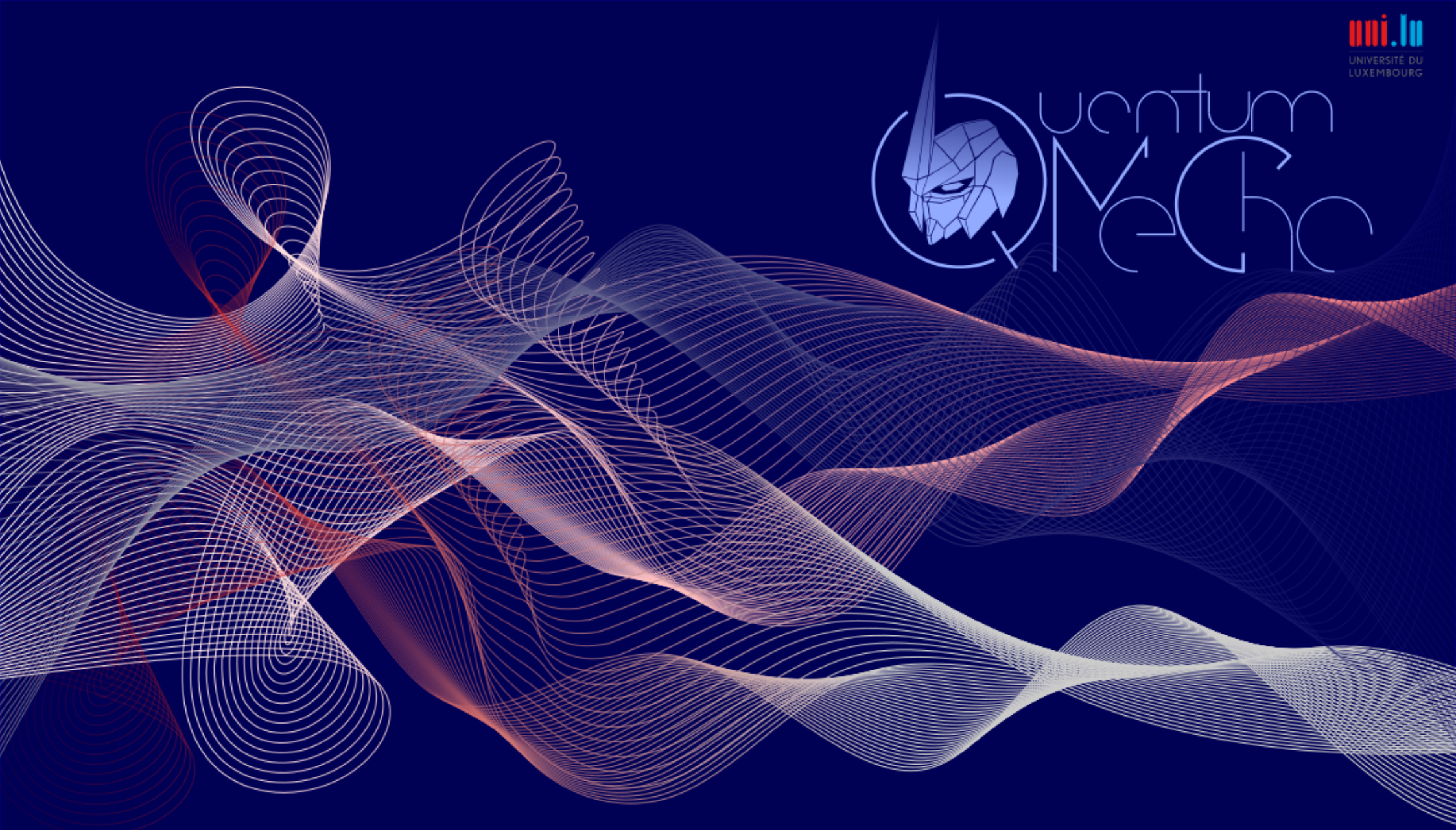

User Manual 0.1

Users manual of the Quantum MeCha (QMeCha), quantum Monte Carlo package.
The term *mecha* /'mɛkə/ is the abbreviation of *mechanical*, which in Japanese manga and anime culture
refers to all the piloted mechanical devices (robots, cars, airships, computers, and others) introduced in the science fiction genre.

QMeCha is a quantum Monte Carlo code developed in Fortran03 for calculations of electronic states of molecular systems,
and published in the github repository https://github.com/QMeCha.

This manual is not complete, and will be extended in the future to include more examples and extensive guidelines for realistic quantum Monte Carlo calculations.
For any questions and to request further details please contact to the creator and main developer.

Creator, main developer and scientific supervisor:
Barborini, Matteo[†,**]

Project participants:
Charry Martinez, Jorge Alfonso[*]: Electron-Positron wave functions
Ditte, Matej[*]: Electrons embedded in quantum Drude Oscillators
Andronikos, Leventis[*]: Electronic wave functions and correlation
Kafanas, Georgios[†]: Compilation, portability and optimization

[†] *Université du Luxembourg -Faculté des Sciences, des Technologies et de Médecine - HPC Platform*; Campus Belval, avenue de la Fonte, L-4364 Esch-sur-Alzette
[*] *Université du Luxembourg - Faculté des Sciences, des Technologies et de Médecine - Département Physique et sciences des matériaux*; Campus Limpertsberg, 162 A, avenue de la Faïencerie, L-1511 Luxembourg

[**] e-mail: matteo.barborini@gmail.com

Contents

List of Publications

QMeCha: quantum Monte Carlo package for fermions in embedding environments
Matteo Barborini, Jorge Charry, Matej Ditte, Andronikos Leventis, Georgios Kafanas, Alexandre Tkatchenko
arXiv **2025**

Extending quantum-mechanical benchmark accuracy to biological ligand-pocket interactions
Mirela Puleva, Leonardo Medrano Sandonas, Balázs D. Lőrincz, Jorge Charry, David M. Rogers, Péter R. Nagy and Alexandre Tkatchenko
Nat. Commun. **16**, 8583 **2025**

Watch out electrons!: Positron Binding Redefines Chemical Bonding in Be_2
Rafael Porras Roldan, Jorge Charry, Felix Moncada, Roberto Flores-Moreno, Marcio Teixeira Varella and Andrés Reyes
Chem. Sci. Accepted Manuscript **2025**

Reproducibility of fixed-node diffusion Monte Carlo across diverse community codes: The case of water–methane dimer
Flaviano Della Pia, Benjamin X. Shi, Yasmine S. Al-Hamdani, Dario Alfé, Tyler A. Anderson, Matteo Barborini, Anouar Benali, Michele Casula, Neil D. Drummond, Matúš Dubecký, Claudia Filippi, Paul R. C. Kent, Jaron T. Krogel, Pablo López Ríos, Arne Lüchow, Ye Luo, Angelos Michaelides, Lubos Mitas, Kousuke Nakano, Richard J. Needs, Manolo C. Per, Anthony Scemama, Jil Schultze, Ravindra Shinde, Emiel Slootman, Sandro Sorella, Alexandre Tkatchenko, Mike Towler, C. J. Umrigar, Lucas K. Wagner, William A. Wheeler, Haihan Zhou and Andrea Zen
J. Chem. Phys. **163 (10)**, 104110 **2025**

Molecule–Environment Embedding with Quantum Monte Carlo: Electrons Interacting with Drude Oscillators
Matej Ditte, Matteo Barborini and Alexandre Tkatchenko
J. Chem. Theory Comput. **21 (9)**, 4466-4480 **2025**

Quantum Drude oscillators coupled with Coulomb potentialas an efficient model for bonded and non-covalent interactions in atomic dimers
Matej Ditte, Matteo Barborini and Alexandre Tkatchenko
J. Chem. Phys. **160**, 094309 **2024**

Molecules in Environments: Toward Systematic Quantum Embedding of Electrons and Drude Oscillators
Matej Ditte, Matteo Barborini, Leonardo Medrano Sandonas and Alexandre Tkatchenko
Phys. Rev. Lett. **131**, 228001 **2023**

Correlated Wave Functions for Electron–Positron Interactions in Atoms and Molecules
Jorge Charry, Matteo Barborini and Alexandre Tkatchenko
J. Chem. Theory Comput. **18(4)**, 2267-2280 **2022**

The three-center two-positron bond
Jorge Charry, Félix Moncada, Matteo Barborini, Laura Pedraza-González, Márcio T. do N. Varella, Alexandre Tkatchenko and Andrés Reyes
Chem. Sci. **13**, 13795-13802 **2022**

List of Symbols

N_n Number of nuclei

N_e Number of electrons

N_p Number of positrons

N_f Number of electrons and positrons in the system ($N_e + N_p$)

N_O Number of Quantum Drude Oscillators

N_C Number of fixed point-charges

$\mathbf{r}$ Coordinate vector of a single fermion electron or positron

σ Spin state of a single fermion electron or positron

$\mathbf{x}$ 4-dimensional vector of Cartesian and spin coordinates of an electron or positron $\mathbf{x} = (\mathbf{r}, \sigma)$

$\bar{\mathbf{x}}$ $4N_f$-dimensional vector of Cartesian and spin coordinates of all electrons and positrons $\bar{\mathbf{x}} = \{\mathbf{x}_i\}_{i=1}^{N_f}$

$\bar{\mathbf{r}}$ $3N_f$-dimensional vector of the Cartesian coordinates of all electrons and positrons $\bar{\mathbf{r}} = \{\mathbf{r}_i\}_{i=1}^{N_f}$

Q Number of orbitals in the electronic Jastrow.

