines the default basis set for all following bond_ data. The default may be changed by another "bas" data. It can be ignored for a particular bond_ data, by giving "bas=" (as in many examples below)

```
example: BOND_ BASREF='O:1S(2) 2P(1),C:1S(2) 2P(1),H:1S(1)'
```

```
.....  
example 1: (single bond)  
.....
```

the example of (a) can be written:

```
bond_ chain='C2-H1',bas='1S(2) 2P(1)', 'H1 1S(1)'
```

The label of the LO is chosen by the program (O_C2H1.SG)

n=2: there are 2 atoms (default value: 2)

```
.....  
example 2: (double and triple bond)  
.....
```

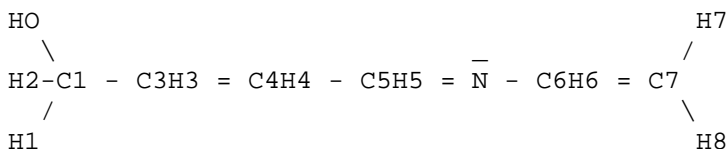
a) bond_ n=2,chain='C2=C1',bas=2*'1S(2) 2P(1)'

generated labels: O_C2H1.SG O_C2H1.PI

b) bond_ n=2,chain='C2~C1',bas=2*'1S(2) 2P(1)'

generated labels: O_C2H1.SG O_C2H1.P1 O_C2H1.P2

```
.....  
example 3: (bonds and lone pairs)  
.....
```



```
chain C1-C3=C4-C5=N-C6=C7:
```

```
bond_ n=7,chain='C1-C3=C4-C5-N(lp1)=C6-C7',bas=7*'1S(2) 2P(1)',
```

```
N(lp1) : atom N has 1 Lone Pair (no limit to n in lpn)
```

```
.....  
example 4: (3 lone pairs)  
.....
```

```
bond_ chain='S1(lp3)-C13',bas=2*'1s(1) 2p(1)'
```

We have: $\begin{array}{c} / \\ | \text{ S1-C13} \\ \backslash \end{array}$

The program generate the following basic data::

```
o_slc13 s1 1s(1) 2p(1) : c13 1s(1) 2p(1) (4 1)
o_sllp s1 1s(1) 2p(1) (3 0)
```

The lone pairs were generated twice (once by each basic data).

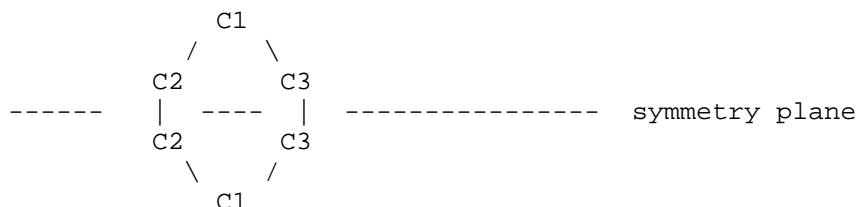
In o_slc13, they are "spread" over the 4 occupied

To correct that, the program generates:

```
subs o_slc13 o_sllp
```

which subtracts the lone pairs from the set (LP+sig+sig*). The sigma appears then to be pure, the LP are unchanged these generated data appear in OLD_DATA_x

```
.....
example 5: (bonds invariant by symmetry)
.....
```



```
bond_ n=2,chain='C1-C3(sy1)'
```

sy1 means that There is a C3-C3 bond, invariant by symmetry. The program will generate the following basic data:

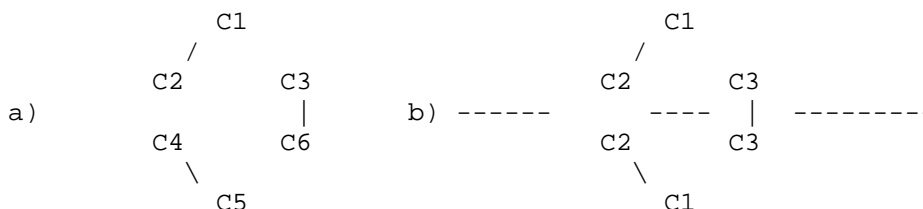
```
o_C1C3 C1 1s(2) 2p(1) : C3 1s(2) 2p(1) (3 3) (3 occ, 3 anti bonding
corresponding to C1-C3-C3-
```

C1)

```
O_C3 C3 1s(2) 2p(1) (1 1) (C3-C3 bond)
subs o_C1C3 c_C3 (subtract c_C3 from o_C1C3)
```

After subtraction, The orbitals with the label o_C1C3 are really on the C1-C3 bond, and not on C3-C3

```
.....
example 6: (pi bonds)
.....
```



a) bond_ n=2,chain='C1=C2',orth='c1,c2,c3'

b) bond_ n=1,chain='C3(sy2)',orth='c1,c3,c3' (bond invariant by symmetry)

only double bonds are represented (Kékulé).

The 3 centers c1,c2,c3 define a plane.

The pi orbitals are orthogonal to this plane.

(orth='c1,c2,c3' means: no component in plane c1,c2,c3)

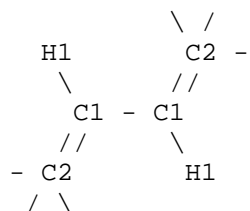
.....
example 7: (All p, or d orbitals of an atom. Here: closed shell Fe)
.....

3d of an iron atom:
a) 3 filled LO's
bond_ n=1,chain='fe(lp3)',bas='3d(1)'
b) 2 empty LO's
bond_ n=1,chain='fe',bas='3d(1)',vi=2

.....
example 8 (several subgraphs in one "bond_" data)
.....

This possibility is sometimes mandatory, if the data (SYn) is used

example of butadiene:



1. C2-C1-C1-C2 bond: bond_ chain='C1(sy1)=c2'
2. C1-H1 bond: bond_ chain='C1(sy1)-H1'

an atom with (SY.) CANNOT appear in two different bond_ lines
use instead:
bond_ chain='C1(sy1)=c2;C1(sy1)-H1'

NOTE: in general, all bonds can be generated with the BOND_ command

c. An elaborated data: DELOC_

build a set of orbitals that are delocized on several atoms

example:
deloc_ n=3,chain='c1-c6-c11',bas=3*'2pz(1)',noc=3,nvirt=3,label='a'
means:
n : number of atoms (can be more by symmetry)
c1-c6-c11 : list of the atoms, separated by "-"
noc, nvirt : number of occupied and anti-bondings (in this example,
we have a 6-atom ring)
label='a' : these orbitals will be active in the MRCI
bas=3*'2pz(1)' : in this example, the Pz pi orbitals were seeked

d. SIGMA_

A set of sigma bonds
sigma_ c* 1S(1) 2P(1) : h* 1S(1) dmax=1.2 ANGSTROM
generates all CH bonds for all (C,H) couples, separated by less
than 1.2 Angstrom.

