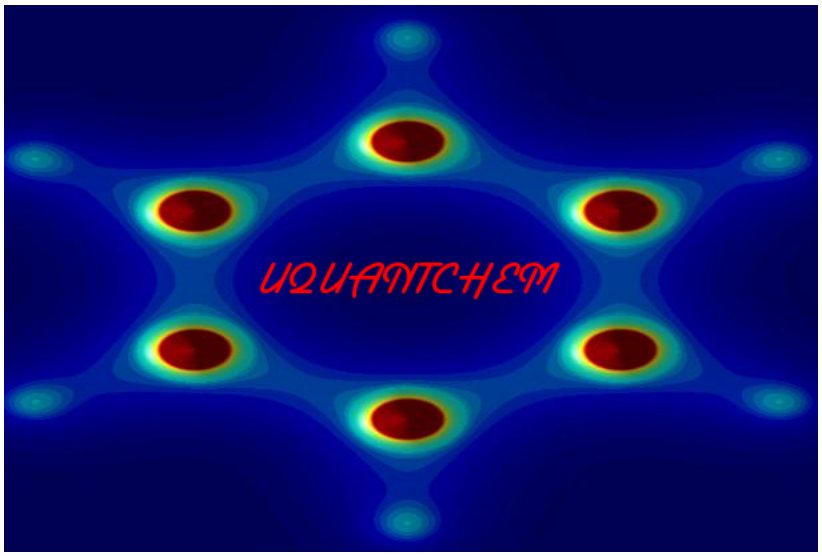
User manual for the Uppsala Quantum Chemistry package UQUANTCHEM

V.32



by
Petros Souvatzis
Department of Physics

Uppsala University 2012

Contents

Chapter 1

Introduction

The Uppsala Quantum Chemistry (UQUANTCHEM) project was started in order to build a transparent, from the point of view of a physicist, development platform for implementing new computational and theoretical ideas in quantum chemistry. The package is written in fortran90.

Due to the ambition of transparency, i.e being able to see the physics in-bedded within the source code, the code might not be as fast as other similar codes. Some of the initial inefficiency as been dealt with by parallelization, which hopefully has resulted in a "proof of principle" level of computational speed, which will be enough to test new ideas on medium sized molecules.

The ultimate goal is to use the platform of UQUANTCHEM to develop new computational schemes within the context of quantum Monte Carlo. .

The UQUANTCHEM software is published under the General Public License Version 3.0 (GPLv3.0). Thus any one is free to use the UQUANTCHEM software.

It is my sincere wish that you will enjoy using the UQUANTCHEM package as much as I have enjoyed writing it, and that it might help any freshman to better understand the techniques used in modern quantum chemistry.

Petros Souvatzis

Chapter 2

Compiling the code

To compile the code you need to have a fortran compiler installed on your machine together with Lapack. If you don't have Lapack installed the latest version of Lapack is provided together with the UQUANTCHEM package. There are several pre-made Make-files provided with the UQUANTCHEM package, which can be used as templates to create a Make-file that corresponds to the specifications of your particular system. Here an example of a more or less generic installation procedure will be given.

ALTERNATIVE 1 Let's assume you have the gfortran compiler available on your system and that you also have lapack and blas preinstalled on your machine. Now assume you want to compile the serial version of the the code, then follow the following steps:

(1) `V.32> cd SERIALVERSION`

(2) `V.32/SERIALVERSION> cp Makefile.gfortran.serial Makefile`

(3) In the `Makefile`, edit the line:
`LAPACKPATH = /Users/petros/UQUANTCHEM/Src/V.21/lapack-3.4.0`
so it reads:
`LAPACKPATH = Path-where-I-have-my-lapack-lib`

(4) In the `Makefile`, edit the line:
`BLASPATH =/Users/petros/UQUANTCHEM/Src/V.21/BLAS`
so it reads:
`BLASPATH = Path-where-I-have-my-blas-lib`

(5) `V.32/SERIALVERSION> make`

ALTERNATIVE 2 Let's assume you have the gfortran compiler available on your system and that you *do not* have lapack and blas preinstalled on your machine. Then you can compile the lapack and blas libraries that comes with the uqantchem package together with uquantchem by following these steps:

(1) `V.32> cd SERIALVERSION`

(2) `V.32/SERIALVERSION> cp Makefile.gfortran.nolapack.noblas.serial Makefile`

(3) In the `Makefile`, edit the line:
`LAPACKPATH = /Users/petros/UQUANTCHEM/Src/V.21/lapack-3.4.0`
so it reads:
`LAPACKPATH = Where-uquantchem-is-located-on-my-machine/lapack-3.4.0`

(4) In the `Makefile`, edit the line:
`BLASPATH =/Users/petros/UQUANTCHEM/Src/V.21/BLAS`
so it reads:
`BLASPATH = Where-uquantchem-is-located-on-my-machine/BLAS`

(5) `V.32/SERIALVERSION> make all`

And similarly if you want to install the openmp or the MPI-version of the code you do just move in to the `OPENMPVERSION`-directory or the `MPI_VERSION` -directory and perform the steps (2)-(5). In the case of the openmp/gfortran version the simplest way to install is to use the pre-made make-file named `Makefile.gfortran.openmp`, or if you have access to any of the Swedish super-computer clusters Lindgren or Neolith there are also pre-made Make files for the MPI-version of the code to make life easier.

Chapter 3

What Can be done with UQUANTCHEM

3.1 Hartree Fock Calculations

There are to types of Hartree-Fock types of Hartree-Fock implemented in UQUANTCHEM. The restricted Hartree-Fock (RHF), and unrestricted Hartree Fock (URHF). The implementation is based on expanding the molecular orbitals of the slater determinant in a basis set consisting of contracted gaussian primitive functions. The implementation is of text-book style based on the the book of Szabo Ostlund [3] and the book of Cook [5]. To calculate the electron electron repulsion integrals $(ij|kl)$ rys quadrature is used.

3.2 Configurational Interaction Calculations

Configuration interaction calculations is possible to perform with UQUANTCHEM with a basis set constructed by double and single excitations of the original Hartree-Fock slater determinant (CISD). For mor details see Szabo and Ostlund, p. 231-269. [3].

3.3 Möller Plesset Calculations (MP2)

Standard many body perturbation theory calculations up to second order, so called Möller Plesset Calculations (MP2), are also possible. For more details see Szabo and Ostlund, p. 350-353. [3].

3.4 Density Functional Theory Calculations (DFT))

Density functional theory calculations are possible to perform with UQUANTCHEM. The following functionals are available: LDA, PBE and B3LYP.

3.5 Time Dependent Density Functional Theory Calculations (TDDFT))

It is also possible to perform time dependent density functional calculations by real time propagation of the density matrix, P. Here, the quantum Liouville equation

$$i\frac{dP}{dt} = [F, P] \qquad , \tag{3.1}$$

is solved by the modified midpoint algorithm [4]

$$P(t + \Delta t) = e^{-2iF(t)\Delta t} P(t - \Delta t) e^{2iF(t)\Delta t}. \tag{3.2}$$

Here, F, is the Kohn-Sham Fockian/Hamiltonian of the system.

3.6 Quantum Montecarlo Calculations

It is possible to perform two types of quantum Monte Carlo calculations. Variational Monte Carlo (VMC) which is mainly used to generate good initial random walker configurations for the Diffusion Quantum Monte Carlo (DQMC) calculations. The implementation of DQMC in UQUANTCHEM follows closely the algorithm outlined in the work of Umrigar and Nightingale [10]. However, in UQUANTCHEM the trial function is constructed with a much simpler Jastrow factor, $\mathcal{J}$, and slater determinants are constructed from cusp corrected gaussian orbitals. The implementation of the cusp correction in UQUANTCHEM follows that described by S. Manten and A. Lüchow [11].

The explicit form of the trial function used for the importance sampling in the DQMC of UQUANTCHEM is given by:

$$\Psi_T = D_\uparrow D_\downarrow \mathcal{J}. \tag{3.3}$$

Where

$$\mathcal{J} = exp(\sum_{i<j} \frac{\delta \cdot \texttt{BJASTROW} \cdot r_{ij}}{(1 + \texttt{CJASTROW} \cdot r_{ij})}, \tag{3.4}$$

$r_{ij} = |r_i - r_j|$ is the distance between electron i and j, $D_\uparrow$ and $D_\downarrow$ are the slater determinants created from spin up respectively spin down orbitals. The orbitals are constructed from the URHF self consistent solution. Here $\delta = 0.25$ if the spin of the electrons i and j are identical otherwise if the spins are opposite $\delta = 0.5$.The jastrow parameters `BJASTROW` and `CJASTROW` are described the section below discussing the input parameters.

3.7 Born Oppenheimer Molecular Dynamics

Molecular dynamics calculations can be performed within the computational framework provided by uquantchem. The nuclei are here propagated with Newtons classical equations of motion, in the context of the Born-Oppenheimer approximation. The inter-atomic forces are calculated analytically from the gradients of the Hartree-Fock energy (RHF or URHF) with respect to the nuclear positions. The extended Lagrangian Molecular Dynamics formalism (XL-BOMD) [12, 15, 16] has been implemented providing a total energy almost completely

free of drift. Furthermore the Fast First Principles Molecular Dynamics formalism (FFP-MD) has also been implemented providing a very stable time propagation without employing self consistency!

Chapter 4

Setting up a UQANTCHEM calculation

4.1 The input files

There are only two input files that one needs to provide the UQUANTCHEM code, the `INPUTFILE`-file contains the required information about which atoms are present in the molecule, there positions and the level of approximation used to deal with the electron correlation. On top of this basic information the user can provide more detailed information about convergence criteria and parameters that deals with other levels of approximation or details about the calculation. These non-basic parameters have been given more or less reasonable default values so that to enable un experienced users to get up in the "air" quickly without being weighted down with to many technical details.

The second input file needed is the `BASISFILE`-file, containing information about the basis set to be used by UQUANTCHEM. More specifically, the `BASISFILE`-file contains information about the orbital quantum numbers of the basis functions the gaussian exponents and the contraction coefficients.

4.1.1 The `INPUTFILE`-file

In the the sub-directory `V.32/EXAMPLE_INPUT_OUTPUT/` several example input-files can be found. Here we only give an example of a INPUT-file specifying a URHF calculation of a water molecule:

```
CORRLEVEL URHF
TOL 1.0E-8
Ne 10
NATOMS 3
ATOM 1 0.453548746355979 1.751220869758844 0.0000000
ATOM 8 0.000000000000000 0.000000000000000 0.0000000
ATOM 1 -1.809000000000000 0.000000000000000 0.0000000
```

4.1.2 The BASISFILE-file

In the sub-directory V.32/BASIS several basis files can be found. To use a specific type of gaussian basis set just copy the file containing the basis you want to use in your calculation from the V.13/BASIS directory to the BASISFILE-file. For example if you want to use the $6-31G^{**}$ basis set just type:

cp ../V.32/BASIS/6-31GST-ST.dat BASISFILE. A word of caution should be said about the BASISFILE-file. **IN ORDER FOR THE CORRECT BASIS TO BE USED FOR A MOLECULE CONSISTING OF ATOMS WITH ATOMIC NUMBERS $Z_1 \leq Z_2 \leq \cdots \leq Z_n$, ALL ATOMS WITH ATOMIC NUMBERS UP TO Z_n MUST BE SPECIFIED IN THE BASISFILE-file FOR THE UQUANTCHEM PROGRAM TO WORK CORRECTLY!**

Unfortunately, the above specification might not always be fulfilled for some combinations of atoms when using some of the BASISFILE-files stored in the V.32/BASIS subdirectory. So take care!.

If there is a basis not provided with the current UQUANTCHEM distribution you might find it on the basis set cite: https://bse.pnl.gov/bse/portal. Here you can by means of cut and past create more basis set files. This is done by the following procedure:

(1) Go to https://bse.pnl.gov/bse/portal

(2) On the leftmost scroll menu there are a plethora of basis sets defined. Scroll down to the basis set you want to use and mark it.

(3) On the periodic table in the middle of the page there will now appear orange colorings in the left bottom corners of the atom for which a basis exists.

(4) Mark the atoms for which you want to use the basis

(5) Select the turbomole format and click on the bottom marked "Get Basis Set".

(6) Now there will appear a window containing the basis set information you require. Copy this and past it into a file BASNAME.raw located in the same directory as the perl-script rawtofortranformat.pl (located in the the sub-directory V.32/BASIS)

(7) replace the third line in the perl-script reading `$header = "aug-pcS-4";`
with `$header = "BASNAME";`.