$\mathbf{r}^d$ 3-dimensional vector of a drudonic coordinate

$\mathbf{r}^e$ 3-dimensional vector of an electron coordinate

$\mathbf{r}^p$ 3-dimensional vector of a positron coordinate

P Number of orbitals in the positronic Jastrow.

$\bar{\mathbf{r}}^d$ $3N_O$-dimensional vector of the drudonic coordinates $\bar{\mathbf{r}}^d = \{\mathbf{r}_i^d\}_{i=1}^{N_O}$

$\bar{\mathbf{r}}^e$ $3N_e$-dimensional vector of the electronic coordinates $\bar{\mathbf{r}}^e = \{\mathbf{r}_i^e\}_{i=1}^{N_e}$

$\bar{\mathbf{r}}^p$ $3N_p$-dimensional vector of the positronci coordinates $\bar{\mathbf{r}}^p = \{\mathbf{r}_i^p\}_{i=1}^{N_p}$

μ, ν, η Indexes running on the basis sets.

m_i, q_i Mass and charge of the ith fermion.

$\chi_\mu(\mathbf{r}^e)$ Atomic orbital μ occupied by an electron.

M_a, Z_a Mass and charge of the ath nucleus.

$\varphi^\mu(\mathbf{r}^p)$ Atomic orbital μ occupied by a positron.

$\mu_i, \rho_i, \omega_i, \mathbf{R}_i^O$ Mass, charge, frequency, and Carthesian vector of the position of the ith quantum Drude Oscillator.

$\mathbf{X}$ Matrix of the atomic orbitals occupied by the electrons($Q \times N_e$).

$\mathbf{\Phi}$ Matrix of the atomic orbitals occupied by the positrons ($P \times N_p$).

$\bar{\mathbf{R}}$ $3N_n$-dimensional vector of the nuclear coordinates $\bar{\mathbf{R}} = \{\mathbf{R}_a\}_{a=1}^{N_n}$

$\mathbf{R}_a$ Vector of the position of the ath nuclei.

$\boldsymbol{\chi}_i$ Atomic orbital vector occupied by the ith electron. ith column of the matrix $\mathbf{X}$.

$\mathbf{R}_i^C$ Vector of the position of the ith point charge in the embedding environment.

$\boldsymbol{\varphi}_h$ Atomic orbital vector occupied by the hth positron. hth column of the matrix $\mathbf{\Phi}$.

Q_i Value of the ith point charge in the embedding environment.

$\boldsymbol{\chi}^\mu$ Vector of the atomic orbital μ occupied by all the electrons. μth row of the matrix $\mathbf{X}$.

List of Symbols

$\hat{\mathsf{H}}_{tot}$ Total Hamiltonian including molecular systems and embedding environment.

$\boldsymbol{\varphi}^{\mu}$ Atomic orbital vector occupied by the hth positron. μth row of the matrix $\boldsymbol{\Phi}$.

$\hat{\mathsf{H}}_{f}$ Molecular Hamiltonian, including positrons if present.

$\boldsymbol{\Gamma}^{ee}$ Coupling matrix for the electron-electron dynamical Jastrow factor.

$\hat{\mathsf{H}}_{O}$ Hamiltonian of the systen of QDOs.

$\boldsymbol{\Gamma}^{pp}$ Coupling matrix for the positron-positron dynamical Jastrow factor.

$\nabla^2_{\mathbf{r}}$ Laplacian operator for the general 3-coordinates vector $\mathbf{r}$.

$\boldsymbol{\Gamma}^{ep}$ Coupling matrix for the electron-positron dynamical Jastrow factor.

$\mathsf{V}(\bar{\mathbf{r}};\bar{\mathbf{R}})$ Potential energy including all the Coulomb interactions between electrons, positrons and nuclei.

$\hat{h}_i^O$ Single particle Hamiltonian for the ith drudon.

$\mathsf{V}_{O,f}$ Coulomb interactions between molecular system and the QDOs.

$\mathsf{V}_{C,f}$ Coulomb interactions between molecular system and the point charges in the embedding environment.

$\hat{\mathsf{V}}^a_{\mathrm{ECP}}(\mathbf{r})$ Effective core potential for the ath nuclei.

$\mathsf{V}^a_{loc}(\mathbf{r})$ Local part of the effective core potential for the ath nuclei.

$\mathsf{V}^a_l(\mathbf{r})$ Non-local part of the effective core potential for the ath nuclei with angular momentum l.

$\hat{\mathsf{P}}^a_l$ Projection operator on the real spherical harmonic centred on the ath nuclei with angular momentum l.

$\mathbf{E}$ Vector of the external static electric field.

$\boldsymbol{\mu}$ Vector of the total dipole of the system

$\mathsf{V}_{\mathbf{E}}$ Energy contribution from the external electric field $\mathbf{E}$.

$\Psi_T(\bar{\mathbf{x}},\bar{\mathbf{r}}^d)$ Total trial wavefunction of the molecular system and the embedding potential.

$\Psi_f(\bar{\mathbf{x}})$ Trial wavefunction of the molecular system including eventually the positronic coordinates.

$\Psi_d(\bar{\mathbf{r}}^d)$ Wavefunction of the drudons in the embedding environment.

$\Psi_{f,d}(\bar{\mathbf{r}},\bar{\mathbf{r}}^d)$ Correlation function between the molecular system and the drudons in the environment.

$\psi_e(\bar{\mathbf{x}}^e)$ Electronic part of the wavefunction, only depends on electronic and parametric nuclear coordinates.

$\psi_p(\bar{\mathbf{x}}^p)$ Positronic part of the wavefunction, only depends on positronic and parametric nuclear coordinates.

$\mathcal{J}(\bar{\mathbf{x}})$ Total Jastrow factor.

$J_c(\bar{\mathbf{x}})$ Cusp terms in the Jastrow factor.

$J_d(\bar{\mathbf{r}})$ Jastrow factor including the 3/4 body terms.

$\mathrm{pf}[\mathbf{P}]$ Pfaffian of the matrix $\mathbf{P}$

$\mathbf{G}$ Singlet geminal matrix.

$\mathbf{T}^{\uparrow},\mathbf{T}^{\downarrow}$ Triplet geminal matrices, for spin up and spin down particles of the same type.

$\mathbf{G}_{ij}=\phi_G(\mathbf{r}_i^{\uparrow},\mathbf{r}_j^{\downarrow})$ Singlet geminal coupling for the ith and jth electrons of opposite spin.

$\mathbf{T}^{\sigma}_{ij}=\phi_T(\mathbf{r}_i^{\sigma},\mathbf{r}_j^{\sigma})$ Triplet geminal coupling for the ith and jth electrons of identical spin ($\sigma=\{\uparrow,\downarrow\}$).

$\mathbf{S}^{\sigma}$ Slater matrices for the two spin populations ($\sigma=\{\uparrow,\downarrow\}$).

$\mathsf{E}\left[\Psi_T(\boldsymbol{\alpha})\right]$ Energy functional on the trial wavefunction $\Psi_T(\boldsymbol{\alpha})$.

$\boldsymbol{\alpha}$ Parameters' vector of the trial wavefunction.

$\mathrm{E}_l(\bar{\mathbf{r}}; \boldsymbol{\alpha})$ Local energy.

$\Pi(\bar{\mathbf{r}}; \boldsymbol{\alpha})$ Probability density

$\bar{\mathrm{E}}_l(\boldsymbol{\alpha})$ Sample mean value of the local energy.

$s^2_{\bar{\mathrm{E}}_l(\boldsymbol{\alpha})}$ Sample variance of the local energy distribution.

$\sigma_{\bar{\mathrm{E}}_l(\boldsymbol{\alpha})}$ Statistical error on the mean of the local energy.

f_{α_k} Generalized force for parameter α. Derivative of the energy functional with respect to α_k.

$\mathcal{O}_{\alpha_k}$ Dervative of the logarithm of the wavefunction with respect to the parameter α_k.

S Covariance matrix of the local operators $\mathcal{O}_{\alpha_k}$ on the sampling.