Note. in C*..., * replaces only NUMBERS, and not letters. So Cu is not considered as a Carbon atom. Do not call the C atoms Ca, Cb...

NOTE: This data makes sometimes problems if there is symmetry. In this case, better use BOND_

e. CORE_ , DIFF_

A set of core or diffuse orbitals
CORE_ C* 1S(1) ---> occupation is (1 0)
DIFF_ C* 1S(1) ---> occupation is (0 1)

f. PI_

similar to SIGMA_, for pi orbitals
example:
PI_ C* 2P(1) : C* 2P(1) DMAX=1.4 ANGSTROM

NOTE: This data makes somtiomes problems if there is symmetry. In this case, use BOND_c1,c2,c3

g. SUBS

subtraction (or projection) of a set of LO from another set of LO,
as it appears in B. example 4

subs o_slc13 o_sllp (the space of o_slc13 is larger than the space
of o_sllp)

h. RENAME

If one enters for example
o_sllp s1 1s(1) 2p(1) (3 0) (basic data)
/ | S1
 \

3 lone pairs are generated, with the same name. The names need to
be different in the following of the calculation

a solution is to use BOND_ instead
bond _ n=1,chain='S1(lp3)',bas='1s(2) 2p(1)'

or to rename the bonds
REN o_sllp o_sllp1 o_sllp2 o_sllp3
(REN common_name individual_names)

i. Multiple data

- using "[", "]"
ex.

```

BOND_ CHAIN='W1',BAS='3D[0 1+ 2+ 2-](1) 3D[0 1+ 2+ 2-](2)',LP=1,VI=1
generates 4 bond data:
BOND_ CHAIN='W1',BAS='3D0(1) 3D0(2)',LP=1,VI=1
BOND_ CHAIN='W1',BAS='3D1+(1) 3D1+(2)',LP=1,VI=1
...
- using "[["
  ex:
  BOND_ CHAIN='O[1 2 3 4 5 6 7 11 13 15](LP1)',BAS='2P[[X Y Z]](1) 2P[[X Y
Z]](2)']
  generates all combinations of the 10 data due to [1 2 3 4 5 6 7 11 13 15]
and the 3 due to [[X Y Z]]
  (a total of 30 bond_data)

```

E. Details of Data

DOLOIN file:

All data are compulsory

```

&smufil prefix='Project.' /
&smu
nprint=0,
orb='test.RasOrb' /                               orbital file name, as a result
                                                    or a SCF or CASSCF calculation
/

&oao /
Concerns data to build OAO (Orthogonalized Atomic Orbitals)
list of Atomic Orbitals, with priority in the orthogonalization
example:
Fe 1s(1) pr=1
--> first s orbital of atom Fe in the list of AO's
    the AO is orthogonalized with highest priority (pr=1)
Fe 2p(1) pr=1
--> idem for all 2px, 2py and 2pz AO's
Fe 3d(1) pr=2
--> idem for the 5 d orbitals. (pr=2: will be orthogonalized, for example,
    to the 1s of C which has (pr=1) (projection orthogonalization)
-- etc...
C* 1s(1) pr=2
N* 1s(1) pr=2
H* 1s(1) pr=2
S 1s(1) pr=2
C* 2p(1) pr=2
N* 2p(1) pr=2
H* 2p(1) pr=2
S* 2p(1) pr=2
fin
&orb symm='XY' /                               end of AO data
                                                    symm: as given in seward after the line
                                                    SYMMetry

```

III. Proj_scf et Schudort

Proj_scf : Programme de projection d'un jeu d'orbitales dans l'espace d'un autre jeu d'orbitales

Schmudort : Programme d'orthogonalisation

Exemple : on a un jeu d'orbitales locales (occ, virt)
et un jeu d'orbitales SCF (occ, virt)
on projette les occupées locales dans les scf
on projette les virt locales dans les scf
on reorthogonalise
ainsi, on a un jeu d'orbitales locales de qualité SCF

ENCHAINEMENT DES PROGRAMMES

```
dolo  
proj_scf  
cp NONORLOC_scf NONORLOC  
schmudort
```

dolo crée un jeu d'orbitales locales non orthogonales (ds le fich NONORLOC)

proj_scf projette sur les orb scf (les met dans NONORLOC_scf,
écrit comme NONORLOC)

cp NONORLOC_scf NONORLOC : on écrase le fichier primitif
les fichiers NONORLOC et NONORLOC_scf sont exprimés dans une base d'OA
orthogonalisées (OAO)
le fichier contenant les OAO s'appelle "OAO"
les 3 fichiers OAO, NONORLOC et NONORLOC_scf sont donnés en clair sous
la forme des fichiers d'orbitales de Molcas, avec des labels

schmudort on orthogonalise

EN ENTREE

fichiers OAO et NONORLOC

un fichier d'orbitales scf ou casscf en orbitales delocalisees
(ScfOrb ou RasOrb), format en clair, orbitales Molcas.

EN SORTIE

fichier NONORLOC_scf, de meme type que NONORLOC

DONNEES

```
&pscf  
prefix=  
calcul='SCF'      CASSCF : il y a des orbitales actives  
fermi=0          donnée facultative si calcul='SCF'  
                  obligatoire si calcul='CASSCF'  
act='A_'         alors les actives sont reconnues par le fait que  
                  leur label commence apr A_  
                  aucune autre donnee n'est testee pour le moment !  
scf=' '          fichier contenant les orbitales de reference  
                  (qui definissent les espaces sur lesquels on projette)  
                  ' ' : alors le nom est prefixScfOrb  
                  si calcul='CASSCF', changer la valeur de scf!
```

/

IV. Noscf

Obtention d'orbitales optimisées par itérations sur la matrice densité.
génère les données et appelle les programmes :

- motra
- molcost ou molcsd
- casdet
- casdi
- locnats

puis itère jusqu'à convergence

Utilisation :

```
cp "Orbitales" INPORB (avec schmud : cp LOCORB INPORB)
export CurrDir WorkDir Project
export PATH=$PATH:"lieu_de_cost"/bin
noscf donnees (ET NON i"noscf < donnees !!!)
orbitales de sortie : LOCORB2
```