(8) run: ./rawtofortranformat.pl and the file BASNAME.dat will be created. This new file can be used by UQUANTCHEM.

4.1.3 The MOLDYNRESTART.dat-file

In this file contains the information needed to continue a Molecular Dynamics Calculation. The file contains the following information: 1-st line of file contains the time step index

(integer), the following NATOMS lines contain three columns with the atomic positions corresponding to the time-step index of the first line, the NATOMS lines following the atomic positions contain three columns with the velocities of the atoms corresponding to the time-step index of the first line, finally the last NATOMS lines contain three colums with the interatomic forces of the atoms corresponding to the time-step index of the first line. IF THE FILE MOLDYNRESTART.dat EXISTS IN THE RUNNING DIRECTORY OF UQUANTCHEM THE CALCULATION WILL BE AUTOMATICALLY CONTINUED. TO RESTART A MOLECULAR DYNAMICS CALCULATION FROM SCRATCH BE SURE TO REMOVE THIS FILE FROM THE RUNNING DIRECTORY.

4.1.4 The INITVELO.dat-file

This file, if provided, is used in a Molecular Dynamics Calculation to specify the initial velocities of the atoms. The file should contain three columns and NATOMS rows of real numbers specifying the initial velocities of the atoms in $[au]$. This file is **not** read if the file MOLDYNRESTART.dat is present in the running directory.

4.1.5 Running Uquantchem

To run, for instance the serial version of UQUANTCHEM, just simply run the command:
RUN_DIR> ./uquantchem.s

in the same directory where the INPUTFILE and BASISFILE are located.

4.2 Input parameters

4.2.1 CORRLEVEL

Type: Character. (Default = None)
CORRLEVEL = {URHF,RHF,CISD,MP2,VMC,DQMC,LDA,PBE,B3LYP}

Used for setting the level of electron correlation employed in the calculation. URHF = Un restricted Hartree-Fock, RHF=Restricted Hartree-Fock, CISD=Configuration Interaction Calculation (singles and Doubles), MP2=Möller-Plesset second order perturbation theory, VMC=Variational Monte Carlo, DQMC=Quantum Diffusion Monte Carlo, LDA= DFT calculation using the Local Density Approximation (LDA) in the spirit of Vosko Wilk and Nusair [1], PBE = DFT calculations using the gradient corrected functional of Perdew, Burke, and Ernzerhof (PBE) [2], B3LYP = DFT [1, 17, 18, 19] calculations using the hybrid functional of B3LYP.

4.2.2 ADEF

Type: Logical. (Default = .FALSE.)
If true then all the time independent calculations will be performed with a static field present. In the case of a molecular dynamics calculation, depenpending on the parameter EPROFILE, the field will have different time evolutions during the molecular dynamics run (see section below describing EPROFILE). The option EPROFILE=DP does not work when ADEF=.TRUE..

There is also an extra option for EPROFILE (not available when when DOTDFT=.TRUE. , i.e for TDDFT/THF calculations), and that is EPROFILE='ST', which will result in all calculations being perfomed in the presence of a static electric field of strength EFIELDMAX.

When doing molecular dynamics calculations in the presence of a time-dependent external electric field, the electrons are assumed to follow the field adiabatically. Therefore great care should be taken to make sure that the system evolves as closely as possible to the ground state of the system. The validity of the adiabatic approximation might be evaluated by calculating the occupation numbers for the hartree-fock/Kohn-Sham orbitals as a function of time by performing a TDDFT/THF calculation with the same field that is going to be used in the MD-run. If the fluctuations of the orbital occupation numbers are not to severe the adiabatic approximation can be used at least with some confidence.

4.2.3 DOTDFT

Type: Logical. (Default = .FALSE.)
If true, then a time dependent density functional calculation (TDDFT) is performed if CORRLEVEL= {LDA,PBE,B3LYP}, otherwise if CORRLEVEL= URHF, then a time dependent Hartree-Fock calculation is performed. This has only been implemented in the MPI and OPENMP version of the code.

4.2.4 EPROFILE

Type: Character. (Default = 'HO')
Defines weather or not the electric field used in a TDDFT/TDHF calculation is homogeneous or not and the modulation, $\epsilon(t)$, of the field. If the EPROFILE='HO' then the amplitude of the electric field is defined by $E(t, \mathbf{r}) = \epsilon(t)sin(\omega t)$ or if EPROFILE='AC' then the electric field is defined by $E(t) = \epsilon(t)sin[\omega t + (c/\omega)\hat{n} \cdot \mathbf{r}]$, where the modulation, $\epsilon(t)$, is defined by:

$$\epsilon(t) = \texttt{EFIELDMAX} \cdot \left(\frac{\omega t}{2\pi}\right) \quad if \quad t < \frac{2\pi}{\omega}$$
$$\epsilon(t) = \texttt{EFIELDMAX} \quad if \quad \frac{2\pi}{\omega} \leq t < (\texttt{NEPERIOD} + 1)\frac{2\pi}{\omega}$$
$$\epsilon(t) = \texttt{EFIELDMAX} \cdot (\texttt{NEPERIOD} + 2 - \frac{\omega t}{2\pi}) \quad if \quad (\texttt{NEPERIOD} + 1)\frac{2\pi}{\omega} \leq t < (\texttt{NEPERIOD} + 2)\frac{2\pi}{\omega}$$
$$\epsilon(t) = 0.0 \quad if \quad t \geq (\texttt{NEPERIOD} + 2)\frac{2\pi}{\omega}$$

Here $\hat{n}$ is the propagation direction of the Electric field (see EDIR), and c the speed of light in vacuum. If EPROFILE='DP' the electric field has been assumed to be a Dirac pulse and equal to EFIELDMAX for $t = 0$ (Just at the precise moment when the TDFT/TDHF time propagation starts), and zero for $t > 0$. This modulation is used to calculate absorption spectra since it corresponds to a Dirac perturbation which is a superposition of all frequencies. Here $\omega =$ OMEGA.

4.2.5 DOABSSPECTRUM

Type: Logical. (Default = .TRUE. if INT(TEND/TIMESTEP) $<$ 10000 otherwise Default = .FALSE.)

If true, then the Fourier-transform of the dipole moment is calculated and stored to the file `ABSSPECTRUM.dat`

4.2.6 NEPERIOD

Type: Integer. (Default = 1)
Number of periods that the amplitude modulation, $\epsilon(t)$, is equal to `EFIELDMAX`.

4.2.7 EFIELDMAX

Type: Double. (Default = 0.030 a.u.)
The strength of the electric dipole field used in a TDFT or a TDHF calculation.

4.2.8 EDIR

Type: Integer (Default = 1)
Defines the propagation direction and polarization of the electric dipole field used in a TDFT/THF calculation. If `EDIR`=1, then the field propagates in the x-direction and is polarized along the y-direction, if `EDIR`=2, then the field propagates in the y-direction and is polarized along the z-direction and if `EDIR`=3, then the field propagates in the z-direction and is polarized along the x-direction. The direction of travel is only important in the case when `EPROFILE='AC'` (Inhomogeneous fields) and the wavelength is comparable to the molecule. In the case of `EPROFILE='DC'` or `EPROFILE='HO'` `EDIR` only defines the polarization of the Electric field.

4.2.9 OMEGA

Type: Double. (Default = 0.076 a.u.)
The frequency of the electric field used in a TDFT or a TDHF calculation if `EPROFILE='AC'` or `EPROFILE='HO'`. The default frequency corresponds to the vacuum wavelength of $\lambda = 600$ nm.

4.2.10 NCHEBGAUSS

Type: Integer (Default = 100)
The number of radial mesh points used in the Chebushev-Gauss quadrature used to integrate the exchange correlation energy and exchange correlation potential matrix elements in the radial direction.

4.2.11 NLEBEDEV

Type: Integer (Default = 3)
This integer is used to choose the angular mesh employed by the Lebedev quadrature which is used to integrate the exchange correlation energy and exchange correlation potential matrix elements in on a spherical surface. The integer and its corresponding number of mesh points are the following:
`NLEBEDEV`=$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24\}$
$\rightarrow$ `Number of mesh points =` $\{110, 170, 194, 230, 266, 302, 350, 434, 590, 770, 974, 1202,$

1454, 1730, 2030, 2354, 2702, 3074, 3470, 3890, 4334, 4802, 5294, 5810}. Thus the default number of angular mesh points is 194.

4.2.12 MOLDYN

Type: Logical (Default = .False.)
Flag specifying weather or not to perform a Molecular Dynamics Calculation. If `MOLDYN= .TRUE.` a Molecular Dynamics Calculation will be performed.

4.2.13 XLBOMD

Type: Logical (Default = .False.)
Flag specifying weather or not to perform a Molecular Dynamics Calculation using the time reversible propagation algorithm by A.M Niklasson [12] for updating the density matrix at each time step. If `XLBOMD` is .TRUE. a Molecular Dynamics Calculation using the A.M Niklasson scheme will be performed.

4.2.14 SOFTSTART

Type: Logical (Default = .False.)
This flag, if true, specifies the following start up scheme when doing XL-BOMD calculations:

(1) FIRST 50 time-steps full scf XL-BOMD with high level (low order) of dissipation, `DORDER` = 4

(2) Next 50 time-steps fast-QMMD still with high level (low order) of dissipation, `DORDER` = 4

(3) Rest of the calculation is done with XL-BOMD or fast-QMMD together with the order of dissipation (`DORDER`) that has been specified by the user in the INPUTFILE. Here depending on the user input, XL-BOMD or fast-QMMD is run.

4.2.15 DORDER

Type: Integer (Default = 8)
Determines the order of the dissipative force term used in the time propagation of the density matrices [16]. The order of the dissipative force equals `DORDER-1`. Is only used if `XLBOMD` is `.TRUE.`

4.2.16 ALPHA

Type: Double Precision (Default = depending on `DORDER`, see Table I in in [16])
Parameter used in the XLBOMD time propagation of the density matrix. Only used if `XLBOMD` is .TRUE.

4.2.17 KAPPA

Type: Double Precision (Default = depending on DORDER, see Table I in [16])
Parameter used in the XLBOMD time propagation of the density matrix. Only used if XLBOMD is .TRUE.

4.2.18 ZEROSCF

Type: Logical (Default = .False.)
If this flag is set to .TRUE. then when performing a Molecular Dynamics Calculation, self consistent calculation to obtain the density matrix is only performed for the initial 10 (first) time steps. This feature is only to be used together with XLBOMD set to .TRUE. since otherwise the density matrix will be constant throughout the MD calculation. The combination of setting both the flags ZEROSCF and XLBOMD equals the method of Fast Quantum Mechanical Molecular Dynamics (FQMMD) as is described by Niklasson et al [15].

4.2.19 ZEROSCFTYPE

Type: Integer (Default = 1) Only available in the OMP and MPI version.
In the case of ZEROSCF = .TRUE. this parameter selects which of two possible energy expressions to use when calculating the total energy and the interatomic forces when performing a molecular dynamics calculation using the XLBOMD scheme with only 1 scf cycle per time step. For simplicity we here assume a non-spin polarized RHF calculation or a non spin-polarized LDA calculation to exemplify the two energy expressions.

ZEROSCFTYPE = 1
$E = 2Tr[hD] + Tr[DG(D)], \qquad (RHF)$
$E = 2Tr[hD] + Tr[DG(D)] + E_{xc}(D), \qquad (LDA)$

ZEROSCFTYPE = 2
$E = 2Tr[hD] + Tr[(2D - P)G(P)], \qquad (RHF)$
$E = 2Tr[hD] + Tr[(2D - P)G(P)] + E_{xc}(D), \qquad (LDA)$

Here: $G = \frac{1}{2}[J - \frac{1}{2}K]$, in the case of a RHF calculation and $G = \frac{1}{2}J$, in the case of a LDA calculation P = density matrix propagated by XLBOMD-scheme $D = \Theta[\mu - F[P]]$, F[P] = Fockian obtained using the density matrix P. Θ = step function. μ = chemical potential, h = one electron hamiltonian and E_{xc} = exchange correlation energy functional.

4.2.20 FIXNSCF

Type: Integer (Default = -1)
This is a parameter that if set such that FIXNSCF>0, in a molecular dynamics calculation, the number of scf cycles are kept constant and equal to FIXNSCF. However, for the first 10 time steps the total energies are fully converged with respect to the energy tolerance TOL.

4.2.21 MOVIE

Type: Logical (Default = .False.)
Flag specifying weather or not to save the atomic positions of every every `SAMPLERATE`:th time-step in a Molecular Dynamics Calculatio. If set to .TRUE. the atomic positions of every `SAMPLERATE`:th time-step will be saved to the file `MOLDYNMOVIE.xsf`.