I

Introduction to the code

I.1 Compilation of the QMeCha code

The code until now only requires gfortran or Intel Fortran compilers and the linking of BLAS library. In particular, we recommend the use of OpenBLAS or MKL. We have not tested the AOCL-LibM from AMD up till now.
In order to compile the code it is first necessary to construct the configuration file. Some examples of this file can be found in the code/config directory.
A standard compilation in serial using Intel OneApi 2023 can be done with the following configuration file

Serial compilation with ifort and MKL

```
# Compiler
FC = ifort
# Compiler flags
FCFLAGS= -Ofast -fpp -free -m64 -auto
# MKL libraries
FCFLAGS+=-I${MKLROOT}/include/intel64/lp64 -I"${MKLROOT}/include"
LDFLAGS= ${MKLROOT}/lib/intel64/libmkl_blas95_lp64.a -Wl,--start-group \
    ${MKLROOT}/lib/intel64/libmkl_intel_lp64.a \
    ${MKLROOT}/lib/intel64/libmkl_sequential.a \
    ${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -lpthread -lm -ldl
```

Notice that the variable `${MKLROOT}` is specified when loading the Intel variables through `${INSTALLATIONDIR}/setvars.sh intel64`. Furthermore, the optimization level `-Ofast` might be too aggressive and could be substituted with `-O2` without substantial loss of performance.

Similar performances are obtained if compiling with `gfortran` and the use of MKL libraries

Serial compilation with gfortran and MKL

```
# Compiler
FC = gfortran
# Compiler flags
FCFLAGS= -O3 -cpp -ffree-form -ffast-math -ftree-vectorize \
    -funroll-loops -fallow-argument-mismatch -ffree-line-length-none
# MKL libraries
FCFLAGS+=-I${MKLROOT}/include/intel64/lp64 -I"${MKLROOT}/include"
LDFLAGS=-m64  ${MKLROOT}/lib/intel64/libmkl_blas95_lp64.a \
    ${MKLROOT}/lib/intel64/libmkl_lapack95_lp64.a -Wl,--start-group \
    ${MKLROOT}/lib/intel64/libmkl_gf_lp64.a \
    ${MKLROOT}/lib/intel64/libmkl_sequential.a \
    ${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -lpthread -lm -ldl
```

In order to be sure of the correct linking we suggest the use the MKL Intel linking tool.
There is a list of different Mathematical libraries that can be used alternatively to MKL. We do suggest in particular the use of OpenBLAS or BLIS. The static linking of the OpenBLAS library within the gfortran compiler can be seen in the following example.

Serial compilation with gfortran and OpenBLAS

```
# Compiler
FC = gfortran
# Compiler flags
FCFLAGS= -O3 -cpp -ffree-form -ffast-math -ftree-vectorize \
    -funroll-loops -fallow-argument-mismatch -ffree-line-length-none
# OpenBLAS libraries
FCFLAGS+=-I"${ROOTOPENBLAS}/include"
LDFLAGS = ${ROOTOPENBLAS}/lib/libopenblas.a -lm -ldl
```

I.1.1 Configuration for OpenMP and MPI

A first level of parallelization in the code, that splits the loop over the Monte Carlo walkers on different cores can be easily obtained with `openmp`. In order to compile with this protocol it is sufficient to add to the config file the line `FCFLAGS+=-qopenmp`

`-D_OMP` after the optimization flags when using the Intel compiler, or the line `FCFLAGS+=-fopenmp -D_OMP` when using the GNU compiler.

Although this approach speeds up the performance, we still recommend in general to run the calculations by parallelizing the Monte Carlo walkers through the MPI protocol.

The code has been successfully tested with IntelMPI and OpenMPI.

The parallel compilation with Intel MPI can be done by using as compiler the wrapper `mpiifort`, and specifying the MPI flag for the preprocessor `FCFLAGS+=-D_MPI` so that the full configuration file becomes

IntelMPI with ifort and MKL

```
# Compiler
FC = mpiifort
# Optimization flags
FCFLAGS= -Ofast -fpp -free -m64 -auto
FCFLAGS+=-qopenmp -D_OMP
FCFLAGS+=-D_MPI08
# MKL libraries
FCFLAGS+=-I${MKLROOT}/include/intel64/lp64 -I"${MKLROOT}/include"
LDFLAGS= ${MKLROOT}/lib/intel64/libmkl_blas95_lp64.a -Wl,--start-group \
    ${MKLROOT}/lib/intel64/libmkl_intel_lp64.a \
    ${MKLROOT}/lib/intel64/libmkl_sequential.a \
    ${MKLROOT}/lib/intel64/libmkl_core.a \
    ${MKLROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.a \
    -Wl,--end-group -liomp5 -lpthread -lm -ldl
```

In the same way, the compilation using only GNU software, such as OpenMPI, OpenBLAS and the GNU Fortran compiler, can be done using the configuration file

OpenMPI with gfortran and OpenBLAS

```
# Compiler
FC = mpif90
# Compiler flags
FCFLAGS= -O3 -cpp -ffree-form -ffast-math -ftree-vectorize \
    -funroll-loops -fallow-argument-mismatch -ffree-line-length-none
FCFLAGS+=-fopenmp -D_OMP
FCFLAGS+=-D_MPI08
# OpenBLAS libraries
FCFLAGS+=-I"${ROOTOPENBLAS}/include"
LDFLAGS=LDFLAGS = ${ROOTOPENBLAS}/lib/libopenblas.a -lm -ldl
```

Through the preprocessor it is possible to specify various headers for the MPI Fortran interfaces `-D_MPIh,-D_MPI,-D_MPI08`. We do recommend the use of the most recent 2008 interfaces with `-D_MPI08`, yet this might lead to compatibility issues one using OpenMPI with Intel Compilers. Thus, if errors occur, please revert to `-D_MPI` or `-D_MPIh`.

I.1.2 Compilation through Makefile

A list of predefined configuration files can be found in the `config/` directory that can be used to take inspiration. Each config file is named according to the info `${C}+${LIB}+${MPI}_${CN}.cnf` that must be then specified when calling the `make` command from the source `src/` directory.
In practice to compile with pure Intel on a laptop machine with the corresponding configuration file it is sufficient to call

Compilation with Intel oneAPI

```
cd src/
make C=ifort LIB=mkl MPI=impi+omp CN=LAPTOP
```

that will use the corresponding `ifort+mkl+impi+omp_LAPTOP.cnf` config file present in the `config/` directory and generate the binary files in the `bin/` directory. Adding the flat `-j` followed by the number of CPUs will automatically enable the compilation in parallel.
To clean the installation it is possible to call the commands `make clean` to remove the

compilation files, and `make veryclean` to also remove the compiled executables.

I.2 Execute the QMeCha code

QMeCha runs by reading a list of arguments that identify the various input files. In practice, to run the code without any kind of restart it is sufficient to write

Running a serial calculation
```
export OMP_NUM_THREADS=1
QMeCha -i input
```

where the flag `-i` takes as argument the input file that contains the links to all the required files and the parameters of the calculation that has to be executed.

For pure OMP calculations, the code can be run as

Running a pure OMP calculation
```
export OMP_NUM_THREADS=4 #(set n_wlk >= 4)
QMeCha -i input
```

where the number of walkers `n_wlk` must be set greater or equal to the number of threads. The number of walkers will specify in this case the total number of parallel Monte Carlo samplings that the code is executing.

For pure MPI calculations, the code can be run as

Running a pure MPI calculation
```
export OMP_NUM_THREADS=1
export MPI_NUM=4
mpiexec -np ${MPI_NUM} QMeCha -i input
```

so that there will be four consecutive Monte Carlo samplings assuming that the number of walkers `n_wlk` is left to default value of 1.

For a mixed OMP and MPI calculation we just need to specify the variable `OMP_NUM_THREADS` which at the moment must be set to be lower then the number of walkers per MPI task (this should be corrected).

Running a mixed MPI/OMP calculation
```
export OMP_NUM_THREADS=4 #(set n_wlk >= 4)
export MPI_NUM=4
mpiexec -np ${MPI_NUM} QMeCha -i input
```

In this case assuming that `n_wlk=4` in the input file, the total number of parallel Monte Carlo samplings will be 16.

There is no reason for the number of walkers to be larger then the number of cores that are used in the Monte Carlo run, at least for variational Monte Carlo and clearly for the optimization procedure that uses variational Monte Carlo for the samplings. It is thus recommended to use `OMP_NUM_THREADS` $\times$ `MPI_NUM` equal to the effective number of cores or at most equal to the effective number of threads in the calculation, that might be double the number of cores in the case of `intel hyper-threading`. In Monte Carlo since the latency times are usually very low, the gain in the use of `hyper-threading` is not so large.

In the following sections, we will describe the input files for the three basic calculations that QMeCha can do: single-point Variational Monte Carlo calculation, single-point Diffusion Monte Carlo calculation, and wave function optimization procedure.

Other details regarding the arguments of the code will be given in the next chapter, here we must point out that in addition to the input file the code also requires the file containing the molecular system's file, the basis set file, and if requested the pseudopotential file.

These files, described in individual chapters, can be assigned through the main input file or through the code's input arguments.

A full list of the input arguments will be discussed in the following section.