Données :

```
&nofil PREFIX=
/
&no
data=' ' 'AUTO' lit quelques données d'un fichier de schmudorb
les données de schmudorb pilotent ainsi une grande partie
de
la suite du calcul (defaut ' ')
FERMI=0 par sym, nb d'orb occupées
(lu de schmudorb si data='AUTO')
MOLCAS=5 (ou 4)
frozen=0 donné par sym
(lu de schmudorb si data='AUTO')
delete=0 donné par sym
(lu de schmudorb si data='AUTO')
maxit=50 nb max de cycles motra...locnats
r=' > ' (les fichiers de sortie sont dans la
directory de travail (sinon r=' # '))
nmat=1 nb de fichiers de mat densité
netat= nb d'états par fichier (nmat données)
metat= numéros des états demandés (Somme(netat)) données
coef=1.d0 coefs des états (Somme(netat)) données
stop=' ' ('MOLCOST' pour arrêt après molcsd, donne la numérotation
des orbitales)
seuil=1.d-4 seuil de convergence du processus (sur les elements des
blocs extérieurs de la matrice densité)
/
&casdi (nmat fois)
syspin='0',
nessai=0,
iprec=6
noac=0, (lu de schmudorb si data='AUTO')
numac= (lu de schmudorb si data='AUTO')
nelac=0, (Attention, non lu ! A donner impérativement)
gener='CAS+S',
```

```
is0=1
ms2=0
/
```

les variables en majuscules sont celles obligatoires
les valeurs données sont celles par défaut

Données supplémentaires:

noscf génère, à partir de &nofil, &no et &casdi les données pour motra, molcost, casdet, casdi et locnats.

Il peut arriver qu'on veuille donner à un de ces programmes une donnée supplémentaire. Par exemple, si on veut donner la donnée twvec a &dav pour le 2ème calcul casdi (alors nmat >=2).

On ajoute a la fin du fichier :

```
DAV2
 twvec=0.01
FIN
```

explication : DAV pour &dav, 2 pour 2eme casdi

la (les) donnee(s)

FIN (de l'ajout a &dav)

la ligne est ajoutée avant la fin de namelist "/"

Rem : si on veut ajouter un vecteur d'essai en donnee

```
DAV1
 nessai=1
/
 &ess file='ESSOUT0'
FIN
```

le fichier de données devient après rajout des 3 lignes :

```
&dav
...
...
nessai=1 \
/ > 3 lignes supplémentaires
&ess file='ESSOUT0' /
/ (déjà présent, donc pas a rajouter)
```

V. Casdi

La chaîne casdet/casdi permet de :

- generer un espace de determinants. Cet espace est forme d'un espace modele (ou generateur) qui est un CAS ou un espace selectionne et d'un espace plus vaste genere a partir de l'espace modele (simples, simples et doubles, DDCI ...)
(Programme Casdet)

- diagonaliser cet espace.
faire une diagonalisation (SC)**2
(Programme casdi)

GENERALITES

Situation du programme :

```
-----
exterieur  --->  Fichiers          | chaine |          vecteurs propres
                  PREFIX.ijnam  ---> | casdi  |          matrice densite
                  PREFIX.ijcl   |       |          ...
-----
```

- Fichiers de donnees

Ce sont en general des namelists. Elles sont lues par un sous-programme propre, donc toujours suivant les memes criteres, quelle que soit la machine utilisee.

En particulier :

- . le caractere de fin est soit "&end" soit "/"
- . les variables sont separees par des virgules
- . La virgule n'est pas necessaire en fin de ligne
- . On peut donner : i=3,6,7 si i est un tableau ou bien i(2)=6,i(3)=7, MAIS PAS i(2)=6,7

Namelist &...fil

C'est la premiere donnee de tous les programmes
Elle permet de gerer les noms des fichiers.
Pour ces namelists, une donnee obligatoire est
prefix='PREFIX.'

La plupart des fichiers s'appelleront alors
PREFIX.suffixe

ou "suffixe" est donne par le programme.

par exemple, si on donne

```
prefix='CH2'
```

le fichier de determinants s'appellera

```
CH2.det
```

e les autres fichiers s'appelleront

```
CH2.xxx
```

Si on veut changer le suffixe, on entre la donnee :

```
det='NOUVEAU_DET'
```

le fichier cree sera alors

```
CH2.NOUVEAU_DET
```

Si on veut changer tout le nom, on entre :

```
det1='DETER'
```

le fichier cree sera alors : DETER

dans tous les exemples donnes dans ce mode d'emploi
on supposera que

```
prefix='PREFIX.'
```

A. Fichiers d'entree et de sortie ou Comment se raccrocher au reste du monde

1. Fichiers d'entrée

Il y a deux fichiers d'entree obligatoires :

- IJCL contient les integrales mono et bielectroniques, et l'energie de reference SCF
- IJNAM contient de l'information sur le probleme etudie

a) Fichier *ijnam* : c'est une *namelist*

```
&ij
  nsym=   nombre de symetries du probleme
  norb=   nombre d'orbitales
  noc=    nombre d'occupees (1 ou 2 electrons)
  itsym=  symetrie des orbitales
  its=    matrice (8,8) donnant la table de
          multiplication des operations de symetrie
  enuc=   repulsion nucleaire
  ycl=    T : ijcl est en clair
          F : ijcl est en binaire (=defaut)
  ydp=    T : dans ijcl, les integrales sont en
          double precision
          F : dans ijcl, les integrales sont en
          simple precision
  ydp n'a un sens que si ycl=F. sa valeur n'est pas
  donnee librement dans ijcl, mais depend de la precision
  des integrales dans le fichier provenant de l'etape
  precedente.
```

* exemple de fichier *ijnam* :

```
&ij
  nsym=4,norb= 84,noc= 4,
  ydp= t
  itsym=
  1,1,3,4,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,
  2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
  3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
  4,4,4,4,4,
  its=
  1,2,3,4,0,0,0,0,0,
  2,1,4,3,0,0,0,0,0,
  3,4,1,2,0,0,0,0,0,
  4,3,2,1,0,0,0,0,0,
  enuc= .519330640508E+01
  ycl=F
  $END
```

Le fichier PREFIX.ijnam est malcommode a generer. Il est preferable de prevoir sa fabrication dans une "interface" semblable a ijcl.