4.2.22 WRITEONFLY

Type: Logical (Default = .False.)
If true, the total energy, kinetic energy, potential energy and temperature will be written on the screen and into the file "`MOLDYNENERGY.dat`" during the course of a Molecular Dynamics Calculation. If .FALSE. the energies will be saved to the same file in the end of the calculation and no output will be written to the screen.

4.2.23 TEMPERATURE

Type: Double Precision. (Default = 300)
The temperature used to set the initial velocities in a Molecular Dynamics Calculation. However, if the file `INITVELO.dat` is present in the run directory of the calculation the initial velocities will be set to those velocities specified in the file `INITVELO.dat`, and the value `TEMPERATURE` will be ignored.

4.2.24 CFORCE

Type: Logical (Default = .False.)
Flag specifying weather or not to calculate the interatomic forces. If `CFORCE = .TRUE.` the interatomic forces will be calculated. For the interested user, the exchange correlation contribution of the force has been calculated in a similar fashion as described by Johnson *et al* [20]

4.2.25 PULAY

Type: Integer (Default = 4)
By setting this parameter the user can select between 4 different ways to calculate the Pulay contribution to the interatomic forces. This feature is mainly present to enable tests of the numerical stability of the time reversible XLBOMD algorithm. All four different approaches are equivalent and only differ due to numerical noise. If `PULAY` is equal to **1** then the following expression is used to calculate the Pulay contribution to the force [13]:

$$F_{Pulay} = 2\sum_i \epsilon_i C_i (\nabla S) C_i \tag{4.1}$$

If `PULAY` is equal to **2** then the following expression is used to calculate the Pulay contribution to the force:

$$F_{Pulay} = 2Tr[S^{-1}FP\nabla S] \tag{4.2}$$

If `PULAY` is equal to **3** then the following expression is used to calculate the Pulay contribution to the force [15]:

$$F_{Pulay} = Tr[(S^{-1}FP + PFS^{-1})\nabla S] \tag{4.3}$$

If PULAY is equal to **4** then the following expression is used to calculate the Pulay contribution to the force [14]:

$$F_{Pulay} = 2Tr[FP(\nabla S)P] \tag{4.4}$$

Here, F is the Fockian, ϵ_i and C_i are the eigenvalues respectively eigenvectors of the matrix equation $FC_i = \epsilon_i SC_i$, S, the overlap matrix and P equals the density matrix:

$$P_{\mu\nu} = \sum_i C_{\mu i} C_{\nu i}. \tag{4.5}$$

4.2.26 TOL

Type: Double Precision. (Default = 1.0E-6)
Used to set the convergence criterion for the self consistent field calculations, i.e the self consistent Hartree-Fock calculations are terminated when the difference between the total energy of two consecutive iterations is < TOL.

4.2.27 RELAXN

Type: Logical (Default = .False.)
Flag specifying weather or not to perform a structure relaxation calculation. If RELAXN=.TRUE. uquantchem performs a structure relaxation calculation.

4.2.28 RELALGO

Type: Integer. (Default = 1)
Only used if RELAXN = .TRUE.. Selects which type of relaxation algorithm to be used when optimizing the atomic positions. If RELALGO = 1, then the optimization will be done by searching for the atomic positions for which all the forces are zero. Here caution must be taken since there is no guarantee that the atomic configuration might end up in a molecular structure corresponding to a local energy maximum. Thus, even though this is by far the most effective algorithm as compared to RELALGO = 2, always keep an eye on the total energy change during the relaxation run. If RELALGO = 2, then the optimization will be done by searching for a minimum of the total energy with respect to the atomic positions.

4.2.29 FTOL

Type: Double Precision. (Default = 1.0E-4)
Used for structure relaxation calculations (RELAXN = .TRUE.). The relaxation of the structure stops when the mean force is < FTOL, or if the number of relaxation moves > NSTEPS. Be careful with using FTOL<1.0E-5, since it might require a huge number of iterations before the structure is relaxed within the prescribed force tolerance.

4.2.30 NSTEPS

Type: Integer. (Default = 500)
Used for structure relaxation calculations (RELAXN = .TRUE.). NSTEPS is the maximum number of relaxation moves permitted.

4.2.31 DR

Type: Double Precision. (Default = 0.5)
Only used if RELAXN = .TRUE.. If RELALGO =2 the parameter DR defines the interval $[0, \texttt{DR}]$ along the gradient which the energy is minimized on. Upon this interval the energy is calculated at NLSPOINTS number of points and fitted (least squares) to a polynomial of degree PORDER-1.
(Only used for structure relaxation calculations RELAXN = .TRUE.). If RELALGO =1, parameter DR is the maximum distance in au, that the atoms are allowed to move in one Newton-Raphson cycle.

4.2.32 NLSPOINTS

Type: Integer. (Default = 10)
Only used if RELAXN = .TRUE.. Here NLSPOINTS = number of points in the interval $[0, \texttt{DR}]$ where the energy is sampled. The data $\{DR_i, E(DR_i)\}_{i=1}^{\texttt{NLSPOINTS}}$, is least squares fitted to a polynomial of degree PORDER-1.

4.2.33 PORDER

Type: Integer. (Default = 7)
Only used if RELAXN = .TRUE.. Here PORDER equals the degree+1 of the polynomial used in the least squares fitting employed in the line-search of the conjugate gradient relaxation of the nuclei.

4.2.34 MIX

Type: Double Precision. (Default = 0.0)
Mixing parameter used to linearly mix the density matrix of the previous, P_{i-1}, iteration with the density matrix of the current iteration, P_i, in a RHF or a URHF calculation. I.e

$$P_i' = P_i(1 - \texttt{MIX}) + P_{i-1}\texttt{MIX} \tag{4.6}$$

4.2.35 ETEMP (Only used for CORRLEVEL = URHF,LDA,PBE,B3LYP)

Type: Double Precision. (Default = -1.0) (Only used for CORRLEVEL = LDA,PBE,B3LYP)
Electronic temperature used to calculate the electronic free energy in the case of a DFT-calculation and a URHF-calculation. If ETEMP> 0, the density matrix, $P_{\mu\nu}$, will be calculated by occupying the states according to Fermi-Dirac statistics, i.e:

$$P_{\mu\nu} = \sum_{i=1}^{Ne} C_{\mu i} C_{\nu i}^{\dagger} \Big[1 + e^{\beta(\epsilon_i - \mu)}\Big]^{-1}. \tag{4.7}$$

Here $\beta = 1/(K_B\texttt{ETEMP})$, μ is the chemical potential and ϵ_i are the energy eigenvalues of the eigenvalue equation $FC_i = \epsilon_i SC_i$, where F is the Kohn-Sham fockian/hamiltonian. Note that if ETEMP> 0 the parameter PULAY will be automatically set to 2, regardless of the specification of PULAY made in the INPUFILE. The entropy, $\mathcal{S}$, is calculated according to:

$$\mathcal{S} = -K_B \sum_i \Big[n_i ln(n_i) + (1 - n_i) ln(1 - n_i)\Big]. \tag{4.8}$$

Where,

$$n_i = \left[1 + e^{\beta(\epsilon_i - \mu)}\right]^{-1}. \tag{4.9}$$

4.2.36 DIISORD

Type: INTEGER. (Default = 3, (MAX = 25))
Mixing parameter equal to half of the order of the direct inversion of the iterative subspace method (DIIS) [7, 8]. If DIISORD=0 (DEFAULT) the DIIS mixing scheme is not used. The number 2DIISORD equals the number of density matrices mixed together according to:

$$P = \sum_{i=1}^{\texttt{2DIISORD}} \alpha_i P_i \qquad , \qquad \sum_{i}^{\texttt{2DIISORD}} \alpha_i = 1. \tag{4.10}$$

such that: $\Delta \mathbf{e}'^2 = 0$ in the mean square sense, where:

$$\Delta \mathbf{e}' = \sum_{i=1}^{\texttt{2DIISORD}} \alpha_i \mathbf{e}'_i \tag{4.11}$$

and,

$$\mathbf{e} = FPS - SPF, \qquad \mathbf{e}' = S^{-1/2}\mathbf{e}S^{-1/2}. \tag{4.12}$$

Here F is the Fockian, P the density matrix used to calculate the Fockian F, S the overlap matrix and $S^{-1/2}$ the Löwdin ortogonalization matrix. . The index refers to the number of iterations performed in the scf cycle.

4.2.37 DIISSTART

Type: INTEGER. (Default = 50)
The DIIS-mixing is used if $\Delta P^2 < 1$ or until the number of scf iterations > DIISSTART.

4.2.38 Ne

Type: Integer. (Default = Total Nuclear charge of Molecule)
Total number of electrons in the molecule.

4.2.39 NATOMS

Type: Integer. (Default = None)
Specifies the total number of atoms in the molecule. After this entry there must follow an equal number of rows specifying the atomic species and there spatial positions given in cartesian coordinates. The only entries allowed after the NATOMS entry are the rows specifying the atomic species and positions!

4.2.40 ATOM

Type: Integer, Double Precision, Double Precision, Double Precision. (Default = None)
Four numbers on the same row specifying the atomic number of an atom in the molecule and its corresponding position in cartesian coordinates. The first number equals the atomic

number and the three following numbers are the cartesian coordinates, x, y, and z of the atom. The entries containing information on the atomic numbers and positions must follow immediately after the NATOMS-entry.

4.2.41 WRITECICOEF

Type: Logical (Default = .False.)
Flag specifying weather or not to save the expansion coefficients of the Configuration interaction (CI) wave function to file or not. If WRITECICOEF=.TRUE. then the CI-wave function expansion coefficients are saved to the file CIEXPANSIONCOEFF.dat.

4.2.42 DIFFDENS

Type: Logical (Default = .False.)
If true and if DOTDFT = .TRUE. and WRITEDENS = .TRUE. the differentiated charge density defined by

$$\begin{aligned}\Delta\rho^{ex}(\mathbf{r},t_j) = &\sum_{i=1}^{N_e^{\uparrow}}\sum_{\mu,\nu=1}^{N_B}(1-n_i^{\uparrow})(t_j)C_{\mu i}^{\uparrow\dagger}(0)C_{\nu i}^{\uparrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &-\sum_{i=N_e^{\uparrow}+1}^{N_B}\sum_{\mu,\nu=1}^{N_B}n_i^{\uparrow}(t_j)C_{\mu i}^{\uparrow\dagger}(0)C_{\nu i}^{\uparrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &+\sum_{i=1}^{N_e^{\downarrow}}\sum_{\mu,\nu=1}^{N_B}(1-n_i^{\downarrow})(t_j)C_{\mu i}^{\downarrow\dagger}(0)C_{\nu i}^{\downarrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &-\sum_{i=N_e^{\downarrow}+1}^{N_B}\sum_{\mu,\nu=1}^{N_B}n_i^{\downarrow}(t_j)C_{\mu i}^{\downarrow\dagger}(0)C_{\nu i}^{\downarrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\end{aligned} \tag{4.13}$$

where, t_j, is given by

$$t_j = \frac{j}{24}\texttt{TEND} \quad , j = 0,\ldots,23 \tag{4.14}$$

is written to the files DENSEXITED_0.xsf, . . . ,DENSEXITED_23.xsf.

4.2.43 WRITEDENS

Type: Logical (Default = .False.)
Flag specifying weather or not to save the charge density to file or not. If WRITEDENS =.TRUE. the the charge density is saved to the file CHARGEDENS.dat. The format of the file is the following: At each row of the file the rightmost number corresponds to the charge density at a spatial point, $r = (x, y, z)$, specified by the three preceding integers on the same row. These integers, lets call them I, J and K, correspond to cartesian coordinates by the

following mapping:

$$x(I) = -\texttt{LIMITS(1)} + 2\frac{\texttt{LIMITS(1)}(I-1)}{\texttt{MESH(1)}-1}$$
$$y(J) = -\texttt{LIMITS(2)} + 2\frac{\texttt{LIMITS(2)}(I-1)}{\texttt{MESH(2)}-1}$$
$$z(K) = -\texttt{LIMITS(3)} + 2\frac{\texttt{LIMITS(3)}(I-1)}{\texttt{MESH(3)}-1}$$

4.2.44 MESH

Type: Integer, Integer, Integer. (Default = 100 100 100)
Specifies the mesh grid to be used when saving the charge density, the homo-density and the lumo-density to file. For more details on the use of integers in `MESH` see the section about the `WRITEDENS` parameter.