I.3 Optional Input arguments

QMeCha uses a set of different input arguments that can be used to specify some input files or calculation options. Usually, all the argument options can also be defined in the main input file of the calculations.

Be aware that the values given in the input file always overwrite the values given in the input arguments.

Table I.1: *Complete List of input argumets.*

Command	Variable	Descrition
`-h`	(empty)	Calls for the example on how to run the code and lists all the main input arguments that have to be specified.
`-pi`	char(3)	Calls printing of example inputs for different calculations (vmc,dmc,opt)
`-i`	char(100)	Name of input file for the calculation.
`-b`	char(100)	Fermionic Basis set file
`-m`	char(100)	Fermionic system's file
`-p`	char(100)	Pseudopotential file
`-bq`	char(100)	Drudonic Basis set file
`-mq`	char(100)	Drudonic system's file
`-sn`	char(100)	Name of the molecular system used for save files
`-j`	char(1)	Specification of the dynamical Jastrow factor type (d,a,f)
`-s`	(empty)	Calls for initialization of the wave function. This flag requires to be followed by additional flags regarding the molecular system's and basis set input files (-m .. -b .. -p .. -j ..).

The use of these input arguments can facilitate the manipulation of the initial wave function or the automatic application of a set of calculations to different systems.

In fact, the initialization of the wave function of a system can be executed by calling the commands

Wave function initialization

```
QMeCha -s -m ${molsysfile} -b ${basissetfile} \
    -p ${pseudopotential} -j f
```

The description of this procedure and its results will be discussed in depth in the next section which will discuss about the save files and input files of the code in general.

Here we want only to add that to run the code with additional input arguments it is sufficient to call its execution according to the example:

Running the code with input arguments

```
QMeCha -i input -m ${molsysfile} -b ${basissetfile} \
    -p ${pseudopotential} -j f
```

Again we must point out that the arguments in this case are always optional, and they are not required if the Molecular system file, the basis set file and the other necessary files are specified in the input file.

I.4 Files and directories

In this section, we report a brief summary of all the input and output files for the QMeCha code, which are discussed in depth in the sections that follow.

In particular, the list of input files, some of them optional, is reported in Tab. I.2 while the list of outputs is reported in Tab. I.3

Table I.2: *Complete list of QMeCha's input files*

File	Descrition
${qmcinputfile}	Main input file that specifies the type of run to be executed and can also specify the other input files, described in section (REF)..
${pseudopotential}	File containing the electronic pseudopotentials described in section (REF).
${basissetfile}	Basis set and wave function file for the fermions described in (REF).
${molsysfile}	Molecular systems' file, containing the fermionic properties and described in (REF).
${drdsysfile}	Drudonic systems' file, containing the properties of the drudons in the system and described in (REF).
${drdbasissetfile}	Basis set and wave function file for the drudons in the system described in (REF).

All the important output files are stored in the `wvfn.save` directory that contains the

saved parameters of the wave functions, together with some system information.

Table I.3: *Complete list of QMeCha's output files in the* `wvfn.save/` *directory*

File	Descrition
`wvfn.save/`	This is the generated directory that contains files containing the parameters of the optimized wave functions and their symmetry tables.
`slde.sav,slde_s.sav`, `sldp.sav,sldp_s.sav`	Molecular orbital file and corresponding symmetry file for the electronic and/or positronic Slater determinant wave function.
`sgme.sav,sgme_s.sav`, `sgmp.sav,sgmp_s.sav`	Singlet AGP wave function and symmetry file for electrons and/or positrons.
`tgme.sav,tgme_s.sav`, `tgmp.sav,tgmp_s.sav`	Triplet geminal wave function and symmetry file for electrons and/or positrons.
`pffe.sav,pffe_s.sav`, `pffp.sav,pffp_s.sav`	Pfaffian wave function and symmetry file for electrons and/or positrons.
`frmbss.sav`	Atomic orbitals' basis set.
`epobss.sav`	Electron-positron orbitals' basis set.
`qdobss.sav`	Drudonic orbitals' (centered on the QDO) basis set.
`frmdst.sav`	Distributions of electrons and positrons around the nuclei.
`jst.sav,jst_s.sav`	Jastrow parameters and symmetries.

The files named with the suffix `_s.sav` are those containing the tables of symmetries and the list of non-zero parameters for every component in the wave function.
The file `frmdst.sav` contains the distribution of fermionic particles, with relative spin, and is used to localize the particle distributions around the atomic centers. This term is important especially when describing a weakly interacting system or fragments in the dissociation limit, since it fastens the convergence of the energy and of the observables.
For an in depth description of the various files please refer to the corresponding sections.

II

Molecular system files

II.1 Fermionic system input file

The molecular system file contains all the information regarding the model that the user wants to study, including the number of atoms and their initial positions, the number of fermionic particles and their total spins, the masses of the fermionic particles, and if present the intensity and direction of an external electric field.
The file includes a first fortran namelist `&molsys` followed (without spaces) by the list of atoms that are present and their coordinate.
For a single Lithium atom for example the molecular system files will be written as:

molsys.fle for the lithium atom

```
&molsys
n_at = 1
chrg = 0
mult = 2
/
 Li   0.000000     0.000000     0.000000
```

For a molecular system that contains only nuclei and electrons the minimal number of varibles that have to be inserted in the `&molsys` namelist are the number of atoms (`n_at`) the total charge of the system (`chrg`, sum of the nuclear and electronic charge) and the spin multiplicity (`mult`) of the electrons.
The last `n_at` lines of the file have to contain the list of atomic coordinates (one per line): in order for each line we must have name_of_the_atom (with the first capital letter) followed by its coordinates expressed in **atomic units**.

Table II.1: *Complete List of variables in the `&molsys` namelist.*

Variable	Type	Descrition
`n_at`	int32	Number of atoms in the system, that are read at the end of the molecular system file.
`chrg`	int32	Total charge of the system. It refers to the sum of all charges: nuclei, electrons and positrons (if present). This variable is not necessary if the number of electrons is specified.
`mult`	int32	Spin multiplicity of the system. It refers to the multiplicity in case only electrons are present. If positrons are present this varible is not considered, instead the specification of the total spins for the two fermionic populations is necessary.
`n_el`	int32	(Optional) Number of electrons. If present overwrites the charge.
`n_po`	int32	(Optional) Number of positrons. If present overwrites the charge.
`spin_e`	real64	(Optional) Total spin of the electrons overwrites the information of the multiplicity.
`spin_p`	real64	(Optional) Total spin of the positrons. Necessary if positrons are present.
`m_p`	real64	(Optional) Mass of the electrons. Default is 1.0.
`m_e`	real64	(Optional) Mass of the positrons. Default is 1.0.
`ext_E_field`	3*real64	(Optional) External electric field.
`units`	char(4)	(Optional) It sets the unit of length of the atomic coordinates to `bohr` (default) or `angs`.

An alternative way to define the charge and multiplicity of the electronic system is through the use of the variables that define the number of electrons (`n_el`) and their total spin (`spin_e`). These variables overwrite `chrg` and `mult`.

For example, the ground state of the lithium atom can be defined also as:

Alternative molsys.fle for the lithium atom

```
&molsys
n_at = 1
n_el = 3
spin_e = 0.5
/
 Li   0.000000     0.000000     0.000000
```

The full list of variables of the `&molsys` namelist is reported in Table II.1.

II.1.1 Ghost atoms

In order to increase the basis set in some regions of the space, to better the bond description or in the case of positrons, it is possible to introduce additional ghost atoms (`Gh`)

molsys.fle for the lithium atom with a ghost atom

```
&molsys
n_at = 2
n_el = 3
spin_e = 0.5
/
 Li   0.000000     0.000000     0.000000
 Gh   0.000000     0.000000     0.000000
```

that are uncharged basis set centers. The ghost atom's basis set must be specified in the basis set file described in the following section. The QMeCha code admits only one basis set for multiple ghost atoms added in the calculation.

II.1.2 External electric field

In order to compute the polarizability of the system we can add an external field by introducing the variable `ext_E_field` (which is a three dimensional vector of real numbers) in the `&molsys` namelist. The introduction of the external field changes the symmetry of the system. This should not worry the user since the code understands that the rotational symmetry is broken in the direction of the external field's vector.

molsys.fle for the lithium atom in an external field along the z axes.

```
&molsys
n_at = 1
n_el = 3
spin_e = 0.5
ext_E_field = 0.000, 0.000, 0.001
/
 Li   0.000000     0.000000     0.000000
```

To be more precise, before computing the symmetries the code centers the molecule in the center of atomic charges, and then tests the rotational symmetries (including inversion). In the presence of the external field, before checking the symmetries the molecule is translated along the direction of the electric field. After the symmetries are tested it is reset in its correct origin before starting the run.