b) Fichier *ijcl* : il peut etre en clair ou en binaire

- si ycl=T dans PREFIX.ijnam :
PREFIX.ijcl est un fichier en clair
dans ce cas il y a une integrale par ligne avec ses indices, en format libre, les integrales arrivant dans un ordre quelconque

```
i j k l h(i,j,k,l)      pour l'integrale (i,j,k,l)
...
...
```

puis, une par ligne, les integrales monoelectroniques
sous la forme :

```
i,j 0 0 hmono(i,j)
```

cette presentation 'en clair' n'est pas la plus efficace,
mais peut etre utile pour des tests ou pour transporter des
fichiers d'integrales

- si ycl=F dans PREFIX.ijnam :
PRE_ijcl est en binaire (par blocs) suivant le meme principe
(voir le code en appendice pour le mode de lecture)

c) Fichiers facultatifs :

- Vecteurs d'essai (voir section " Vecteurs d'essai")
- Habillage (SC)**2 (voir section correspondante)

2. Fichiers de sortie

- fichier davec : valeurs propres et vecteurs propres
- Habillage (SC)**2 (voir section correspondante)
- fichier dens : matrice densite

3. Se raccrocher a MOLCAS

- molcsd

on utilise le programme molcsd, qui fabrique les fichiers ijnam et ijcl
a partir de TraInt et TraOne

a) Compilation

Elle peut etre delicate, a cause du fichier en c cio.c, pour lequel les
options de compilation ne sont pas forcément evidentes. Puisque vous
avez sans doute molcas, une solution est d'aller trouver ces
options dans le makefile correspondant de molcas, ou bien de copier cio.o

b) Donnees voir III

B. Fichiers de donnees

1. Gestion des fichiers

```
Namelists &__fil / :
```

Les noms des fichiers sont donnes dans les namelists

- &cdfil pour casdet
- &cdifil pour casdi

Pour ces 2 namelists, une donnee obligatoire est

prefix='PREFIX.'

par exemple, si on veut que les fichiers s'appellent CH2.xxx

prefix='CH2.'

Le reste du nom de fichier (le suffixe) est automatiquement donne par le programme. Par exemple, casdet creera, dans l'exemple precedent,

le fichier de determinants

CH2.det

Si on veut changer le suffixe, on entre la donnee : det='NOUVEAU_DET'

le fichier cree sera alors

CH2.NOUVEAU_DET

Si on veut changer tout le nom, on entre : det1='DETER'

le fichier cree sera alors : DETER

2. PROGRAMME molcost

```
-----  
molcas ----> | PREFIX.TraInt          ----> molcost ----> | PREFIX.Info  
              | PREFIX.TrOne          ----> | PREFIX.Mono ----> casdi  
              | PREFIX.TrOne          ----> | PREFIX.ijcl  
-----
```

2 namelists :

&cost prefix=... (voir III.1.A)

ycl=

F (default) les integrales sont ecrite dans

PREFIX.ijcl en binaire

T les integrales sont en clair

fermi=

par symetrie, orbitales en-dessous du niveau de Fermi

s'il n'y a pas de gel dans motra, c'est en general

la meme donnee que pour le programme scf de molcas

apres la ligne "OCCUPIED" (default 8*0)

frozen=

par symetrie, orbitales gelees dans motra (ce sont

les memes nombres) (default 8*0)

/

3. PROGRAMME casdet

```
-----  
PREFIX.ijnam ----> casdet ----> | PREFIX.det  
                                | autres fichiers...  
-----
```

2 namelists :

```

&cdfil prefix=... / (voir III.1.A)
  fichiers dont on peut modifier le nom (voir III.1.A)
  ijnam (lu)
  det (ecrit pour casdi)
  cas (ecrit pour casdi)
  cdi (ecrit pour casdi)

```

```

&cd
noac= nombre d'actives (max=16)
numac= numero des actives
ms2= double de la valeur de Sz
is0= groupe de symetrie des fonctions d'onde cherchees
nprint= 0 ou 1 (default 0)
gener= espace genere :
      'CAS+SD' (CAS + Simples et Doubles) (default)
      'CAS+S' (CAS + Simples)
      'CAS+DDCI' (CAS + espace DDCI)
      'CAS+DDC2' (CAS + espace DDCI2)
      'CAS' (CAS seul)
      'SEL+SD' (esp selectionne + Simples et Doubles)
      'SEL+S' (esp selectionne + Simples)
      'SEL+DDCI' (esp selectionne + espace DDCI)
      'SEL+DDC2' (esp selectionne + espace DDCI2)
      'SEL' (esp selectionne seul)

```

si gener = SEL...

une namelist &mrs par determinant de reference fourni en donnee

```

&mrs
  ic= occupation du CAS. Si par exemple on veut occuper les
      orbitales 3 et 4 par 2 et un electrons : ic=3,3,4.
      ic=3,-3,-4 est aussi accepte, les signes '-' sont mis a +.
      ce qui permet eventuellement d'utiliser les fichiers
      ESSOUT ou ESCASOUT generes par le programme casdi, fichiers
      pour lesquels les spins sont specifies.
      Tous les equivalents de spin sont generes automatiquement

  it= trous dans les references (max=2 memes commentaires que pour
ic)

  ip= virtuelles occupees (max=2 memes commentaires que pour ic)

  file= nom de fichier contenant les namelists &mrs
        si "file" est donne (donc different du default file=' '), le
        programme cesse de chercher les determinants dans le
        fichier de donner, et les cherche dans le fichier "file"

  presel= fabrication d'un espace de determinants.
          methode : 2 passages casdet
          passage 1 : presel='S' ou 'SD' (default ' ')
          ' ' : pas de preselection
          'S' : on genere tous les simples a partir d'un espace
                de determinants donnes individuellement
          'SD' : idem, generation SD
          On donne ensuite les determinants generateurs
          &mrsel ic=.... /
          &mrsel ic=.... /
          ...