Also, if `DOTDFT = .TRUE.` and if `WRITEDENS =.TRUE.` the charge density or differentiated charge density is saved to the files `DENSEXITED_0.xsf`, . . . ,`DENSEXITED_23.xsf`.

4.2.45 LIMITS

Type: Double Precision, Double Precision, Double Precision. (Default = 5.0 5.0 5.0)
Specifies the the part of space in which the charge density, the homo-density and the lumo-density is to be calculated and saved to file. The part of space in which these densities are to be calculated is defined by:

$$x \in [-\texttt{LIMITS(1)}, \texttt{LIMITS(1)}]$$
$$y \in [-\texttt{LIMITS(2)}, \texttt{LIMITS(2)}]$$
$$z \in [-\texttt{LIMITS(3)}, \texttt{LIMITS(3)}]$$

4.2.46 HFORBWRITE

Type: Logical (Default = .False.)
If true the Hartree-Fock eigen function (orbital) with index `IOSA` is calculated on the same mesh as defined by `MESH` and limits defined by `LIMITS`. The values of the orbital is saved to the file `ORBITAL.dat`.

4.2.47 IOSA

Type: INTEGER (Default = (`Ne`-MOD(`Ne`,2))/2 +MOD(`Ne`,2), THE HOMO-orbital)
The index of the orbital which corresponding charge density being saved to the file `ORBITAL.dat` if `HFORBWRITE = .TRUE.`

4.2.48 AORBS

Type: INTEGER (Default = 0)
If `AORBS` $\neq 0$, this corresponds to the index of the orbital which is being saved to the file `ARBORB.xsf`. If `AORBS` > 0 the spin-up orbital is saved and if `AORBS` < 0 the spin-down orbital is saved. Assuming that the *xcrysden* software package have been installed on your

machine, you only have to give the following command on the command line in order to plot the orbital iso surface:

```
V.32/SERIALVERSION> xcrysden --xsf AORBS.xsf
```

For more details on how to obtain the iso-surface see the section describing the `WHOMOLUMO` input flag.

4.2.49 IORBNR

Type: INTEGER (Default = 0)
If `IORBNR`$\neq 0$ then a orthogonality constrained density functional calculation for electronic excited states will be performed according to [J. Phys. Chem. A **117**, 7378-7392, (2013)]. The Index of the orbital being excited and treated as a hole is defied by `IORBNR = 0`. If `IORBNR = 0`, then no excitation will be performed. If `IORBNR` > 0, then the spin-up orbital with index |`IORBNR`| will be treated as a hole, if `IORBNR` < 0 then the spin-down orbital with index |`IORBNR`| will be treated as a hole.

4.2.50 WHOMOLUMO

Type: Logical (Default = .False.)
Flag specifying weather or not to save the charge density and the orbital-values of the highest occupied molecular orbital (homo) and the lowest un-occupied molecular orbital (lumo) to file or not. If `WHOMOLUMO =.TRUE.` the the charge densities are saved to the files `HOMODENS.dat` and `LUMODENS.dat` and the orbital values are saved to `HOMO.xsf` and `LUMO.xsf`. The output format for the `HOMODENS.dat` is the same as for the `CHARGEDENS.dat`-file, and the output format of the `HOMO.xsf` and `LUMO.xsf` files are that of the *xrysden* software package. Assuming that the *xcrysden* software package have been installed on your machine, you only have to give the following command on the command line in order to plot the HOMO iso surface:

```
V.32/SERIALVERSION> xcrysden --xsf HOMO.xsf
```

after which you go to the `Tools` drag-down menu of the *xcrysden* program and select the "iso-surface" alternative. Then choose the `Data Grid` alternative, click ok. Finally you fill in your `Isovalue:` (usually 0.1 is a good default to start with), click the `Render +/- isovalue` and click on the `Submit` button in the lower right corner of the pop-up menu, and you will have created a isosurface of your HOMO orbital.

4.2.51 APPROXEE

Type: Logical (Default = .TRUE.)
if `APPROXEE=.TRUE.` then elements of the four-electron tensor, $(i,j|k,l)$, for which the following condition is satisfied:

$$\sqrt{(i,j|i,j)(k,l|k,l)}P_{max} < \texttt{EETOL} \tag{4.15}$$

$$P_{max} = Max(2|P_{ij}|, 2|P_{kl}|, |P_{ik}|, |P_{jk}|, |P_{il}|, |P_{jl}|) \tag{4.16}$$

will not be calculated and approximately set to zero. This is to enable a more effective alternative to calculating the electron repulsion tensor $(i,j|k,l)$ between orbitals that are centered far from each other. If `APPROXEE=.FALSE.` all elements of the tensor will be calculated.

4.2.52 EETOL

Type: Double Precision (Default = 1.0E-10)
The threshold used in the approximation of the four-electron tensor, $(i,j|k,l)$. See above definition of `APPROXEE` for more details.

4.2.53 BETA

Type: Double Precision. (Default = 1.0)
Parameter used for updating the total energy estimate, E_R, used in quantum Monte Carlo calculations. Whenever the time step number, I, satisfies $MOD(I,\texttt{NRECALC}) \neq 0$, the energy estimate is updated according to

$$E_R(I) = \langle E_R \rangle_{I-1} + \frac{\texttt{BETA}}{\tau_{eff}\texttt{NRECALC}} log\Big(\frac{\texttt{NREPLICAS}}{W_I}\Big) \tag{4.17}$$

here, I, is the time step number, W_I = Number of replicas of the current time-step, and `NRECALC` = Integer number set by the user (Default = $Int(1/\texttt{TIMESTEP})$) which defines the frequency in which the energy update instead is done by following prescription

$$E_R(I) = \langle E_L \rangle_{I-1} + \frac{1}{\tau_{eff}\texttt{NRECALC}} log\Big(\frac{\texttt{NREPLICAS}}{W_I}\Big) \tag{4.18}$$

i.e whenever $MOD(I,\texttt{NRECALC}) = 0$. Here

$$\langle E_x \rangle_n = \frac{1}{n}\sum_{j=1}^{n} E_x(j), \quad x = L,R, \quad \tau_{eff} = \texttt{TIMESTEP}\frac{N_a(I)}{W_I} \tag{4.19}$$

Here $N_a(I)$ is the number of walkers accepted to move. See for instance Umrigar, *et al* [10].

4.2.54 NREPLICAS

Type: Integer. (Default = 2000)
Total number of initial random walkers/replicas used in the Diffusion Monte Carlo calculation.

4.2.55 SAMPLERATE

Type: Integer. (Default = 10)
First use of this parameter is the sample rate employed in the variational Monte Carlo calculation used to calculate the initial distribution of random walkers. Since the Metropolis algorithm is used for the generation of this initial distribution, only random walkers separated by a number of `SAMPLERATE`-Metropolis moves are used for the initial distribution. This is to avoid that the walkers are correlated [9].

Second use of this parameter is to specify how often uquantchem is to save the file `MOLDYNRESTART.dat`. This file contains the information necessary for a continuation of a Molecular Dynamics Calculation. The file will be saved every `SAMPLERATE`:th time step.

Third use of this parameter is to specify the sampling rate of the atomic positions used to create the file `MOLDYNMOVIE.xsf`, if the flag `MOVIE` is set to .TRUE. . The atomic positions of `SAMPLERATE`:th time step will be used as a movie frame and saved to the `MOLDYNMOVIE.xsf` -file. To create at gif-movie you need to use the program xcrysden in the following way: `>>xcrysden --xsf MOLDYNMOVIE.xsf`

4.2.56 NPERSIST

Type: Integer. (Default = 50)
NPERSIST is the number of generations a random walker in the diffusion quantum Monte Carlo algorithm is permitted to stay in the same position. If a walker stays more than NPERSIST generations in one place the acceptance probability is increased by a factor of

$$1.1^{(N_g - \texttt{NPERSIST})}. \tag{4.20}$$

Here N_g is the number of generations (time-steps) a walker has been rejected to move. This is to avoid a population catastrophe. For more details see Umrigar *et al* [10].

4.2.57 NRECALC

Type: Integer. (Default = Int(1/`TIMESTEP`))
Integer number deciding how often the mean total energy estimate, $\langle E_R \rangle$, should be updated so that the number of walkers is kept close to the initial number of walkers `NREPLICAS`. See the the description of the related parameter `BETA` for a more detailed description.

4.2.58 REDISTRIBUTIONFREQ (Only used in the MPI-version)

Type: Integer. (Default = 0)
The frequency in which the random walkers in a diffusion Monte Carlo calculation are redistributed evenly over the MPI-threads (processors). For example `REDISTRIBUTIONFREQ`=3 results in a redistribution of walkers every third time step. If `REDISTRIBUTIONFREQ`=0 then the walkers are redistributed only if there is a threat of all walkers being killed on one computational node (thread/processor) or if there is a risk of overpopulation of walkers at one node. Observe that in order to enable a restart of a DQMC calculation one has to set `REDISTRIBUTIONFREQ`>0. This will force uquantchem to save all the information needed to continue a DQMC calculation to the file `DQMCRESTART.dat` every `REDISTRIBUTIONFREQ`:th time-step. If the file `DQMCRESTART.dat` exists uquantchem will continue the DQMC calculation from the time-step at which the file `DQMCRESTART.dat` was latest updated.

4.2.59 TEND

Type: Double Precision. (Default=10.0)
Specifies the run time of the Diffusion Quantum Monte Carlo calculation, aMolecular Dynamics Calculation or a TDDFT calculation.

4.2.60 TSTART

Type: Double Precision. (Default=TIMESTEP)
Specifies at which time one will start calculating the mean value of the local energy, E_L, i.e

$$\langle E_L \rangle = \frac{\texttt{TIMESTEP}}{\texttt{TEND} - \texttt{TSTART}} \sum_{t=\texttt{TSTART}}^{\texttt{TEND}} E_L(t) \tag{4.21}$$

4.2.61 TIMESTEP

Type: Double Precision. (Default=0.0025)
Specifies the time-step of the Diffusion Quantum Monte Carlo calculation, a Molecular Dynamics Calculation or a TDDFT calculation.

4.2.62 CUTTOFFFACTOR

Type: Double Precision. (Default=1.0)
Specifies the cut-off for the local energy, E_L, in a QDMC calculation. Is used to discard pathological configurations. If the the positions of a walker generation, I, generates a local energy, $E_L(I)$, such that

$$\frac{E_R - E_L(I)}{|E_R|} > CUTTOFFFACTOR \tag{4.22}$$

4.2.63 CUSPCORR

Type: LOGICAL. (Default= .TRUE.)
If CUSPCORR=.TRUE. then the basis functions are corrected so that they have the correct nuclear cusp behavior close to the nuclei. This correction is done in the spirit of S. Manten *et al* [11]. This correction is only used for quantum Monte Carlo calculations.

4.2.64 rc

Type: Double Precision. (Default=0.10)
If nuclear cusp correction is used the basis functions are corrected at distances $\leq$ rc from the nuclei at which they are centered.

4.2.65 CORRALCUSP

Type: LOGICAL. (Default= .TRUE.)
If true all basis functions will be cusp corrected. If false only basis functions constructed from contractions of more than 1 primitive gaussiam will be cusp corrected.

4.2.66 BJASTROW

Type: Double Precision. (Default=1.0)
Parameter used in the Jastrow factor, $\mathcal{J}$, containing the explicit electron correlation in the trial function

$$\Psi_T = D_\uparrow D_\downarrow \mathcal{J}. \tag{4.23}$$

Where

$$\mathcal{J} = exp(\sum_{i<j} \frac{\delta \cdot \texttt{BJASTROW} \cdot r_{ij}}{(1 + \texttt{CJASTROW} \cdot r_{ij})}, \tag{4.24}$$

$r_{ij} = |r_i - r_j|$ is the distance between electron i and j, $D_\uparrow$ and $D_\downarrow$ are the slater determinants created from spin up respectively spin down orbitals. The orbitals are constructed from the URHF self consistent solution. Here $\delta = 0.25$ if the spin of the electrons i and j are identical otherwise if the spins are opposite $\delta = 0.5$. Observe that `BJASTROW` should be kept equal to one since this help avoid infinities in the local energy, E_L, when $r_i = r_j$. Thus the only variational parameter that can be used in variational Monte Carlo Calculations (`CORRLEVEL = VMC`) is `CJASTROW`.

4.2.67 CJASTROW

Type: Double Precision. (Default=0.5)
See above description of `BJASTROW`.