II.1.3 Electron-positron calculations

When a system contains both electrons and positrons, the `chrg` and `mult` variables become completely insufficient and thus can be ignored.
For example, to define the system of lithium and positronium (LiPs), where we have four electrons of Li^- in a singlet state and one positron with up spin, we write the input as

molsys.fle for the LiPs system.

```
&molsys
n_at = 1
n_el = 4
spin_e = 0.0
n_po = 1
spin_p = 0.5
/
 Li   0.000000     0.000000     0.000000
```

Here, a part from defining the total number of electrons with their relative spin, we also have to introduce the variable for the number of positrons (`n_po`) and for the total spin of the positrons (`spin_p`).
Additional flags that can be introduce are those defining the electronic and positronic masses. These can be changed through the respective variables `m_e` and `m_p` that per

default are set to 1 au (in atomic units).

II.1.4 Molecular structures

As discussed above, the QMeCha code is able to find basic rotational symmetries of a molecule compound, but it does not find the principle symmetry axes, thus it is recommended that, if symmetries are present, the choice of the molecular structure mirrors those symmetries, like in the example of the water molecule.

molsys.fle for H_2O.

```
&molsys
n_at = 3
n_el = 10
spin_e = 0.0
units = 'angs'
/
O 0.0000  0.0000  0.1173
H 0.0000  0.7572 -0.4692
H 0.0000 -0.7572 -0.4692
```

If is also possible to insert the molecular structure in Angstrom, by specifying the flag `units = 'angs'`. The structure is then converted in atomic units by the code, using the conversion factor of 1.0 au = 0.529177210903(80)Å.

II.1.5 Effective core potentials

In QMeCha it is possible to replace core electrons through effective core potentials (ECP) written in the gaussian expansion form. The code supports pseudopotentials from M. Burkatzki, C. Filippi, M. Dolg[1,2], from J.R. Trail and R.J. Needs[3,4,5] and the most recent correlation consistent ones in the pseudopotentiallibrary.org[6,7,8,9,10,11,12].

To flag that the core electrons of a certain atom have to be replaced by and ECP it is sufficient to add an asterisk before the atom's name. In the following molecular input file, for example, the Carbon atom is replaced by a pseudo atom, while the Hydrogen atom is left unchanged.

molsys.fle for CH.

```
&molsys
n_at = 2
chrg = 0
mult = 2
units = 'angs'
/
*C 0.0000  0.0000 0.0000
 H 0.0000  0.0000 1.1200
```

It is essential to add the corresponding basis set (in the basis set file) associated with the `*C` atom. As a matter of fact, the code can also work with different carbon atoms, some described without approximation, and defined through `C` and others described with the ECP and defined through `*C`. These two types of atoms and pseudoatoms will be associated to different basis sets.

II.2 Pseudopotential file

When using effective core potentials it is essential to add file that contains all the relative information, that is read before the start of the calculations. Let us suppose that we are describing a water molecule, with ECPs on the oxygen and hydrogen atoms from M. Burkatzki, C. Filippi, M. Dolg[1,2], where for the latter no electrons are substituted but the divergence of the nuclear potential is removed.
This file is written as

pseudo.fle for H_2O.

```
&pseudo
n_pseudo = 2
ene_psd_cut = 0.1d-5
n_psd_grd_pnts = 6
/
O 2 2
 3 1
 1  9.29793903   6.00000000
 3  8.86492204  55.78763416
 2  8.62925665 -38.81978498
```

```
 2  8.71924452  38.41914135
H 1 0
 3
 1  4.47692410   1.00000000
 3  2.97636451   4.47692410
 2  3.01841596  -4.32112340
```

In this case at the beginning of the `pseudo.fle` we have the `&pseudo` namelist that contains the parameters of the pseudopotential integration.
In particular, it contains the variable `n_pseudo` that defines the number of pseudopotentials to be read, the energy cut-off to be used (that defines the accuracy of the intergration used for the non-local part (`ene_psd_cut`) and also the number of integration points used for the non-local part of the pseudopotentials (`n_psd_grd_pnts`). The detailed list of variables can be found in Table II.2.

Table II.2: *Complete List of variables in the `&pseudo` namelist.*

Variable	Type	Descrition
`n_pseudo`	int32	Number of effective core potentials to be read in the file.
`ene_psd_cut`	real64	(Optional) Accuracy of the energy, for lower values the pseudopotential effect is considered to be null. This defines the radius of the pseudopotential. Default value is 1.0d-5 [Ha].
`n_psd_grd_pnts`	int32	(Optional) Number of integration points on the sphere for the non-local components. Depends on the angular momentum of the pseudopotential. Default value is 6.
`psd_int_mthd`	int32	(Optional) Pseudo integration method. Switches between Locality approximation (LA) `psd_int_mthd=0` (Default) and Determinant Locality approximation (DLA) `psd_int_mthd=1`.

The namelist is followed (without spaces) by the list of pseudopotentials that are used in the calculation, in this case the one for Oxygen and the one for Hydrogen.
To describe the way the pseudopotential has to be written in the file, let us consider the first one related to the Oxygen atoms.
The first line contains in order: the name of the atom (0), the number of components in the pseudopotential (2) and finally the number of electrons that are replaced by the ECP (2, in this case the ones contained in the 1s orbital).
The second line contains the number of elements for each of the two components of the pseudopotential. We must point out that the first component must be the local one, followed by the non-local ones with increasing angular momentum. The local component will have three terms each defined by three numbers corresponding to: the exponential order, the exponent, and the coefficient of the expansion. The non local component associated to the angular momentum $l = 0$, is defined by one element for which the three number in the row represent the same variables as for the local component.

II.3 Drudonic system input file

The drudonic file system can be specified through the argument `-mq ${name_of_file}` or through the input variable `fle_coord_qdo` in the `&molsys` section of the main input file.

II.3.1 Purely drudonic system input file for pseudo-Argon dimer

A basic drudonic system file containing two interacting drudons using the Argon parametrization from ref. is written in the following way:

qdosys.fle for the pseudo-argon dimer

```
&qdosys
n_qdo = 2
/
# Label     |q|   omega   mu  \sigma_{center} \sigma_{drudon}  x y z
    q1    1.3314 0.7272 0.3020    0.0 0.0                  0.0  0.0 -2.5
    q2    1.3314 0.7272 0.3020    0.0 0.0                  0.0  0.0  2.5
```

The structure of the file is similar to that of the fermionic systems, yet additional param-

eters must be specified in order to characterize the various QDOs present in the system. In the next updates of the code this section might be progressively simplified, yet, up till now, we prefer to characterize the various QDOs one by one.

In the `&qdosys` section the main information regarding the number of QDOs in the system and other properties are contained. A full list of variables for this section is reported in Tab. II.3.

Table II.3: *Complete List of variables in the* `&qdosys` *namelist.*

Variable	Type	Descrition
`n_qdo`	int32	Number of quantum Drude oscillators in the system.
`n_pchrg`	int32	(Optional) Number of point charges in the system (default is 0).
`el_sigma`	real64	Cut-off for the electron-drudon interaction. Optional for purely drudonic sytems.
`ext_E_field`	3*real64	(Optional) External electric field.
`units`	char(4)	(Optional) It sets the unit of length of the atomic coordinates to `bohr` (default) or `angs`.
`qdoham`	char(3)	(Optional) Specifies the type of interaction between the QDOS, `'dip'` for dipole and `'cou'` for Coulomb (default).

The `&qdosys` section is then followed by a comment line starting with # that is read and ignored by the code, introducing the section that contains the list of the QDOs' coordinates and parameters.

In this example, the comment before the list specifies the type of parameters that have to be included to define a single QDO.

Thus in the lines:

```
# Label     |q|   omega   mu  \sigma_{center} \sigma_{drudon}  x y z
    q1    1.3314 0.7272 0.3020    0.0 0.0                 0.0  0.0 -2.5
    q2    1.3314 0.7272 0.3020    0.0 0.0                 0.0  0.0  2.5
```

the first is the label that is a `char(3)` variable in the code, the second, third and fourth parameters correspond respectively to the charge, frequency and mass of the QDOs, and are followed by the cut-off discussed in section [REF]. for the center of the QDO and for the drudonic particle. If these last two cut-offs are set to zero the Cusp functions in the Jastrow factor, described in section [REF], are automatically introduced in the wave function. The last three real numbers correspond to the positions of the centers of the QDOs. The code works by default with atomic units, thus by setting the variables `units='angs'` one can call the conversion to Bohr units inside the code, as seen for the fermionic systems.