```

Attention, on ne peut donner que ic, les generateurs sont donc definis a partir du CAS

```
passage 2 :  
gener='SEL+S' out 'SEL+SD'  
&mrs file='PRESEL' /
```

```
exemple :  
passage 1 :  
&cdfil prefix='n2.' /  
&cd gener='CAS+S',noac=7,numac=2,3,4,5,10,13,15,nelac=8,  
presel='SD',nprint=0 /  
&mrsel ic=2,2,3,3,4,4,5,5, /  
&mrsel ic=2,2,3,3,4,4,5,10, /  
passage 2 :  
&cdfil prefix='n2.' /  
&cd gener='SEL+S',noac=7,numac=2,3,4,5,10,13,15,nelac=8,  
nprint=1 /  
&mrs file='PRESEL' /
```

Remarques :

- On peut donner plusieurs fois le meme determinant. Les redondants sont elimines
- Le programme casdet cree un fichier CASOUT qui contient tous les determinants du CAS sou la forme des namelists &mrs
- Le programme casdi cree un fichier ESSOUT de la forme :
 &ess it=...,ic=...,ip=...,v=... /
on peut utiliser ce fichier en remplaçant ess par mrs. Les signes "-" sont ignores, ainsi que les coefficients v
- TOUTES les orbitales qui, dans une donnee mrs sont :
 . non doublement occupees en dessous du niveau de fermi
 . non vides au-dessus
sont considerees comme ACTIVES (attention donc a la limite de 16 actives)

yf_act=

.true. (defaut) les orbitales actives declarees dans &cd restent les actives de casdi. Dans ce cas, on NE PEUT PAS utiliser les donnees it et ip

.false. la liste des orbitales actives est modifiee. Sont retirees celles toujours doublement occupees en-dessous du niveau de Fermi, et celles toujours vides au-dessus. Sont rajoutees toutes celles qui apparaissent dans it et ip.

&end

REMARQUE IMPORTANTE : quand gener = 'CAS+...', tous les determinants possibles sont generes. La seule limitation est le nombre de trous et particules. Par exemple, si gener = 'CAS+SD', tous les determinants possibles ayant 2 trous et 2 particules au plus sont presents, MEME S'ILS NE SONT PAS ENGENDRABLES A L'ORDRE 2 a partir du CAS.

Quand gener='SEL+...', c'est different. Seuls les determinants dont un equivalent de spin peut etre genere a l'ordre 2 a partir du CAS sont presents.

Ainsi, si, avec gener='SEL+SD', on donne le CAS complet en donnee,

on obtiendra MOINS de determinants que avec la donnee gener =
'CAS+SD'

4. PROGRAMME casdi

```
-----  
| PREFIX.ijnam          | PREFIX.davec  
| PREFIX.ijcl   ----> casdi   ----> | PREFIX.dens  
| PREFIX.det          | ...  
-----
```

```
&cdifil prefix=... /          (voir III.1.A)  
fichiers dont on peut modifier le nom (voir III.1.A)  
(donnees facultatives)  
ijnam   (lu)  
ijcl    (lu)  
det     (lu) (determinants generes par casdet)  
ess     (lu) (vecteur d'essai, resultat d'un calcul precedent)  
cdi     (lu, donnee rarement utilisee)  
sc2ex   (lu ou ecrit, voir "calculs SC2")  
dens    (ecrit, matrice densite)  
ymatoly (def : F) T : lit la matrice sur disque et la diagonalise,  
sans avoir besoin d'autre donnee ou fichier supplementaires
```

&dav

La namelist &dav contient un tres grand nombre de donnees
Cependant, toutes les donnees sont facultatives
D'autre part, seul un peiti nombre est a connaitre absolument

Donnees "tres utiles"

```
nvec=   nombre de valeurs propres (defaut 1)  
ihab=   0   diagonalisation (defaut)  
ihab=   1 ou 2   calcul SC2 (voir cette rubrique)  
iprec=  precision : seuil d'arret a 10**(-iprec) (defaut 6)  
niter=  nombre d'iterations (defaut 50)  
nessai= 0,1 ou 2 (defaut 0) vecteurs d'essai (voir cette rubrique)  
twvec=  seuil d'ecriture des coefficients des vecteurs propres  
sur la sortie  
twess=  seuil d'ecriture des coefficients des vecteurs propres  
dans le fichier ESSOUT (en clair)  
syspin= '0' (defaut) pour avoir tous les vecteurs propres  
'+' pour avoir les vecteurs symetriques par retournement  
des spins (singulets, quintuplets...)  
'-' pour avoir les vecteurs antisymetriques par  
Retournement des spins (triplets...)  
Attention, si ms2 est different de 0, syspin='0'  
ydens=  1 (defaut) calcul et ecriture sur file de la matrice  
densite  
ywcas=  T (defaut F) ecriture en sortie de tous les determinants  
du CAS. Creation d'un fichier ESCASOUT (en clair, comme  
ESSOUT)  
maxda2  nombre max de vecteurs gardes sur file
```

Autres donnees :

```

calcul= 'M', 'D' ou 'MD'
- calcul= 'M' : calcule seulement la matrice et la stocke sur
file a la maniere Moyen ou Inidet (reels en
simple precision)
- calcul= 'D' : diagonalise la matrice calculee par un passage
calcul= 'M' ou toute autre matrice ecrite de
la meme facon
- calcul= 'MD' : diagonalisation directe (default)

stop= 'FIN', 'DEL', 'ORDET', 'ITERNAT'
- stop='FIN' : (par default) le calcul va jusqu'a la fin
- stop='DEL' : le calcul va jusqu'a la fin, et n'efface pas les
fichiers
temporaires. Pour faire un restart
- stop='ITERNAT' : le calcul va jusqu'a la fin, et n'efface pas
Certains fichiers temporaires. Pour les calculs iternat.
Voir le mode d'emploi de ITERNAT
- stop='ORDET' : s'arrete apres avoir reordonnee les determinants en
fonction de leur adresse et avoir cree le fichier
correspondant ORDET.

restart= 'INT', 'DET', 'INTDET' ( apres un calcul stop='del' ou
eventuellement un calcul mal fini)
- restart= 'INT'. On ne reordonne pas les integrales, parce que
ce sont les memes pour 2 calculs (ex. meme geometrie,
2 symetries differentes)
- restart= 'DET'. On ne recalcule pas les listes de determinants. (CAS
de 2 calculs avec meme espace IC, mais 2 geometries
differentes par exemple)
- restart= 'INTDET'. Tout est conserve. Les 2 calculs sont identiques.
(ex. le 1er n'a pas converge, ou on a oublie de demander
un habillage ...)
- irest=1 : equivalent a restart= 'INTDET' (default 0)
- sint : seuil en dessous duquel une integrale moleculaire est
consideree comme nulle (default 0.d0)
- ydens : T ecriture de la matrice densite dans un fichier
Prefixdens (default T)

```

C. Vecteurs d'essai

&dav

```

nessai= 0, 1, 2 (default 0)
- 0 les vecteurs d'essai ont la forme suivante :

```

```

1. 0. 0. 0.
0. 0. 0. 0.
0. 1. 0. 0.
0. 0. 0. 1.
0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 1. 0.
0. 0. 0. 0.
.....