4.2.68 NVMC

Type: INTEGER. (Default = 1000000)
Number of Metropolis configurations used in the variational Monte Carlo calculation. Note that there are `SAMPLERATE` number of metropolis moves in between every Metropolis configurations used. Thus there are a total of `NVMC` $\cdot$ `SAMPLERATE` Metropolis moves performed.

4.2.69 LEXCSP

Type: LOGICAL. (Default= .FALSE.)
If true the slater determinants used in the configuration interaction calculation (`CORRLEVEL = CISD`) will be limited to the slater determinants created by exchanging the `NEEXC` highest occupied Hartree-Fock orbitals with unoccupied Hartree-Fock orbitals that have energy eigen values $<$ `ENEXCM`.

4.2.70 ENEXCM

Type: Double Precision. (Default= None)
See description of `LEXCSP`.

4.2.71 NEEXC

Type: INTEGER. (Default = `Ne`, if `LEXCSP=.FALSE.`)
See description of `LEXCSP`.

4.2.72 SPINCONSERVE

Type: LOGICAL. (Default= .TRUE.)
If `SPINCONSERVE=.TRUE.` then the total spin of the system is conserved for all excitations used to create the basis set of slater determinants used in the configuration interaction calculations (CISD) or MP2 calculations. if `SPINCONSERVE=.FALSE.` then spin-flips will be

allowed and the spin of the system is not conserved when creating the CISD basis set or calculating the MP2 correctons.

4.2.73 RESTRICT

Type: LOGICAL. (Default= .FALSE.)
If true then the CISD and MP2 calculations will be based un slater determinants created from a restricted Hartree-Fock (RHF) calculation.

4.3 The output files

4.3.1 The CHARGEDENS.dat-file

In this file the charge density calculated with Hartree-Fock or Diffusion Quantum Monte Carlo is saved. The file consists of four columns. The first three columns (counting from left to right) contain integer numbers corresponding to the coordinate mesh indexes I, J, K described together with the input parameters WRITEDENS and MESH. The fourth column contains the calculated values of the charge density $\rho(x(I), y(J), z(K)) = \rho(x, y, x)$.

4.3.2 The RHFEIGENVALUES.dat-file

In this file the energy eigenvalues of the restricted Hartree-Fock calculation (RHF) are stored. First column eigenvalue index and second column eigenvalues. of the spin down orbitals.

4.3.3 The UHFEIGENVALUES.dat-file

In this file the energy eigenvalues of the unrestricted Hartree-Fock calculation (URHF) are stored. First column eigenvalue index, second column eigenvalue of spin up orbitals and third column eigenvalues of the spin down orbitals.

4.3.4 The ORBITAL.dat-file

In this file the Hartree-Fock eigen function, $\Psi_{\texttt{IOSA}}$, calculated on the mesh specified by the input parameters WRITEDENS and MESH is saved. The file consists of Five columns. The first three columns (counting from left to right) contain integer numbers corresponding to the coordinate mesh indexes I, J, K described together with the input parameters WRITEDENS and MESH. The fourth column contains the calculated values of the Hartree-Fock eigen function, $\Psi_{\texttt{IOSA}\uparrow}(x(I), y(J), z(K)) = \Psi_{\texttt{IOSA}\uparrow}(x, y, x)$, and the fifth column contains the calculated values of the Hartree-Fock eigen function, $\Psi_{\texttt{IOSA}\downarrow}(x(I), y(J), z(K)) = \Psi_{\texttt{IOSA}\downarrow}(x, y, x)$.

4.3.5 The ABSSPECTRUM.dat-file

Is produced if DOABSSPECTRUM = .TRUE. or if DOABSSPECTRUM is not explicitly set by the user and INT(TEND/TIMESTEP) $<$ 10000 then the ABSSPECTRUM.dat-file is produced which contains the absorption spectrum, i.e, the time-Fourier transform of the dipole moment,

$$\mu_p(t) = \sum_{atom} (\mathbf{R}_{atom} \cdot \hat{p}) Z_{atom} - Tr[P(t) d_p]. \tag{4.25}$$

Here $\hat{p}$ is the polarization direction of the electric field, P, the density matrix and, d_p, the dipole tensor in the polarization direction.

4.3.6 The `TDFTOUT.dat`-file

Contains information from the TDDFT/TDHF calculation. First column = time-step index, second column = time, third column = dipole moment, fourth column = expectation value of the effective hamiltonian/Fockian, fifth column = total electron charge, sixth column = electric field.

4.3.7 The `OCCUPATIONUP.dat`-file

First column contains the time and the second column contains the orbital occupation numbers, $n_i(t)$ of the spin-up orbitals,

$$n_i^{\uparrow}(t) = C_i^{\uparrow\dagger}(0)P(t)C_i^{\uparrow}(0) \tag{4.26}$$

projected out from the density matrix, $P(t)$ of a TDDFT/TDHF calculation.

4.3.8 The `OCCUPATIONDOWN.dat`-file

First column contains the time and the second column contains the orbital occupation numbers, $n_i(t)$ of the spin-down orbitals,

$$n_i^{\downarrow}(t) = C_i^{\downarrow\dagger}(0)P(t)C_i^{\downarrow}(0) \tag{4.27}$$

projected out from the density matrix, $P(t)$ of a TDDFT/TDHF calculation.

4.3.9 The `DENSEXITED_0.xsf, . . . ,DENSEXITED_23.xsf`-files

If a TDDFT or a TDHF calculation is done and `WRITEDENS = .T.` and `DIFFDENS=.FALSE.` these 24 files containing the charge density of the excited, $\rho^{ex}(\mathbf{r},t)$, states calculated from the occupation numbers, $n_i^{\uparrow}(t) = C_i^{\uparrow\dagger}(0)P(t)C_i^{\uparrow}(0)$ and $n_i^{\downarrow}(t) = C_i^{\downarrow\dagger}(0)P(t)C_i^{\downarrow}(0)$, are produced. The excited states charge density is given by

$$\begin{aligned}\rho^{ex}(\mathbf{r},t_j) = &\sum_{i=1}^{N_B}\sum_{\mu,\nu=1}^{N_B} n_i^{\uparrow}(t_j)C_{\mu i}^{\uparrow\dagger}(0)C_{\nu i}^{\uparrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r}) + \\ &\sum_{i=1}^{N_B}\sum_{\mu,\nu=1}^{N_B} n_i^{\downarrow}(t_j)C_{\mu i}^{\downarrow\dagger}(0)C_{\nu i}^{\downarrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\end{aligned} \tag{4.28}$$

where, t_j, is given by

$$t_j = \frac{j}{24}\texttt{TEND} \quad , j = 0, \ldots, 23 \tag{4.29}$$

If however `DIFFDENS=.TRUE.` then the differentiated charge density defined by

$$\begin{aligned}\Delta\rho^{ex}(\mathbf{r},t_j) = &\sum_{i=1}^{N_e^{\uparrow}}\sum_{\mu,\nu=1}^{N_B}(1-n_i^{\uparrow})(t_j)C_{\mu i}^{\uparrow\dagger}(0)C_{\nu i}^{\uparrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &-\sum_{i=N_e^{\uparrow}+1}^{N_B}\sum_{\mu,\nu=1}^{N_B}n_i^{\uparrow}(t_j)C_{\mu i}^{\uparrow\dagger}(0)C_{\nu i}^{\uparrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &+\sum_{i=1}^{N_e^{\downarrow}}\sum_{\mu,\nu=1}^{N_B}(1-n_i^{\downarrow})(t_j)C_{\mu i}^{\downarrow\dagger}(0)C_{\nu i}^{\downarrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\\ &-\sum_{i=N_e^{\downarrow}+1}^{N_B}\sum_{\mu,\nu=1}^{N_B}n_i^{\downarrow}(t_j)C_{\mu i}^{\downarrow\dagger}(0)C_{\nu i}^{\downarrow}(0)\phi_{\mu}^{\dagger}(\mathbf{r})\phi_{\nu}(\mathbf{r})\end{aligned} \tag{4.30}$$

is written to the files. The density (or differentiated charge density) is calculated on a mesh defined by the input-parameters: `MESH` and `LIMITS`, and saved in the `.xsf`-format so that they can be utilized by the XcrySden program.

4.3.10 The `HOMODENS.dat`-file

In this file the charge density corresponding to the highest occupied molecular orbital (HOMO) calculated with Hartree-Fock is saved . The file consists of four columns. The first three columns (counting from left to right) contain integer numbers corresponding to the coordinate mesh indexes I, J, K described together with the input parameters `WRITEDENS` and `MESH`. The fourth column contains the calculated values of the charge density $\rho_{HOMO}(x(I), y(J), z(K)) = \rho_{HOMO}(x, y, x)$

4.3.11 The `LUMODENS.dat`-file

In this file the charge density corresponding to the lowest unoccupied molecular orbital (LUMO) calculated with Hartree-Fock is saved . The file consists of four columns. The first three columns (counting from left to right) contain integer numbers corresponding to the coordinate mesh indexes I, J, K described together with the input parameters `WRITEDENS` and `MESH`. The fourth column contains the calculated values of the charge density $\rho_{LUMO}(x(I), y(J), z(K)) = \rho_{LUMO}(x, y, x)$

4.3.12 The `ENERGYDQMC.dat`-file

In this file the time step index (first column) the mean value of the local energy, $\langle E_L \rangle$, (second column), the estimate of the ground state energy, E_R, (third column) and the number of random walkers (fourth column) of the Diffusion Quantum Monte Carlo (DQMC) calculation are printed.

Don't forget to check the resulting DQMC energy by running the utility program `dqmc_check.pl`. This program (located at `V.32/UTILITYPROGRAMS`) takes the `ENERGYDQMC.dat`-file as input and checks the correlation between walker populations separated by `$m` time-steps (`$m` = 5 default, and is set by editing the perl script `dqmc_check.pl`). The `dqmc_check.pl` script will

also calculate the mean value of the local energies E_L separated by `$m` time-steps. Remember that uquantchem has no offset when sampling the local energy, E_L, i.e `$m` = 1. Thus the mean value for the local energy $\langle E_L \rangle$, given as output directly from uquantchem, might very well have been calculated over correlated populations.

4.3.13 The `DQMCRESTART.dat` -file

In this file all the information needed to restart a DQMC calculation is stored. If this file is present in the running directory of uquantchem, then the DQMC calculation will be continued from the time-step where the `DQMCRESTART.dat` -file was most recently updated. Observe that the condition `REDISTRIBUTIONFREQ` > 0 must be satisfied in order for the `DQMCRESTART.dat` to be created. The `DQMCRESTART.dat` -file is updated every `REDISTRIBUTIONFREQ`:th time-step. The possibility to continue a DQMC calculation only exists for the MPI-version of the code.

4.3.14 The `MOLDYNENERGY.dat`-file

The energies and temperature are saved to the file: `MOLDYNENERGY.dat` ,first column = time step index, second column = total energy, third column = kinetic energy, fourth column = potential energy, fifth column = temperature and sixth column number of self consistent cycles performed.

4.3.15 The `MOLDYNRESTART.dat`-file

In this file contains the information needed to continue a Molecular Dynamics Calculation. The file contains the following information: 1-st line of file contains the time step index (integer), the following `NATOMS` lines contain three columns with the atomic positions corresponding to the time-step index of the first line, the `NATOMS` lines following the atomic positions contain three columns with the velocities of the atoms corresponding to the time-step index of the first line, finally the last `NATOMS` lines contain three colums with the interatomic forces of the atoms corresponding to the time-step index of the first line. IF THE FILE `MOLDYNRESTART.dat` EXISTS IN THE RUNNING DIRECTORY OF UQUANTCHEM THE CALCULATION WILL BE AUTOMATICALLY CONTINUED. TO RESTART A MOLECULAR DYNAMICS CALCULATION FROM SCRATCH BE SURE TO REMOVE THIS FILE FROM THE RUNNING DIRECTORY.

4.3.16 The `MOLDYNMOVIE.xsf`-file

This file contains atomic positions, in [Å], sampled from a Molecular Dynamics Calculation and can be used by the xcrysden program to create a gif-movie of the motion of the molecule. To do this just: `>>xcrysden --xsf MOLDYNMOVIE.xsf`, then once xcrysden starts the rest is pretty self explanatory. The resulting movie is most easily viewed by any web browser.

4.3.17 The `ATOMPOSITIONS.dat`-file

This file contains the latest update of the atomic positions during and after a structural relaxation calculation.

4.3.18 The `HOMO.xsf`-file and the `LUMO.xsf`-file

Contain the HOMO and LUMO orbital values printed on a 3D mesh, and can be used by the xcrysden program to plot iso surfaces of the respective orbitals.