II.3.2 QDOs and point charges

If the molecular compounds in the QDOs quantum Force-Field possess an intrinsic dipole, it is necessary to introduce also external point charges, such as in the work of Martyna *et al.*[REF]. In Martyna's representation, for example, the system file for the pseudo-water dimer is written as

qdosys.fle for the pseudo-water dimer

```
&qdosys
  n_qdo    = 2
  n_pchrg  = 6
/
# label  |q|  omega    mu   sigma_{center} sigma_{drudon}  x y z
   W1  1.1973 0.6287 0.3656      1.200 0.100  -0.41585 -0.00000 -0.28474
   W2  1.1973 0.6287 0.3656      1.200 0.100   0.42701  0.00000  5.23517
# label q sigma x y z
   W1  0.605   0.100  -0.91097 -1.43545 -0.62375
   W1  0.605   0.100  -0.91097  1.43545 -0.62375
   W1 -1.210   0.100  -0.41585 -0.00000 -0.28474
   W2  0.605   0.100   0.17458  0.00000  3.68976
   W2  0.605   0.100   1.69564  0.00000  6.13113
   W2 -1.210   0.100   0.42701  0.00000  5.23517
```

For this system, we need to specify the number of point charges through the flag `n_pchrg = 6`. These are read after the section listing the QDOs, starting with an empty comment line that is used to separate the variables' lines.

For the six point charges one needs to specify on a line, after the label that associates the point charge to a particular QDO, the charge of the point charge, and the cut-off that

screens the Coulomb divergence of the point charge according to what is described in [REF]. These parameters are then followed by the position of the point charge in Bohr (or Angstrom). Be aware that the units used for the positions of the QDOs must correspond to those used for the point charges.

II.3.3 Electron-QDOs embedding systems

In order to describe mixed systems of QDOs and electrons it is important to input also the molecular fermionic system file. The only additional parameter that is necessary to specify in the `&qdosys` section of the QDO system file, is the electron cut-off `el_sigma = 0.1` that dumps the interaction between the electrons and the Drudons and between the electrons and the QDO centers. The value of 0.1 Bohr is a reasonable choice of the parameter, as shown in REF.

III

Basis set files

III.1 Wave function file for electronic systems

The basis set file in QMeCha includes the information regarding the type of wave function used in the calculation, the type of cusp functions (for both electron-nucleus and electron-electron interactions) that we want to use, and the basis sets for each atom which include the fermionic orbitals and also the orbitals of the dynamical Jastrow factor. Let us consider the Helium atom, for which a basis set file can be defined as below.

basset.fle for He.

```
# Wave function type
 rsd
# Type and order of the jastrow one-body cusp
 1 0
# Type and order of the jastrow two-body cusp
 2 0
# Helium truncated cc-pVDZ basis set
He 2 3
S 3
  5.77000D+00  1.54891D-01 1G
  1.24000D+00  4.69987D-01 1G
  2.97600D-01  5.13027D-01 1G
P 1
  1.27500D+00  1.00000D+00 1G
# Jastrow factor basis set
S 1
  3.50000D+00  1.00000D+00 1G
S 1
  1.50000D+00  1.00000D+00 1G
P 1
  0.50000D+00  1.00000D+00 1G
```

The first two lines of the file refer to the wave function used for the calculation

```
# Wave function type
 rsd
```

that can be set to the different values listed in Table III.1. We must underline that, when there is only one electron the wave function is changed internally to the unrestricted Slater determinant. Moreover, when the system is spin polarized, the Triplet geminal wave functions (and the Pfaffian wave function) must be set to the unrestricted case in order to correctly describe the spin occupations.

Table III.1: *List of wave functions.*

Code	Wave function
`rsd/usd`	Restricted/Unrestricted Slater determinant (Sec. IV.1.1).
`rsg/usg`	Restricted/Unrestricted Antisymmetrized Geminal Power (Sec. IV.1.2).
`rtg/utg`	Restricted/Unrestricted Triplet Geminal Power (Sec. IV.1.4).
`rpf/upf`	Restricted/Unrestricted Pfaffian wave function (Sec. IV.1.3).
`psg`	Electron-positron geminal (Only usable for positrons).

The second four lines of the file are related to the parameters of the cups functions

```
# Type and order of the jastrow one-body cusp
 1 0
# Type and order of the jastrow two-body cusp
 2 0
```

The first two of these lines refer to the one-body cusp while the second two are related to the parameters of the same charge fermion-fermion cusps conditions. In both cases the first integer number refers to the type of cusp function used and the second refers to the order of that function. The complete list of types and orders accepted by QMeCha can be found in Table III.2 and a detailed description can be found in the initial sections

of this chapter. Setting the cups type to zero is equal to removing the cusps from the Jastrow factor.

It is recommended, in order to describe the cusp condition between opposite charge particles like electrons and nuclei, to use cusp functions that present the fast decay cusp $f(r) = \frac{1}{\beta_0} e^{-\beta_0 r}$ that is of type 1. On the other hand, for particles of the same charge it is best to use cusp functions that present the slowly decaying cusp $f(r) = \frac{1}{\beta_0(1+\beta_0 r)}$ that is of type 2.

Table III.2: *List of types of available cusp functions. The codes are valid for both nucleus-fermion and fermion-fermion cusps. For more details see Secs. IV.2.1 and ssec:2bcsp.*

Type	Order	Cusp description
0	0	No cusp function.
1	0-N	$\mathcal{F}(r) \propto \frac{1}{\beta_0} e^{-\beta_0 r} + \sum_{n=1}^{N} \beta_n e^{-\zeta_n r^2}$.
2	0-N	$\mathcal{F}(r) \propto \frac{1}{\beta_0(1+\beta_0 r)} + \sum_{n=1}^{N} \beta_n e^{-\zeta_n r^2}$.
3	0-N	Cusp function introduced by Drummond, Towler and Needs[13], see details in sections IV.2.1 and IV.2.2

After these first parts we have the list of basis sets for all the atoms that are present in the molecular system.

The basis set for a given atom is defined at first by a comment line followed by a line containing three parameters:

```
# Helium truncated cc-pVDZ basis set
He 2 3
```

that are respectively the name of the atom, the number of orbitals of the fermionic basis set, the number of orbitals of the Jastrow basis set. The name of the atom must correspond to the atom present in the molecular system file described in the previous section. In the case the basis set is used for a pseudoatom the name must be preceded by an asterisk. In this case the helium atom would be ***He**.

This session is followed by the list of fermionic orbitals that are written as

```
S 3
  5.77000D+00  1.54891D-01 1G
  1.24000D+00  4.69987D-01 1G
  2.97600D-01  5.13027D-01 1G
P 1
  1.27500D+00  1.00000D+00 1G
```

where the first two parameters define the angular momentum and the number of primitive functions that define the orbital. This descriptor line is followed by the corresponding primitive functions defined by three parameters each, that are respectively the exponential parameter of the primitive function, the linear parameter (coefficient of the orbitals' linear combination) and finally the parameter that defines the type of the primitive and its polynomial order.

The primitives' type (nX) is always defined by the number n, that defines the order of the polynomial r^{n-1} that multiplies the primitive function, followed by a character X, that defines the type of primitive function. A short list of all the possible orbitals is reported in Table III.3; details can be found in section IV.7.

Table III.3: *List of orbital functions (nX). The first number n is the order of the r^{n-1} factor that multiplies the primitives in this table.*

nX	Primitive form	Orbital description
nG	$r^{n-1} e^{-\zeta r^2}$	Gaussian type orbital.
nS	$r^{n-1} e^{-\zeta r}$	Slater type orbital.
nM	$r^{n-1} e^{-\frac{(\zeta r)^2}{1+\zeta r}}$	Mixed orbital type introduced by Petruzielo, Toulouse and Umrigar. The function is a Gaussian near the cusp that decays as a Slater type function[14,15].
nE	$r^{n-1}(1+\zeta r) e^{-\zeta r}$	Exponential type orbital without cusp. Can be used for all angular momentums.

The list of fermionic orbitals is then followed by a comment file and by the list of Jastrow related atomic orbitals that are constructed in the same way as the fermionic ones, although the primitive functions are not normalized. To the best of my knowledge, up to today, it is best to use uncontracted orbitals for the Jastrow factor that do not interfere with the nuclear-cusp, thus it is best to use G, M or E type orbitals or to use primitive

functions with higher order polynomials.

III.2 Electron-positron basis set file and wave function

For electron-positron systems the basis set file requires, together with the wave function type used to describe the electrons, also the one used to describe the positrons

basset.fle for electron-positron wave functions

```
# Electron-positron Wave function type
 usd usd
# Type and order of the nucleus-electron and nucleus-positron cusp
 1 0 2 0
# Type and order of the elec.-elec./pos.-pos. and elecn-pos. cusp
 2 0 1 0
# Basis sets
...
```

In the example above an unrestricted Slater determinant is used to represent both electrons and positrons, separately, in the field of the nuclei. The QMeCha code accepts all the wave function types associated to the electrons and reported in Tab. III.1.

Within the lines associated to the electron-nucleus cusp, one needs to add two names which are associated respectively to the type and order of the positron-nucleus cusp, and within the lines usually associated to the electron-electron cusp one needs to add the type and order of the electron-positron cusp conditions. The positron-positron cusp function is fixed to the same used for the electron-electron, due to the fact that the interactions are between same charge particles.

Please notice that the cusps recommended for the positrons are opposite to the ones recommended for the electrons due to the fact that nuclei and positrons repel each other, while electrons and positrons attract each other.