```

ou les "1" sont places sur les determinants dont les valeurs diagonales sont les plus basses. Si on utilise la symetrie de spin (donnee syspin), ces orbitales sont symetrisees, ce qui peut donner

un meilleur point de depart.
c'est un peu primaire, mais ca marche en general assez bien

On peut rajouter la donnee :
lvec= les numeros de vecteurs lus sur un fichier PREFIX.ess issu
d'un calcul precedent. par exemple
nvec=5, lvec=1,2 : on demande 5 vecteurs, les 2 premiers ont pour
vecteurs d'essai ceux du calcul precedent (utile quand on
s'aperçoit qu'on n'a pas demande assez de vecteurs).

- 1 les vecteurs d'essai sont donnees "en clair" :

- 1) en namelist sous la forme :

```
&ess it=... ic=... ip=... v=... /
&ess it=... ic=... ip=... v=... /
&ess it=... ic=... ip=... v=... /
&ess it=... ic=... ip=... v=... /
&ess it=... ic=... ip=... v=... /
.....
a raison d'une namelist par coefficient du vecteur pour
les coefficients importants

it : trous      it=3 : trou alpha en 3
                it=2,-4 : trou alpha en 2 et beta en 4
ip : particules meme principe
ic : CAS donne en OCCUPATION ic=2,-2,4 : 2 electrons
    alpha en 2 et 4, un beta en 2
v : coefficients (estimes) du (des) vecteur(s) propre(s)

Un fichier contenant des vecteurs d'essai donnees sous cette
forme est genere au cours de chaque calcul. Son nom est ESSOUT.
Il suffit de le coller a la fin des donnees. Le premier calcul
peut etre par exemple un CAS_CI (gener='CAS' dans la namelist
cd de CASDET. Si ywcas=T, on genere aussi un fichier ESCASOUT,
qui contient tout les determinants du CAS
```
- 2) par leurs numeros (peu commode)
ces numeros apparaissent dans la sortie de casdi, a droite des
coefficients des vecteurs propres

```
&ess idet=... /
```
- 3) par le nom d'un fichier contenant les donnees &ess
c'est commode si on veut utiliser un calcul precedent
exemple :

```
calcul CASCI avec ywcas=T
calcul CAS+SD avec nessai=1 et
&ess file='ESCASOUT' /
```

- 2 les vecteurs d'essai sont lus a partir d'un calcul
precedent sur le fichier PREFIX.ess
cd fichier PREFIX.ess a la meme structure que le fichier
PREFIX.davec correspondant a la sortie d'un calcul precedent

D. Details techniques

la dimension des vecteurs propres doit etre inferieure au parametre iadri
du fichier iadri.prm. En fait, il peut arriver que iadri doive etre un

peu plus grand (superieur aux valeurs qui s'affichent apres le titre "dimension des listes de determinants")
 le programme demande 2 vecteurs propres en memoire (donc 2 fois iadri)
 Il peut y avoir d'autres parametres a augmenter a la demande.

Jusqu'a present, tous les essais d'utilisation de memoire dynamique ont provoque un ralentissement non negligeable du programme. Cette option n'existe donc pas pour le moment.

Si on manque d'espace disque...

maxda2 : dimension maximale de l'hamiltonien effectif diagonalise dans le processus Davidson. On stocke sur disque 2.N.vecteurs, ou N est la dimension de l'hamiltonien effectif. Diminuer maxda2 pour economiser l'espace disque. Minimum=2*nvec. (defaut 50)

idelint : 1 efface le fichier d'integrales (ijcl) quand on n'en a plus besoin, c'est-a-dire avant les iterations. Il faudra les recalculer avant de faire un autre calcul, ou utiliser restart. (defaut 0)

ideldet : 1 efface le fichier de determinants issu de casdet apres l'avoir copie (reordonne) dans ORDET. Pour un autre calcul, on peut : - refaire casdet. - copier ORDET dans detcl. - utiliser restart. (defaut 0)

E. Calculs sc2

Les donnees sont dans la namelis &dav de casdi

La donnee qui gouverne les calculs SC2 est ihab

ihab = 0 diagonalisation simple

ihab = 1 calcul SC2

- si l'habillage se fait par rapport a l'etat fondamental et que celui-ci est couches fermees, pas d'autre donnee
- dans le cas contraire, on doit remarquer que l'habillage se fait par rapport :
 - * a un etat (on utilise ses coefficients)
 - * a un determinant (dominant en general dans cet etat)
- le determinant dominant est defini par la namelist deth &deth ith=... ,iph=... /
 ou ith et iph sont les trous et particules du determinant de reference par rapport a la REFERENCE COUCHES FERMEES occupee jusqu'au niveau de Fermi

exemple d'un systeme a 10 electrons, ou les orbitales 5 et 6 sont actives
 le determinant donne dans ess par
 &ess it=-2,3,ic=5,-6,ip=10,-12 /
 sera ecrit
 &deth ith=-2,3,-5,iph=-6,10,-12 /
 pour un nombre impair d'electrons, on ecrit un particule en moins
 &deth ith=-2,3,-5,iph=-6,-10, /
 de facon a avoir plus de spins beta que alpha

- l'etat de reference est CHOISI par le programme parmi les ihvec etats les plus bas : c'est celui pour lequel le determinant donne dans deth a le plus grand coefficient. ihvec = 1 par default. Il arrive que, au cours des iterations d'habillage, l'etat de reference "descende" au-dessous de ceux qui le precedaient. Il restera ainsi le vecteur "habilleur".

maxith nombre maximal d'iterations d'habillage. (default 10)

sh seuil d'arret du processus SC2. Si le coefficient du determinant de reference est inferieur a sh, le programme s'arrete (default 0.5)

ihab = 2 calcul SC2 par rapport a une autre reference
ce calcul suit necessairement un calcul ihab=1, et se sert du fichier PREFIX.sc2ex
Il permet d'obtenir les autres racines excitees de la Matrice SC2 obtenue avec ihab=1
ATTENTION : par rapport au calcul ihab=1, seule la symetrie de la fonction d'onde peut changer. En particulier, ni le nombre d'electrons, ni l'espace actifs ne peuvent etre modifiees.

yfspi : attention, peu teste !
pour un calcul SC2, oblige le vecteur a rester fonction propre de spin (ce que n'assure pas SC2)

yhmono : habillage par rapport aux monoexcites

Autres donnees concernant l'habillage

hab= 'SC2', 'ACPF', 'AQCC' (si ihab=1)

(default='SC2') fait un calcul ACPF ou AQCC au lieu de SC2

ipresc= precision pour l'habillage (10**(-iprec))

ihprec= dans un calcul SC2 : abaissement de la precision de diagonalisation pour les premieres iterations d'habillage (car on peut converger la diagonalisation de facon Approximative quand on n'a pas encore habille)
(default : 3 soit un abaissement de precision de 10**(-3))
mettre ihprec=0 si on veut connaitre le resultat du calcul SDCI

ihvec : nombre de vecteurs parmi lesquels on choisit le vecteur d'habillage. (default 1)

VI. Ijkloc

Programme de transformations d'intégrales moléculaires écrites en orbitales de symétrie à des intégrales moléculaires écrites en orbitales desymétrisées.