Chapter 5

Example Calculations

In the sub-directories `V.32/EXAMPLE_INPUT_OUTPUT/` one can find the results of different uquantchem calculations.

RUN1: contains the output and input of a URHF calculation of a water molecule using a $6-31G^{**}$ basis set.

RUN2: contains the output and input of a MP2 calculation of a He atom using a cc-pVQZ basis set.

RUN3: contains the output and input of a CISD calculation of a H_2 molecule atom using a cc-pVDZ basis set.

RUN4: contains the output and input of a VMC calculation of a H_2 molecule atom using a $6-31G^{**}$ basis set.

RUN5: contains the output and input of a DQMC calculation of a H_2 molecule atom using a $6-31G^{**}$ basis set.

RUN6: contains the output and input of a DQMC calculation of a He atom using a cc-pVTZ basis set.

RUN7: contains the output and input of a RHF charge density calculation for a H_2O molecule using a $6-31G^{**}$ basis set.

RUN8: contains the output and input of a RHF of a Cr atom using a $6-31^{**}$ basis set. Here the one of the d-orbitals is plotted in the z=0 plane.

RUN9: contains the output and input of a DQMC calculation of a He atom using a cc-pVTZ basis set. Here the calculation is done with a slightly smaller time step than in RUN6. Also the charge density has been accumulated from the walker configurations at the different time steps.

RUN10: contains the output and input of a VMC calculation of a H_2O using a $6-31G^{**}$ basis set. Also, the charge density has been calculated from the generated Metropolis configurations.

RUN11-20 I have forgotten the exact type of calculations performed, but I think most of them are DQMC or VMC calculations.

RUN21: contains the output and input of a structure relaxation calculation for CH_4 using URHF and a 3-21G basis set .

RUN22,26: contains the output and input of a structure relaxation calculation for H_2O using URHF and a $6-31G^{**}$ basis set .

RUN27: contains the output and input of a structure relaxation calculation for H_2 using URHF and a cc-pVDZ basis set .

RUN31: contains the output and input of a structure relaxation calculation for H_2O using the PBE functional and a cc-pVTZ basis set .

RUN31: contains the output and input of a structure relaxation calculation for H_2O using the B3LYP functional and a STO-3G basis set .

RUN33X, RUN33Y, RUN33Z: contains the output and input three TDDFT calculations for H_2O using the PBE functional and the 6-31G basis set . See figure 5.1 for the resulting absorption spectrum.

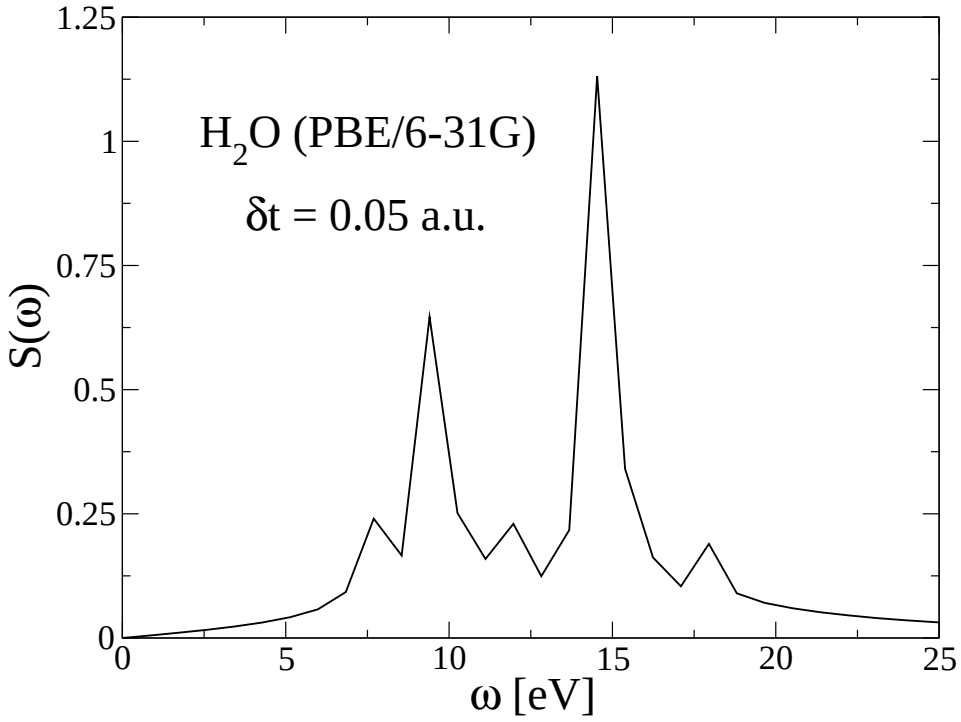


Figure 5.1: Absorption spectrum calculated with uquantchem's TDDFT implementation. Here 3 Dirac pulses where used to produce the spectrum. A 6-31G basis set together with the PBE functional was used.The corresponding input files are INPUTFILE.RUN33X, INPUTFILE.RUN33Y and INPUTFILE.RUN33Z. The time step used was 0.05 a.u.

Chapter 6

Utility Matlab programs for plotting

In the sub-directory `V.32/UTILITYPROGRAMS` there are a couple of simple matlab scripts that can help you visualize charge-densities and atomic orbitals calculated with UQUANTCHEM.

Chapter 7

For the developer

In this chapter the different subdirectories of the directory `V.32/` and their respective content will be briefly discussed.

7.1 File-map

The following subdirectories are included in the `V.32` package:

7.1.1 `BASIS`

This directory contains the files of all the different gaussian basis-sets that accompany this release of uquantchem.

7.1.2 `BLAS`

This directory contains the basic linear algebra sub-programs (BLAS) attached to this release of uquantchem to enable the compilation of the code in the event that there is no native BLAS on the machine at which uquantchem is to be installed.

7.1.3 `lapack-3.4.0`

This directory contains the linear algebra package (LAPACK) attached to this release of uquantchem to enable the compilation of the code in the event that there is no native LAPACK on the machine at which uquantchem is to be installed.

7.1.4 `UTILITYPROGRAMS`

This directory contains a number of utility scripts written in perl or to be used with matlab. Here follows a list of the included programs and a short description of how the are utilized:

`velautocorr.pl`
This script takes the uquantchem movie-file: `MOLDYNMOVIE.xsf` and calculates the velocity-auto-correlation function. The result is saved in the file: `VELAUTO.dat` and performs the temporal fouriertransform of the very same function and saves it to the files `SPECTRUM_THz.dat`

and SPECTRUM_cm.dat

velautocorr2.pl
A slightly modified version of velautocorr.pl.

pdbtoau.pl
For converting the atomic coordinates in pdb-files to a coordinate file suitable for the uquantchem code.

dqmc_check.pl
Calculates the mean local energy E_L by using every the local energy at every m:th (integer specified by user inside of script) time-step from the out-put file of a dqmc calculation ENERGYDQMC.dat. It also calculates the correlation function.

distance.pl
This script takes the uquantchem movie-file: MOLDYNMOVIE.xsf and calculates the interatomic distance between atom A and B as a function of time.

isodens.m
This matlab script uses the out-put file CHARGEDENS.dat to calculate and plot an iso-density surface of the charge density.

isohomolumo.m
This matlab script uses the out-put files LUMODENS.dat and HOMODENS.dat to calculate and plot an iso-density surface corresponding to the LUMO and HOMO orbitals.

7.1.5 TESTS

This subdirectory contains the subdirectories: RUN1, RUN2, ... ,RUN7 which in turn contains input and output files of seven different test calculations performed with the openmp version of the uquantchem code. A more detailed specification of these test calculations can be found in the README-file.

7.1.6 README

This file contains basic information on how to compile the uquantchem code and information on the different test calculations located in the subdirectory TESTS.

7.1.7 manual.pdf

This manual that you are now reading.

7.1.8 SERIALVERSION

This subdirectory contains all the fortran source files (*.f90) of the serial version of the uquantchem code together with different Makefiles. Here follows a complete list of the different fortran source files and a short description:

`uquantchem.f90`
Main program.

`datatypemodule.f90`
Module in which the different datatypes specific to the uquantchem code have been defined.

`exchcorrmodule.f90`
Module in which the different exchange correlation functionals have been defined in terms of subroutines and functions.

`random.f90`
Module for different random number generators to be used in the diffusion quantum monte Carlo calculations.

`Afunc.f90`
This is the function A_lri(l1,l2,Ax,Bx,Cx,gamma) defined at the top of page 245 in the Cook Book.[5]

`RHF.f90`
Subroutine for performing restricted Hartree Fock calculations.

`Fnurec.f90`
This subroutine calculates the values F_nu(X) (see Cook Book p.280) [5] for nu=0,1,2,...,NUMAX, Through the recursion formula given on the bottom of p. 280 of the Cook Book.

`Fnu.f90`
The function F_nu(x) described in the Cook Book [5] on pages 244 and 280, is here calculated from the Kummer confluent hypergeometric function. See WIRE's Computational Molecular Science, 2012, **2**: 290-303

`CISD.f90`
Subroutine for the construction and diagonalization of the configuration interaction Hamiltonian constructed from singles and doubles excitations.

`getK.f90`
Subroutine calculating the the exchange matrix K_{il} by contracting the electron-electron integrals $(ij|kl)$ with the density matrix P_{jk}.

`getKv.f90`
Vectorized version of `getK.f90`.

`getJ.f90`
Subroutine calculating the the Hartree matrix J_{ij} by contracting the electron-electron integrals $(ij|kl)$ with the density matrix P_{kl}.

`getJv.f90`
Vectorized version of `getJ.f90`.

`gammaf.f90`
calculates the value of the GAMMA-function $\Gamma(I+0.5)$ gfortan has the gamma function as an intrinsic function whereas ifort on osx does not seem to have it.

`fj.f90`
The function f_j(l,m,a,b) on page 237 in the Cook book [5].

`fac.f90`
simple factorial function: n!

`gto.f90`
Calculates the value at argument for a spherically symmetric gto basis function consisting of contracted primitive gaussians.

`gtop.f90`
Calculates the radial derivative, at argument r, of a spherically symmetric gto basis function consisting of contracted primitive gaussians.

`gtopp.f90`
Calculates the second radial derivative, at argument r, of a spherically symmetric gto basis function consisting of contracted primitive gaussians.

`gtoppp.f90`
Calculates the third radial derivative, at argument r, of a spherically symmetric gto basis function consisting of contracted primitive gaussians.

`eeints.f90`
Subroutine that calculates the electron-electron integrals $(ij|kl)$ by Rys quadrature.

`diaghHF.f90`
Subroutine that calculates the eigenvalues, ϵ, and eigenvectors, C, to the eigenvalue problem: $FC = \epsilon SC$. Here, F, corresponds to the Fockian matrix and S the overlap matrix.

`diagh.f90`
Subroutine that calculates the eigenvalues, ϵ, and eigenvectors, C, to the eigenvalue problem: $HC = \epsilon C$. Here, H, typically corresponds to the Hamiltonian matrix.

`dfac.f90`
Function that calculates the double factorial, n!!.

`checkinputfile.f90`
This subroutine checks wheater or not the file 'INPUTFILE' exists and if so, how many lines, NLINES, it contains before the entry of the number of atoms, NATOMS.

`chargedenssave.f90`
This subroutine calculates and saves the charge-density on disk.

`exciteornot.f90`
Function used to decide which excitations to use when performing a CISD calculation.

`hforbitalsave.f90`
Subroutine that calculates the values of a Hartree-Fock or Kohn-sham orbital on a mesh and saves the result to disk.

`binomfac.f90`
calculates the binomial factor $\binom{N}{M} = \frac{N!}{M!(N-M)!}$

`basfunkval.f90`
Calculates the value of a real basis-function, ψ_i, at the spatial point $r \in \mathbf{R}^3$. I.e, $\psi_i(r)$.

`URHF.f90`
Subroutine for performing unrestricted Hartree Fock calculations.

`primeeintegral.f90`
Function that calculates the electron repulsion integrals $(\mu\nu|\kappa\lambda)$ over the primitive gaussian functions $\phi_\mu, \phi_\nu, \phi_\kappa, \phi_\lambda$, i,e:

$$(\mu\nu|\kappa\lambda) = \int d^3\mathbf{r} \int d^3\mathbf{r}' \frac{\phi_\mu(\mathbf{r}), \phi_\nu(\mathbf{r}), \phi_\kappa(\mathbf{r}'), \phi_\lambda(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}$$

`potential.f90`
Subroutine that calculates the potential matrix, V_{ij}, and the nuclear gradients of the potential matrix, $\nabla_{\mathbf{R}} V_{ij}$. Where V_{ij} is defined by

$$V_{ij} = -\sum_k \int d^3\mathbf{r} Z_k \frac{\psi_i(\mathbf{r})\psi_j(\mathbf{r})}{|\mathbf{r} - \mathbf{R}_k|}.$$

Here, $\mathbf{R}_k$ are the positions of the nuclei in the molecule and Z_k are the respective atomic numbers.