As described at the beginning of this chapter in order to better include correlation effects in electron-positron systems it is essential to add a wave function that explicitly describes the bound states. This wave function, that uses the tag `psg`, is constructed through positronium orbitals that are functions of the electron-positron pairs' distances and are not centred on the atoms.

In order to optionally add this positronium basis set, and the corresponding wave function, one must include in the basis set file atomic orbitals associated with the fictitious `Ps` atom writing for example:

Positronium basset.fle electron-positron wave functions

```
...
# Positronium basis set
Ps 1 1
S 2
  0.1569198E+00   0.3768996E+00 1G
  0.3665274E-01   0.6308656E+00 1G
# Electron-Positron Jastrow
S 1
  0.5000000E+00   0.1000000E+01 1G
# And other atoms
  ...
```

In this example the orbital basis is built from one contracted S orbital, with 2 primitive Gaussian functions. As you see we also add a Jastrow orbital, that has for the electron-nuclei interaction is now centered on the positron, and not normalized.

III.3 Drudonic basis set file and wave function

The Drudonic basis set file linked to the code by specifying the variable `fle_basis_qdo = 'name_of_file'` in the `&basset` name list in the main input file. Alternatively, it can be specified following the input argument `-bq` as anticipated in Tab. I.1.

The Drudonic basis set file is similar to the one constructed for the fermionic systems. For example, for the pure pseudo-Argon dimer it can be written as

basset.fle for pseudo-Argon dimer

```
# Wave function
 prd
# QDO-frm coupling (0=non, 1=dip)
 0
# One-body cusp for Drudon- Different QDO center
 0 0
```

```
# Two-body cusp for Drudon-Drudon
 0 0
# Basis set for the QDOs in prd wave function
 q1 2
 S 1
   0.75000000E+00  0.10000000E+01 1G
 S 3
   0.51753146E+00  0.19063064E+00 1G
   0.24400460E+00  0.53481227E+00 1G
   0.10683689E+00  0.11649066E+01 1G
#
 q2 2
 S 1
   0.75000000E+00  0.10000000E+01 1G
 S 3
   0.51753146E+00  0.19063064E+00 1G
   0.24400460E+00  0.53481227E+00 1G
   0.10683689E+00  0.11649066E+01 1G
```

The first two lines are used to define the wave function used for the QDO system, this can be `dip` or `prd` see sections [REF].

The third and fourth lines are used to specify the presence of the electron-Drudon coupling, which in this case is set to zero.

Finally, the lines

```
# One-body cusp for Drudon- Different QDO center
 0 0
# Two-body cusp for Drudon-Drudon
 0 0
```

are used to specify the cusps function within the Drudonic systems, with Coulomb interactions. The types of cusps functions that are admitted are the same used for the fermionic species, with the only difference being that here all Drudons are distinguishable. If the Dipole Hamiltonian is used these are ignored and always set to zero. It is important to remember that if the cusp conditions are not activated, the Coulomb potentials must be screened in order to remove any divergences, thus the cut-offs that appear in the QDO system file must be defined, to a positive value, see Sec. II.3.

If the `prd` wave function is used, as in this example, the cusp section is followed by the basis set for each QDO. Every QDO is considered independent in this example, thus the basis function is repeated for `q1` and `q2`. The structure of the basis set and the types of primitive and contracted type orbitals that can be used are the same that are used in the atomic basis set for fermions.

If the `dip` wave function is selected the basis set, even if present, is ignored.

IV

Wave functions

IV.1 Fermionic part of the wave function for electrons

The QMeCha code contains different types of fermionic wave functions based on single determinants or Pfaffians, that are summarized in Tab. III.1 Each wave function type appears in spin-restricted or unrestricted form, the former being an eigenstate also of the $\hat{S}^2$ operator, while the latter, which violates spin symmetry, is a combination of eigenstates of the $\hat{S}^2$ operator.

The wave function of a purely electronic system, with Cartesian and spin coordinates in the vector $\mathbf{x} = (\mathbf{r}, \boldsymbol{\sigma})$ is written as the product of two terms,

$$\Psi_T(\mathbf{x}) = \mathcal{F}(\mathbf{x})\mathcal{J}(\mathbf{x}) \tag{IV.1}$$

that are respectively a Fermionic part $\mathcal{F}(\mathbf{x})$, that describes the spatial and spin symmetries of the total wave function, and a Jastrow factor $\mathcal{J}(\mathbf{x})$ which is a Bosonic term, that reproduces the real wave function's cups conditions around the Coulomb potential divergences and includes explicit correlation between all the coordinates of the system.

Both the fermionic part of the wave function and the Jastrow factor can be described by different functional forms, which in the QMeCha code are based on single determinants or Pfaffians, summarized in Tab. III.1.

In the next sections, we will describe these types of wave functions and the implemented Jastrow factor, explaining the corresponding parameters in the `.sav` files stored in the `wvfn.save` directory (Sec. I.4).

IV.1.1 Slater determinants (SD)

The simplest fermionic wave function for many-electron systems is the Slater determinant (SD) wave function. Given a system of $N_e^{\uparrow}$ spin-up electrons and $N_e^{\downarrow}$ spin-down electrons, we can define the Slater determinant wave function as the product

$$\mathcal{F}(\mathbf{x}) = \det[\mathbf{S}^{\uparrow}]\det[\mathbf{S}^{\downarrow}], \tag{IV.2}$$

of two separate determinants (one for each spin population) of two square matrices $\mathbf{S}^{\uparrow}$ and $\mathbf{S}^{\downarrow}$ respectively of $N_e^{\uparrow} \times N_e^{\uparrow}$ and $N_e^{\downarrow} \times N_e^{\downarrow}$ elements, corresponding to the values of the molecular orbitals occupied by each electron

$$\mathbf{S}_{ij}^{\sigma} = \phi_i\left(\mathbf{r}_i^{\sigma}\right) = \sum_{q=1}^{Q} l_q^{(i)} \varphi_q\left(\mathbf{r}_j^{\sigma}\right) \tag{IV.3}$$

where $\sigma\ (z)$ is equal to $\uparrow$ or $\downarrow$. The set of coefficients $l_q^{(i)}$ can be grouped into a rectangular matrix $\mathbf{L}^{\sigma}$ of Q rows and N_e^{σ} columns. In order to be an eigenstate of the $\hat{S}^2$ operator the coefficients of the molecular orbitals for spin-up and spin-down electrons must be identical $\mathbf{L}^{\uparrow} = \mathbf{L}^{\downarrow}$. In this case, the wave function is defined as 'restricted' (`rsd`). The `rsd` flag in the wave function section of the basis set file is also used in the case of restricted open shell wave functions, which are used if the number of spin-up and spin-down electrons differ.

In the 'unrestricted' (`usd`) version of the Slater determinant $\mathbf{L}^{\uparrow} \neq \mathbf{L}^{\downarrow}$ introducing spin contamination. This might be necessary to describe correctly systems that present spin-localization such as diradical species.

IV.1.2 Antisymmetrized Geminal Power (AGP)

The Full Configuration Interaction (FCI) wave function for two electrons in a singlet state is written as a singlet geminal pairing function, defined as the linear combination of products of two atomic orbitals

$$\phi_G(\mathbf{r}^{\uparrow}, \mathbf{r}^{\downarrow}) = \sum_{q,p=1}^{Q} \lambda_{qp} \varphi_q(\mathbf{r}^{\uparrow}) \varphi_p(\mathbf{r}^{\downarrow}) \left|0, 0\right\rangle, \tag{IV.4}$$

which includes all the static correlation for two electrons systems, in a singlet spin state $|0,0\rangle = \frac{1}{2}\left(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle\right)$, where Q is the total number of atomic orbitals in the basis set. Since the geminal wave function describes two electrons in a singlet state, which is antisymmetric with respect to the spin coordinates, the spatial part of the wave function must be symmetric in order to be a pure eigenstate of the $\hat{S}^2$ operator, and thus $\lambda_{qp} = \lambda_{pq}$.
This symmetry reduces the number of independent parameters to $\leq Q(Q+1)/2$, where Q is the length of the atomic basis set, which can be further reduced by taking into account the spatial symmetries in the molecular system.
The antisymmetrized geminal power (AGP) wave function is defined as the determinant

$$\mathcal{F}(\bar{\mathbf{x}}) = \det[\mathbf{G}] \tag{IV.5}$$

of the geminal matrix $\mathbf{G}$ of $N_e^\uparrow \times N_e^\uparrow$ elements. For closed shell systems $(N_e^\uparrow = N_e^\downarrow)$ each element represents a geminal function (eq. IV.4):

$$\mathbf{G}_{ij} = \phi_G(\mathbf{r}_i^\uparrow, \mathbf{r}_j^\downarrow) \qquad \begin{matrix} i \in [1, N_e^\uparrow] \\ j \in [1, N_e^\downarrow] \end{matrix} . \tag{IV.6}$$