2 etapes : 1. desymetrisation des orbitales. C'est une partie complètement indépendante
2. transformation des intégrales, obtention des orbitales locales

Utilisation :

- Etape scf

- Etape localisation --> orbitales locales
- motra-molcost --> files ijcl,Info,Mono
- ijkloc
- exsci (casdiloc)

Fichiers :

- En entree : - file d'integrales ~ijcl
- ~Mono
 - ~Info
 - SCH_FERMI (creee par schmud)
 - ('~' =PREFIX)

- En sortie : - ~Ijcllocinfo (information pour exxc)
- ~ijclloc2 (integrales bielectroniques a 2 indices differents)
 - ~ijclloc3 (3 indices differents)
 - ~ijclloc4 (4 indices differents)
 - ~Infoloc (fichier de type Info, pour les orbitales locales)
 - ~ijclloc (eventuellement) fichier ijcl pour casdi
 - ~IJCLLOC (eventuellement) fichier IJCL pour casdi
 - DESYMORB orbitales desymetrisees
 - ORB_IJKLOC orbitales locales

Donnees :

Donnee (maj: correspond a des donnees obligatoires)

&ijkloc

Donnees obligatoires :

PREFIX='...'

Donnees facultatives :

		Défaut
nprint	0,1,2	0
symm	donnees de seward qui suivent la ligne "SYMMETRY" (dans le meme ordre !) si symm='SEWARD', la donnee est lue dans les donnees de seward (a condition que les donnees de seward et de ijkloc soient dans le meme fichier)	'SEWARD'
info1	file Info en entree	~Info
mono1	file Mono en entree	~Mono
ijcl1	file ijcl en entree	~ijcl
inporb	orbitales locales symetriques	INPORB
mem	memoire demandee en Mo	500
ycdloc	creation des files ijclloc* pour excci	T
ycd	T : genere des fichiers (ijclloc, Infoloc, Monoloc) Pour enchaîner sur casdi il faut renommer les files ijclloc, Infoloc, Monoloc en ijcl, Info, Mono	F
ycl1	(si ycd) T : ecriture en clair sur IJCLLOC	F

VII. Casdiloc

CI Program for localized orbitals. Long-range interactions between occupied and virtual orbitals are neglected

Input files:

Files created by ijkloc:
ijcllocinfo, ijclloc2, ijclloc3, ijclloc4
Monoloc, Infoloc (identical to Mono and Info of molcost)

eventually:
topo: links between orbitals

Data:

&locci

Compulsory data:

prefix=

Facultative data:

noac=	-1	number of active orbitals if noac=-1, noac,numac are recalculated using information from preceding programs.
numac=	16*0	active orbitals
ms2=	0	Sz*2
nessai=	0	trial vectors 0: on the lowest value(s) of the diagonal 1: given by the user. give &loccs namelist(s) 2: read from previous calculation (not yet implemented) 3: CAS prediagonalization
nvec	1	number of vectors
dim_heff	50	maximal size of the effective hamiltonian, i.e. which corresponds to the number of Psi and H*Psi stored on disk. Mustbe larger than twice nvec
nprint	0	0,1,2,3 print option
niter	50	number of iterations
calcul	'MD'	'MD' normal. Direct diagonalization 'M' Matrix calculation, stored on mat file diagonaliser ensuite avec casdi (calcul='D') il faut alors : - mettre ygendet=T dans &locci, pour avoir les det - renommer la matrice MAT en Project.mat
gener	'CAS+SD' 'SEL+SD'	CAS+SD, CAS+S, CAS+DDCI, CAS+DDC2, CAS Selected Active Space. The Active Space is given with individual space determinants. All spin distributions are authomatically generated. possible data: SEL+SD, SEL+S, SEL+DDCI, SEL+DDC2,

SEL

The determinants are given as follows:
after the &locci data: &locci ... /
&selec
for each determinant, a list of occupations in
I1 format (noac numbers 0, 1 or 2,
corresponding to the occupation of active
orbitals, according to
their position in the list of orbitals.
end the Data by a line FIN or END
example:

```

&lloci ... /
&selec /
2200
1111
end
syspin      '0'      '0' no spin symmetry
              '+' gives singlets, quintets...
              '-' gives triplets...
liens       'KIJ'    'KIJ' links obtained from exchange integrals
              'FILE' links given in toto ascii file
              'ALL' all orbitals linked
              'ATOM' see below "liens='ATOM'"
zone        ' '      To define different zones of the molecule
              with different sl thresholds
              default: one zone (the whole molecule), one sl
              ex of 3 zones: zone 1: a part of the molecule,
              defined by the atoms A1,A2,A3. zone 2: idem,
              atoms A4,A5. Zone 0: the rest of the molecule.
              Data:
              zone='A1,A2,A3','A4,A5',sl=0.03,0.01,0.001
              (zone 0: sl=0.03, zone 1: sl=0.01, zone 2:
              sl=0.001
ycaspair    F        T: when two bonds (occ,virt) are linked to the
              CAS, they automatically form a bond pair
sl          0.01     when liens='KIJ': threshold of the exchange
              integrals to consider two orbitals as linked.
s           0.03     threshold for printing of the determinants and
              coefficients of the eigenvector(s)
info        'Infoloc' suffix of the name of infoloc file
info1      ' '      name of the infoloc file. If info1=' ', the
              name is prefix//info
              other file names: same rules
              mono='Monoloc', topo, ijclloc2, ijclloc3,
              ijclloc4, davec,

***** Size Consistent Dressing
mhab        0        0: no dressing
              1: CEPA_0 dressing
              2: improved CEPA_0 dressing
              3: Class-dressed CI (improved MR-AQCC)
nhab        1        Reference eigenvector for dressing
iterh       2        davidson iteration after which dressing begins
note:      if mhab/=0  nessai is put to 3

*****if nessai=1:
for each vector:
&locess
  it=          holes
  ip=          particles
  c=          active occupation
  v=          0. coefficient
end of vector:
&locess c='FIN' / or
&locess c='END' /

example (nvec=2)

&locess c='20',v=1., /
&locess c='fin' /

```



```
&laccess c='+-',v=1. /
&laccess c='-+',v=-1. /
&laccess c='fin' /
```

```
*****if liens='ATOM':
```

1. Condition: each bond name (O_...) must contain the name of the atoms involved in the bond. No other information must contain the name of an atom (be careful with "sigma" if there is a "Si" atom...)
2. all couple of bonds containing the same atom are automatically linked
3. links with the CAS: data:

```
&loc_at
CAS
list of atoms
end
```

all bonds containing an atom of the list are linked to the CAS

VIII. Naturalt et locnats

Détermine un nouveau jeu d'OM à partir d'un mélange (moyenne, différence, etc...) des matrices densités utilisées, quel que soit la symétrie d'espace ou de spin des états calculés précédemment par casdi. Naturalt a été développé à Tarragone, locnats par Toulouse.