`overlap.f90`
Subroutine that calculates the overlap matrix, S_{ij}, and the nuclear gradients of the overlap matrix, $\nabla_{\mathbf{R}} S_{ij}$. Where S_{ij} is defined by

$$S_{ij} = \int d^3\mathbf{r} \psi_i(\mathbf{r})\psi_j(\mathbf{r}).$$

`oneint.f90`
This function calculates the Ix(ta) term on p. 974 in J. Comp. Chem. **11**, 972-977 (1990) which is used to calculate the two electron integrals through Rys-quadrature.

`normalize.f90`
This subroutine calculates the normalization constants for all of the basis functions.

`ROOTWEIGHT5.f90`
Subroutine that calculates the roots and weights of 5:th order Rys polynomials. The calculation is done by polynomial fits of pree-calculated roots and weights. The fits are taken from the GAMES-package (Same as in Pyquante), a very fast process.

`leastsq.f90`
Performs a least square fit of the function $f(r) = Ae^{-\lambda r} + C$ to the N data points $r(i) = \frac{rc}{N-1} * (i-1), \quad y(i) = GTO(r(i)), \quad i = 1, 2, \ldots, N$

`makedens.f90`
Subroutine that calculates the density matrix, $P_{\mu\nu}$, from the eigen-vectors of the Fockian/Khon-Sham Hamiltonian, C_i, i.e

$$P_{\mu\nu} = \sum_{i=1}^{N_{occ}} C_{\mu i} C_{\nu i}^{\dagger}$$

`kinetic.f90`
Subroutine that calculates the Kinetic energy matrix, T_{ij}, and the nuclear gradients of the kinetic energy matrix, $\nabla_{\mathbf{R}} T_{ij}$. Where S_{ij} is defined by

$$T_{ij} = -\frac{1}{2} \int d^3\mathbf{r} \psi_i(\mathbf{r}) \nabla^2 \psi_j(\mathbf{r}).$$

`ijkl.f90`
Function that contacts the 4-component index i,j,k,l, used to label the different electron-electron repulsion integrals $(ij|kl)$, into one single index.

`homolumosave.f90`
Subroutine that calculates the densities and orbital values of the HOMO and LUMO orbitals on a mesh and writes the result to the files `HOMODENS.dat`, `LUMODENS.dat`, `HOMO.xsf` and `LUMO.xsf`.

`gettotalbasis.f90`
If the logical input variable COUNTER=.TRUE. , this subroutine calculates the size of the total basis set from the local basis sets centered at the different atoms If COUNTER=.FALSE. , this subroutine returns the total basis set.

`rysquad.f`
This subroutine calculates the roots and weights of a Rys polynomial of arbitrary order.

`rho.f90`
Function that calculates the charge density, ρ, from the basis functions and the density matrix P at one point in space.

`readin.f90`
Subroutine that reads the user specified data stored in the file `INPUTFILE`.

`ROOTWEIGHT4.f90`
Subroutine that calculates the roots and weights of 4:th order Rys polynomials. The calculation is done by polynomial fits of pree-calculated roots and weights. The fits are taken from the GAMES-package (Same as in Pyquante), a very fast process.

`interpol.f90`
This routine calculates the interpolating cubic polynomial linking together the second derivative of an sto-type basis function: $sto''(r) = A\lambda^2 e^{-\lambda r}$ at the point $r = r_c$, and a the second derivative of a gto-type basis function: $gto''(r)$ at $r = r_c + \delta r$ so that both the second and the third derivative of the polynomial matches that sto at $r = r_c$ and the gto at $r = r_c + \delta r$. This is done according to J. Chem. Phys. **115**, 5362 (2001).

`gradbasfunkval.f90`
Subroutine that calculates the gradient of a real basis-function, ϕ_μ, at the spatial point $\mathbf{r} \in \mathbf{R}^3$. I.e, the gradient = $\nabla\phi_\mu(\mathbf{r})$.

`laplacebasfunkval.f90`
Function that calculates the value of the laplacian of a real basis-function, ϕ_μ, at the spatial point $\mathbf{r} \in \mathbf{R}^3$. I.e, the laplacian = $\nabla^2\phi_\mu(\mathbf{r})$.

`hforbitalval.f90`
Function that calculates the value of a Hartree-Fock orbital or a Kohn-Sham orbital at the spatial point $\mathbf{r} \in \mathbf{R}^3$, given the eigen-vector C_i. I.e the value $\Psi_i(\mathbf{r})$.

`gradhforbitalval.f90`
Subroutine that calculates the gradient of a Hartree-Fock orbital or a Kohn-Sham orbital at the spatial point $\mathbf{r} \in \mathbf{R}^3$, given the eigen-vector C_i. I.e the gradient = $\nabla\Psi_i(\mathbf{r})$.

`laplacehforbitalval.f90`
Subroutine that calculates the laplacian of a Hartree-Fock orbital or a Kohn-Sham orbital at the spatial point $\mathbf{r} \in \mathbf{R}^3$, given the eigen-vector C_i. I.e the laplacian = $\nabla^2\Psi_i(\mathbf{r})$.

`det.f90`
This function calculates the determinant of a NxN matrix A, by first transforming it into upper diagonal form.

`stoexponent.f90`
Finds the exponent, λ, and coeficient, A, of a slater type orbital, $Ae^{-\lambda r}$ by fiting it to the gaussian basis function, $gto(r)$, at the inflexion point r_c, i.e at the point where $gto'(r_c) = min$. The exponent, λ, and coeficient, A are calculated from the continuity conditions:
(1) $gto(r_c) = Ae^{-\lambda r_c}$
(2) $gto'(r_c) = -A\lambda e^{-\lambda r_c}$
leading to $\lambda = -gto'(r_c)/gto(r_c)$, and $A = gto(r_c)e^{-gto'(r_c)r_c/gto(r_c)}$

`slaterdet.f90`
This fuction calculates the slater determiant from:
(a) N = Total Number of electrons,
(b) BAS = Hartree Fock basis
(c) C = Fockian eigenvectors,
(d) r = positions the electrons 3N-long array
(e) UP = .TRUE. Spin up slater determinant is calculated, otherwise spin-down slater determinant is to be calculated.
(f) NS = slater determinant dimension.

`gradslaterdet.f90`
This subroutine calculates the gradient of a slater determiant, with respect to the electron coordinate I, from:
(a) N = Total Number of electrons,
(b) BAS = Hartree Fock basis
(c) C = Fockian eigenvectors,
(d) r = positions the electrons 3N-long array
(e) UP = .TRUE. if I = spin upp orbital, else UP = .FALSE.
(f) NS = Slaterdeterminant dimension

`laplaceslaterdet.f90`
This fuction calculates the laplacian of a slater determiant, with respect to the electron coordinate I, from:
(a) N = Total Number of electrons,
(b) BAS = Hartree Fock basis
(c) C = Fockian eigenvectors,
(d) r = positions the electrons 3N-long array
(e) NS = Slaterdeterminant dimension.
(f) UP = .TRUE. if I = spin upp orbital, else UP = .FALSE.

`jastrowup.f90`
This function calculates the Jastrow factor contribution from the spin-up electrons. N = total number of electrons r = coordinates of all electrons, r is a 3*N-long array b,c = jastrow parameters.

`jastrowdown.f90`
This function calculates the Jastrow factor contribution from the spin-down electrons. N = total number of electrons r = coordinates of all electrons, r is a 3*N-long array b,c = jastrow parameters.

`jastrowud.f90`
This function calculates the Jastrow factor contribution from the cross-terms between spin-up and spin-down electrons. N = total number of electrons r = coordinates of all electrons, r is a 3*N long-array b,c = jastrow parameters.

`jastrow.f90`
This function calculates the (Total) Jastrow factor. N = total number of electrons r = coordinates of all electrons, r is a 3*N-long array b,c = jastrow parameters.

`DIIS.f90`
This is the direct inversion in the iterative subspace method DIIS, by P. Puley in CHEM. Phys. Lett. **73**, 393 (1984). The routine is used to estimate the self-consistent density-matrix in the region where the density dependence of the energy can be approximated well up to second order. Also, see T. Halgaker, P. Jorgensen and J. Olsen *"Molecular Electronic Structure Theory"* p.460-463

`findclosestatom.f90`
Subroutine that finds the atomic nucleus located closest to a particular electron. The motivation for the use of this routine can be found in J. Chem. Phys. **99** , 2865-2890, (1993).

`gradjastrow.f90`
This subroutine calculates the gradient of the Jastrow factor, with respect to the electron coordinate I.
N = total number of electrons
r = coordinates of all electrons, r is a 3*N-long array
b,c = jastrow parameters.

`laplacejastrow.f90`
This function calculates the laplacian of the Jastrow factor, with respect to the electron coordinate I.
N = total number of electrons
r = coordinates of all electrons, r is a Nx3-array
b,c = jastrow parameters.

`guideforce.f90`
This subroutine calculates the guiding force used in the DQMC importance sampling.

`trialfnk.f90`
This function calculates the value of the trial function, used in the DQMC importance sampling, at a specific point in the $3N_e$ dimensional coordinate space of the electrons. (N_e=number of electrons)

`laplacetrialfnk.f90`
This function calculates the value of laplacian of the trial function, used in the DQMC importance sampling, at a specific point in the $3N_e$ dimensional coordinate space of the electrons. (N_e=number of electrons)

`EL.f90`
This function calculates the Local energy of the trial function, at a specific point in the $3N_e$ dimensional coordinate space of the electrons (N_e=number of electrons). Used in the DQMC importance sampling.

`dqmc.f90`
This routine performs the diffusion quantum Monte Carlo or the Variational Quantum Monte Carlo calculations. This is the routine that is at the top of the hierarchy/program-tree, i.e the control routine, of all the quantum Monte Carlo calculations.

`preparedifusion.f90`
Subroutine that performs a series of calculations in order to enable a diffusion step in the DQMC calculation. For details see J. Chem. Phys. **99** , 2865-2890, (1993), and the comments embedded in the source of this routine.

`ROOTWEIGHTMAX3.f90`
Subroutine that calculates the roots and weights of Rys polynomials of order ≤ 3. The calculation is done by polynomial fits of pree-calculated roots and weights. The fits are taken from the GAMES-package (Same as in Pyquante), a very fast process.

`readbasis.f90`
This subroutine reads in the expansion coefficients and exponents of the gaussian basis-set stored in the file `BASISFILE`.

`primpotential.f90`
This function calculates the potential energy integral given on the bottom of p. 244 in the Cook Book [5].

`primoverlap.f90`
This function calculates the overlap integral $S = S_x S_y S_z$ as given by the expressions on the pages 234-238 in the Cook book [5].

`primkinetic.f90`
This function returns the integral $\langle GTO_1|-\frac{1}{2}\nabla^2|GTO_2\rangle$ as given by the cook Book on pages 234-239 [5]. Here GTO_i are primitive gaussian functions.

`MP2.f90`
Subroutine that performs second order Møller-Plesset calculations.

`gradprimoverlap.f90`
This subroutine calculates the nuclear gradients of the overlap integral $S = S_x S_y S_z$ as given by the expressions on the pages 234-238 in the Cook book [5].

`gradprimkinetic.f90`
This subroutine calculates the nuclear gradients of the integral $\langle GTO_1| - \frac{1}{2}\nabla^2|GTO_2\rangle$ as given by the cook Book on pages 234-239 [5]. Here GTO_i are primitive gaussian functions.

`gradprimpotential.f90`
This subroutine calculates the nuclear gradients of the potential energy integral given on the bottom of p. 244 in the Cook Book [5].

`gradprimeeintegral.f90`
This subroutine calculates the nuclear gradients of the two-electron integral between primitive gaussians.

`forces.f90`
This subroutine calculates the interatomic forces on all atoms of the molecule.

`dAfunc.f90`
This function calculates the derivative of the function $A_{lri}(l_1, l_2, A_x, B_x, C_x, \gamma)$ (defined at the top of page 245 in the Cook Book [5]) with respect to C_x.

`relax.f90`
This subroutine calculates the atomic positions of the molecule that corresponds to the minimal energy.

`linesearchmin.f90`
This subroutine fits a number of NPOINTS energies and moves to 4:th order polynomial in the interval $[0, DR]$ and calculates the position of the minimum of this polynomial in the interval $[0, DR]$. This routine is used by `relax.f90` in order to find the move involved a steepest decent or a conjugate gradient calculation.

`massa.f90`
This function just give the atomic mass as output given the atomic number as input. The atomic masses have been taken from Aschcroft and Mermins *"Solid State Physics"*.

`moleculardynamics.f90`
Subroutine that performs molecular dynamics calculations.

`moleculardynamicssoft.f90`
Subroutine that performs molecular dynamics calculations, with a so called soft-start. See, the description of the input parameter `SOFTSTART`.

`invert.f90`
Subroutine that inverts a $N \times N$ real matrix.

`vxc.f90`
This subroutine calculates the exchange correlation potential.

`vxcalt.f90`
Subroutine not used for the moment.

`dvxc.f90`
Subroutine not used for the moment.

`ngbasfunkval.f90`
Calculates the value of the gradient of a real basis-function at the spatial point $\mathbf{r} \in \mathbf{R}^3$, with respect to the nuclear coordinates at which the basis function is centered.

`nggradbasfunkval`
Calculates the value of nuclear gradient of the gradient of a real basis-function, Ψ, at the spatial point $\mathbf{r} = (x, y, z)$. I.e, the elements of the tensor $= \nabla_{\mathbf{R}}[\nabla\Psi(\mathbf{r})]$. The output is stored in the 3x3 matrix nggrad, according to the following ordering:

$$
\begin{aligned}
nggrad(1,1) &= \tfrac{\partial^2\Psi}{\partial R_x \partial x}, \quad nggrad(1,2) = \tfrac{\partial^2\Psi}{\partial R_y \partial x}, \quad nggrad(1,3) = \tfrac{\partial^2\Psi}{\partial R_z \partial x} \\
nggrad(2,1) &= \tfrac{\partial^2\Psi}{\partial R_x \partial y}, \quad nggrad(2,2) = \tfrac{\partial^2\Psi}{\partial R_y \partial y}, \quad nggrad(2,3) = \tfrac{\partial^2\Psi}{\partial R_z \partial y} \\
nggrad(3,1) &= \tfrac{\partial^2\Psi}{\partial R_x \partial z}, \quad nggrad(3,2) = \tfrac{\partial^2\Psi}{\partial R_y \partial z}, \quad nggrad(3,3) = \tfrac{\partial^2\Psi}{\partial R_z \partial z}
\end{aligned}
$$

`nglaplacebasfunkval.f90`
Calculates the value of nuclear gradient of the laplacian of a real basis-function at the spatial point $\mathbf{r} \in \mathbf{R}^3$. I.e, The function $= \nabla_{\mathbf{R}}[\nabla^2\Psi(\mathbf{r})]$ The output is stored in the array, nglapl, according to the following ordering:

nglapl(1) $= d\nabla^2\Psi(\mathbf{r})/dR_x$
nglapl(2) $= d\nabla^2\Psi(\mathbf{r})/dR_y$
nglapl(3) $= d\nabla^2\Psi(\mathbf{r})/dR_z$

`getvxcr.f90`
This subroutine calculate the r-dependent matrix elements $\Phi_i(\mathbf{r})V_{xc}(\mathbf{r})\Phi_j(\mathbf{r})$ and the nuclear gradient of these matrix elements. After integration over the entire 3-DIM space with some quadrature these will become the exchange correlation potential matrix elements to be plugged into the Kohn-sham hamiltonian, and the corresponding force expression for the exchange correlation contribution to the interatomic forces.

`atomicradii.f90`
This function takes the atomic number as input and returns the empirical radii of Bragg and Slater, J. C. Slater, J. Chem. Phys. **41**, 3199 (1964).

`lebedev.f90`
This subroutine provides the weights and roots of the Lebedev quadrature used for the integration on the surface of a sphere.

`chebgauss.f90`
This subroutine calculates the weights and roots of the Chebyshev-Gauss quadrature of the second kind used for the integration along the radial direction.

`sofmu.f90`
This is the function $s(\mu) = \frac{1}{2}(1 - f(\mu))$ described by Eqn (3) in A. D. Becke, J. Chem. Phys. **88**, 2547 (1988). Here the recommended third order (k=3) is used.

`pvoronoi.f90`
This function is given by Eqn. (13) in A. D. Becke, J. Chem. Phys. **88**, 2547 (1988).

`getvxc.f90`
This subroutine calculates the exchange correlation potential matrix elements, $\langle i|V_{xc}|j\rangle$, and the nuclear gradients of these matrix elements, $\nabla_{\mathbf{R}}\langle i|V_{xc}|j\rangle$, using the resolution of the identity into fuzzy polyheadra prescribed by A. D. Becke, J. Chem. Phys. **88**, 2547 (1988), together with Gauss-Chebysshev and Lebedev quadrature.

`excdens.f90`
This function calculates the exchange correlation energy density.

`excdr.f90`
This function calculates the exchange correlation energy density.

`exc.f90`
This function calculates the exchange correlation energy from the exchange correlation energy density by using the resolution of the identity into fuzzy polyheadra prescribed by A. D. Becke, J. Chem. Phys. **88**, 2547 (1988), together with Gauss-Chebysshev and Lebedev quadrature.

`DFT.f90`
Subroutine performing density functional theory calculations.

`quadcheck.f90`
Calculates the total number of electrons by integrating the charge density, $\rho(\mathbf{r})$ by using the resolution of the identity into fuzzy polyheadra prescribed by A. D. Becke, J. Chem. Phys. **88**, 2547 (1988), together with Gauss-Chebysshev and Lebedev quadrature. This is used to check the quality of the quadrature mesh.

`TRACE.f90`
Calculates the trace of a square matrix.

`hessianbasfunk.f90`
Calculates the hessian of a real basis-function, Ψ, at the spatial point $\mathbf{r} \in \mathbf{R}^3$. I.e, the matrix $h_{ij} = \frac{\partial^2 \Psi}{\partial x_i \partial x_j}$

`hessianrho.f90`
This subroutine calculates the hessian matrix of the charge density, $\rho(\mathbf{r})$, i.e:
$h_{i,j,k} = \frac{\partial^2 \rho}{\partial R(k)_i \partial r_j}$ where $R(k)_i$ = i:th coordinate of the nucleus positioned at $\mathbf{R}(k)$ and r_j = the j:th coordinate of the spatial electron coordinate $\mathbf{r}$.

`dftforcedens.f90`
This subroutine calculates the exchange correlation force density at the point $\mathbf{r} \in \mathbf{R}^3$ in space.

`getxcforce.f90`
This subroutine calculates the exchange correlation contribution to the interatomic forces by integration of the exchange correlation force density. This is done by using the resolution of the identity into fuzzy polyheadra prescribed by A. D. Becke, J. Chem. Phys. **88**, 2547 (1988), together with Gauss-Chebysshev and Lebedev quadrature.

`pnonorm.f90`
This function is given by Eqn. (13) in A. D. Becke, J. Chem. Phys. **88**, 2547 (1988).

`tmu.f90`
This function is the function $t(\mu)$ described in appendix B Eqn (B9) In J. Chem. Phys. **98**, 5612, (1993),

`gradpvoronoi.f90`
This subroutine calculates all the nuclear gradients of the Becke weight function centered at atom I. See Appendix B in J. Chem. Phys. **98**, 5612 (1993).

`gradoverlap.f90`
Here we calculate the gradient of the overlap matrix when all the basis functions have been shifted with the vector $\mathbf{r} \to \mathbf{R} + \mathbf{r}$, i.e $\Phi(\mathbf{r}) \to \Phi(\mathbf{r} + \mathbf{R})$.

`relaxf.f90`
This subroutine searches for the atomic positions at which all nuclear forces are $<$ `FTOL`.

`mulliken.f90`
Subroutine that calculates the Mulliken charges of the different atoms and prints the result in the stout.

7.1.9 OPENMPVERSION

This subdirectory contains all the fortran source files (*.f90) of the opemp version of the uquantchem code together with different Makefiles. The source files within this directory have the same names and functionality as the *.f90 files located in the `SERIALVERSION` - directory. For a complete listing of the fortran source files contained in the `OPENMPVERSION`-directory se the preceding subsection.

7.1.10 MPI_VERSION

This subdirectory contains all the fortran source files (*.f90) of the mpi version of the uquantchem code together with different Makefiles. The source files within this directory have the same names and functionality as the *.f90 files located in the `SERIALVERSION` - directory. For a listing of the fortran source files contained in the `MPI_VERSION`-directory se the above section describing the `SERIALVERSION`. However, there are a couple of source files

in the `MPI_VERSION`-directory that are unique to the mpi-version of the code. Below follow a complete listing of these files:

`diagscalapack.f90`
Subroutine that uses the ScaLapack routine PDSYEVD to diagonalize the CISD hamiltonian matrix by the divide and conquer algorithm.

`gridsetup.f90`
This subroutine factorizes the number of processors (nproc) into nprow and npcol, that are the sizes of the 2-d processors mesh.

`countee.f90`
Calculates the diagonal electron-electron integrals $(ij|ij)$ and counts the number of integrals that give a considerable contribution to the exchange and Hartree matrices K_{ij} and J_{ij}.

Here we use the Schwartz inequality $|(i,j,k,l)| \leq [|(i,j,i,j)(k,l,k,l)]^{1/2}$ (See Eqn 9.12.25 p.404 in T. Helgaker et al's *"Molecular Electronic-Structure Theory"*) together with Eqn. (56) and (57) page. 86, in Haettig, *"Multiscale Simulation Methods in Molecular Sciences"* , J. Grotendorst, N. Attig, S. Blugel, D. Marx (Eds.), Institute for Advanced Simulation, Forschungszentrum Julich, NIC Series, Vol. **42**, ISBN 978-3-9810843-8-2,pp. 77-120, 2009, to decide wich integrals give considerable contributions.

Bibliography

[1] S. H. Vosko, L. Wilk and M. Nusair, Can. J. Phys. **58**, 1200 (1980).

[2] John P. Perdew, Kieron Burke, Matthias Ernzerhof, Phys. Rev. Lett, **77**, 3865 (1996)

[3] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, Dover, Mineola, New York (1996)

[4] X. Li, S. M. Smith, A. N. Markevitch, D. A. Romanov, R. J. Lewis and H. B. Schlegel, Phys. Chem. Chem. Phys. **7**, 233 (2005)

[5] D. B. Cook, Handbook of Computational Chemistry, Dover, Mineola, New York (2005)

[6] J. M. Thijssen, *Computational Physics*, Page 330. (Cambridge)

[7] P. Puley, Chem. Phys. Lett. **73**, 393 (1980)

[8] P. Puley, J. Comp. Chem., **3**, 556 (1982).

[9] S. E Koonin and D. C. Meredith, *Computational Physics*, Page 213-214, (West wide Press 1990)

[10] C. J. Umrigar, M. P. Nightingale, K. J. Runge, J. Chem. Phys. **99**, 2865 (1993)

[11] S. Manten and A. Lüchow, J.Chem.Phys. **115**, 5362 (2001)

[12] A. M. N. Niklasson, C. J. Tymczak, and M. Challacombe, Phys. Rev. Lett. **100**, 123004 (2008); Phys. Rev. Lett. **97**, 123001 (2006)

[13] J. A Pople, R. Krishnan, H. B. Schlegel and J. S. Binkley Int. J. Quantum Chemistry: Quantum Chemistry Symposium **13**, 225-241, see eqn (21) p. 229

[14] H. Bernhard Schlegel, Theor. Chem. Acc. **103**, 294-296, here see eqn 3-7

[15] Anders M. N. Niklasson and Marc J. Cawkwell , Phys. Rev. B **86**, 174308 (2012)

[16] Anders M. N. Niklasson, Peter Steneteg,Anders Odell, Nicolas Bock, Matt Challacombe, C. J. Tymczak, Erik Holmstrm, Guishan Zheng and Valery Weber, J. Chem. Phys. **130** , 214109 (2009)

[17] B. Mielich, A. Savin, H. Stoll and H. Preuss, Chem. Phys. Lett. **157**, 200 (1989)

[18] A. D. Becke, Phys. Rev. A, bf 38, 3098 (1988)

[19] P. J. Stephens, F. J. Devlin, C. F. Chabalowski and M. J. Frisch J. Phys. Chem bf 98 11623 (1994)

[20] B. G. Johnsonm P. M. W. Gill and J. A. Pople, J. Chem Phys. **98** 5612 (1993)