Donnees Naturalt:

```
&natfil prefix='...' /
&mixdat
nsym= nombre de calculs casdet/casdi en parallele (M caluls, M fichiers
      cdin)
netat=m,n,p,... nombre de val/vect propres par calcul
      (nsym valeurs donnees, pour N=m+n+p+... val/vect propres)
metat= numeros des racines dans chaque calcul
      (m+n+p+... valeurs)
coef= m+n+p+... valeurs : coefficients relatifs de chaque val/vect propre
ymolcas=t,
ngel= donnees "FROZEN" de molcost
ndel= donnees "DELETE" de molcost
yw... donnees pour l'impression
seuil= seuil de convergence. (= recouvrement max des orbitales)
      (defaut=1.d-4). Si la convergence est atteinte, un fichier est cree
```

Données Locnats :

```
&locn

obligatoires :
prefix=
orbv='orbitales_d'entree'
orbn='orbitales_de_sortie'

facultatives
```

```

ngel=  donnees frozen de motra
ndel=  donnees delete de motra
nmat=  nombre de fichiers de matrices densite
netat (1,nmat) nombre de vecteurs consideres dans chaque fichier
densmat (1,nmat) noms des fichiers
metat numeros des vecteurs consideres
coef poids sur les diff vecteurs
ex 3 fichiers, fich1 vect 2 ; fich2 vect 3 et 4 ; fich3 vect 1
nmat=3,netat=2,4,1,metat=2, 3,4, 1,
coef=1.,0.5,0.5, 1., 0.7, (par ex)

```

IX. Iternat

Programme d'optimisation d'orbitales moléculaires par moyenne d'orbitales naturelles. Ce programme existe toujours, mais noscf le remplace la plupart du temps (plus simple d'utilisation et moins de données pour le même résultat).

Le programme est un script écrit en kshell

UTILISATION

1) Appel du programme

On fait tourner une chaîne molcas ordinaire, jusqu'à obtenir des orbitales moléculaires. par exemple :

```

molcas run seward
molcas run scf
molcas run rasscf
molcas run rasread

```

puis on doit rajouter les 2 lignes :

```

cp $Project.RasOrb INPORB
iternat < iter_in

```

ou : - le script iternat doit être dans le path
- le fichier de données iter_in contient :

```

NITER
10                (nombre max d'iterations sur les OM)
MOTRA
motra.in          (fichier de données de motra)
MOLCSD
molcsd.in         (fichier de données de molcsd)
NATU
natu.in           (fichier de données de naturalt)
MOLCSDX
/work0/daniel/babel/exe/molcsd  (programme molcsd a executer)
CASDETX
det16s            (programme casdet a executer)
CASDIX
dil6s             (programme casdi a executer)
NATUX
naturalt          (programme naturalt a executer)
FIN

```

2) Conditions de fonctionnement

- on doit
- avoir necessairement la ligne :
export Project WorkDir CurrDir
 - appeler Project, WorkDir et CurrDir par ces noms-la, avec ces orthographes
 - avoir n fichiers de donnees pour l'ensemble casdet/casdi, si on optimise les orbitales sur n calculs differents.
 - ces fichiers doivent OBLIGATOIREMENT s'appeler cdin1, cdin2 ... cdinn
 - chaque fichier cdini ressemble a :

```
&cdfil prefix='co.' /
&cd noac=2,nelac=2,numac=3,6,is0=1,ms2=0,gener='DDCI' /
&cdifil prefix='co.' /
&dav nessai=0,internat=0 /
```

en particulier les donnees nessai=(0,1, ou 2) et internat=0 sont OBLIGATOIRES.

- avoir des donnees pour motra et molcsd
- avoir un fichier de donnees pour naturalt. Ce fichier doit s'appeler \$Project.natu. par exemple :

```
&natfil prefix='co.', /
&mixdat
nsym=2,metat=1,1,coef=0.5,0.5,ymolcas=t,
ywrn=t, ywrn=t,ngel=2,0,0,0,ndel=2,0,0,0, /
seuil=1.d-6
```

(voir plus bas les donnees de naturalt)

- les fichiers ijnam et ijcl pour casdet/casdi doivent s'appeler Project.ijnam et Project.ijcl (dans l'exemple ici

:

co.ijcl et co.ijnam).

3) Interventions sur le script internat :

PATH=\$PATH: ... : mettre dans le path casdet, casdi et naturalt
alias molcas=\$MOLCAS/ksh/molcas.ksh (a changer eventuellement)

4) Donnees de Naturalt

&mixdat

nsym= nombre de calculs casdet/casdi en parallele (M caluls, M fichiers cdin)

netat=m,n,p,... nombre de val/vect propres par calcul

(nsym valeurs donnees, pour N=m+n+p+... val/vect propres)

metat= numeros des racines dans chaque calcul

(m+n+p+... valeurs)

coef= m+n+p+... valeurs : coefficients relatifs de chaque val/vect

propre

ymolcas=t,

ngel= donnees "FROZEN" de motracsd

ndel= donnees "DELETE" de motracsd

yw... donnees pour l'impression

seuil= (seuil de convergence. (= recouvrement max des orbitales)

defaut=1.d-4)

X. Orb2q5

- Reads molcas Orb File (in old format)
- add to an existing Q5 File

Input files: - Q5 File
 - Orb File

Output files: - modified Q5 File

Input Data:

1. Name of Q5 File
2. tag to be written in modified Q5 file
3. tag of existing group (usually "default")
4. Name of Orb file

XI. Q52molden

Generates a molden file from Q5

Input files: - Q5

Output files: - molden file

Data files:

- Name of Q5 file
- MO tag (tag indentifying MO to be drawn)
- AO tag (usually "default")