For a spin-polarized systems $(N_e^\uparrow > N_e^\downarrow)$ the more general solution can be extended by adding $N_e^u = N_e^\uparrow - N_e^\downarrow$ columns to the $\mathbf{G}$ matrix, containing unpaired molecular orbitals occupied by the spin-up electrons:

$$\mathbf{G}_{ij} = \phi_j(\mathbf{r}_i^\uparrow) = \sum_{q=1}^{Q} l_q^{(j)} \varphi_q(\mathbf{r}_i^\uparrow) \qquad \begin{matrix} i \in [1, N_e^\uparrow] \\ j \in [N_e^\downarrow + 1, N_e^\downarrow + N_e^u] \end{matrix} . \tag{IV.7}$$

The AGP wave function is a constraint multideterminantal expansion[16,17,18,19] thus containing the single Slater determinant in its variational space. The relationship between the AGP and the Slater determinant can be understood by considering that the geminal functions in eq. IV.4 with an appropriate transformation[20,21] can be rewritten as

$$\phi_G(\mathbf{r}_i^\uparrow, \mathbf{r}_j^\downarrow) = \sum_{p=1}^{P} \tilde{\lambda}_p \varphi_p(\mathbf{r}_i^\uparrow)\varphi_p(\mathbf{r}_j^\downarrow) \, |0,0\rangle \, , \tag{IV.8}$$

where $\varphi_p(\mathbf{r}) = \sum_{q=1}^{Q} l_q^p \psi_q(\mathbf{r})$ are a set of molecular orbitals (doubly occupied by the electronic pair in a singlet state) and $\tilde{\lambda}_p$ are the set of linear coefficients that weight the doubly occupied orbitals in the expansion. If we restrict the summation to the minimum number of doubly occupied molecular orbitals in the system, *ie* $N_e/2$, the geminal reduces to the sum $\phi_G(\mathbf{r}_i^\uparrow, \mathbf{r}_j^\downarrow) = \sum_{p=1}^{N_e/2} \varphi_p(\mathbf{r}_i^\uparrow)\varphi_p(\mathbf{r}_j^\downarrow) \, |0,0\rangle$, where all the weights are $\tilde{\lambda}_p = 1$. In this situation, we obtain a reduced geminal matrix as the matrix product $\mathbf{G}^* = \mathbf{S}^{\uparrow\top}\mathbf{S}^\downarrow$ between the square matrices $\mathbf{S}^\uparrow$ and $\mathbf{S}^\downarrow$ containing respectively the values of the $N_e/2$ molecular orbitals computed on the $N_e^\uparrow$ spin up and $N_e^\downarrow$ spin down electrons.

This reduced gemimal matrix corresponds to the Slater determinant wave function finally written as the product of the two determinants of $\mathbf{S}^\uparrow$ and $\mathbf{S}^\downarrow$:

$$\mathcal{F}(\mathbf{x}) = \det\left[\mathbf{S}^{\uparrow\top}\mathbf{S}^\downarrow\right] = \det\left[\mathbf{S}^\uparrow\right] \det\left[\mathbf{S}^\downarrow\right] . \tag{IV.9}$$

IV.1.3 The Pfaffian wave function

The Pfaffian, introduced by the mathematician Arthur Cayley[22] and named after Johann Friedrich Pfaff, is defined as the polynomial of the elements of a skew-symmetric matrix with an even number of elements per dimension (for a matrix with odd number of elements per dimension the Pfaffian is zero).

For a skew-symmetric matrix $\mathbf{P}$ with $2n \times 2n$ elements p_{ij}, we can define a set Π of $(2n-1)!!$ partitions of the $2n$ elements into pairs $\pi = [(i_1, j_1), (i_2, j_2), \ldots, (j_n, j_n)]$ with $i_k < j_k$ and $i_1 < i_2 < \ldots < i_n$[23].

The Pfaffian of the matrix $\mathbf{P}$ will be given by the sum over the products of the matrix elements organized according to the full set of possible partitions

$$\mathrm{pf}[\mathbf{P}] = \frac{1}{2^n n!} \sum_{\pi \in \Pi} \mathrm{sgn}(\pi) p_{i_1,j_1} p_{i_2,j_2} \ldots p_{i_n,j_n} \tag{IV.10}$$

each multiplied by the sign of the corresponding elements' permutation[24,20,23].

For example in the case of a 4×4 matrix[24,20] the Pfaffian corresponds to:

$$\mathrm{pf}\begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} \\ -p_{12} & 0 & p_{23} & p_{24} \\ -p_{13} & -p_{23} & 0 & p_{34} \\ -p_{14} & -p_{24} & -p_{34} & 0 \end{pmatrix} = \\ = \frac{1}{8}\left(p_{12}p_{34} - p_{13}p_{24} + p_{14}p_{23}\right).$$

To describe the Pfaffian wave function

$$\mathcal{F}(\mathbf{x}) = \mathrm{pf}[\mathbf{P}] \tag{IV.11}$$

for now let's consider a system with an odd number of electrons N_e [24,20].
Let us assume that the electrons are ordered with respect to their spin so that the first $N_e^{\uparrow}$ have spin up while the last $N_e^{\downarrow}$ have an opposite spin; by doing so we can partition the skew-symmetric matrix in four blocks

$$\mathbf{P} = \begin{pmatrix} \mathbf{T}^{\uparrow\uparrow} & \mathbf{G} \\ -\mathbf{G}^{\top} & \mathbf{T}^{\downarrow\downarrow} \end{pmatrix} \tag{IV.12}$$

the elements of each block describe the pairing of a subset of electrons in a fixed spin state.
Clearly, the elements of the block matrix **G** are the geminal functions defined in (eq. IV.4).
On the other hand, the elements of the $\mathbf{T}^{\uparrow\uparrow}$ and $\mathbf{T}^{\downarrow\downarrow}$ block matrices describe the coupling between electrons with parallel spins in a triplet state, and are defined through the linear combinations of products of two atomic orbitals $\psi_q(\mathbf{x})$ modulated by a set of coupling coefficients $\zeta_{qp}^{\uparrow\uparrow}$ and $\zeta_{qp}^{\downarrow\downarrow}$

$$\mathbf{T}_{ij}^{\uparrow\uparrow} = \phi_T(\mathbf{r}_i^{\uparrow}, \mathbf{r}_j^{\uparrow}) = \sum_{q,p=1}^{Q} \zeta_{qp}^{\uparrow\uparrow}\psi_q(\mathbf{r}_i^{\uparrow})\psi_p(\mathbf{r}_j^{\uparrow})\,|1,1\rangle \tag{IV.13}$$

$$\mathbf{T}_{ij}^{\downarrow\downarrow} = \phi_T(\mathbf{r}_i^{\downarrow}, \mathbf{r}_j^{\downarrow}) = \sum_{q,p=1}^{Q} \zeta_{qp}^{\downarrow\downarrow}\psi_q(\mathbf{r}_i^{\downarrow})\psi_p(\mathbf{r}_j^{\downarrow})\,|1,-1\rangle \tag{IV.14}$$

where the indexes q and p run on the full set of basis set orbitals Q, and $|1,1\rangle$ and $|1,-1\rangle$ correspond respectively to the triplet spin states $\left|\frac{1}{2},\frac{1}{2}\right\rangle\left|\frac{1}{2},\frac{1}{2}\right\rangle$ and $\left|\frac{1}{2},-\frac{1}{2}\right\rangle\left|\frac{1}{2},-\frac{1}{2}\right\rangle$. In order to fix the spin symmetry, the spatial part of the geminal functions must be antisymmetric with respect to the exchange of the electronic coordinates thus the conditions that $\zeta_{qp}^{\uparrow\uparrow} = -\zeta_{pq}^{\uparrow\uparrow}$ and $\zeta_{qp}^{\downarrow\downarrow} = -\zeta_{pq}^{\downarrow\downarrow}$ must always hold: Moreover, since two electrons with parallel spin cannot occupy the same orbital, it must be that $\zeta_{qp}^{\uparrow\uparrow} = 0$ and $\zeta_{qp}^{\uparrow\uparrow} = 0$. Because of these two conditions it is clear that the set of coefficients $\zeta_{qp}^{\uparrow\uparrow}$ and $\zeta_{qp}^{\downarrow\downarrow}$ form two skew-symmetric matrices $\mathcal{Z}^{\uparrow}$ and $\mathcal{Z}^{\downarrow}$.
In the case in which the number of electrons in the system is odd, we can extend the Pfaffian wave function by adding a column (and a row) of elements, representing the electrons in an unpaired molecular orbital[24,20]. Within this generalization, the Pfaffian matrix will be defined as:

$$\mathbf{P} = \begin{pmatrix} \mathbf{T}^{\uparrow\uparrow} & \mathbf{G} & \bar{\phi}_{\uparrow} \\ -\mathbf{G}^{\top} & \mathbf{T}^{\downarrow\downarrow} & \bar{\phi}_{\downarrow} \\ -\bar{\phi}_{\uparrow}^{\top} & -\bar{\phi}_{\downarrow}^{\top} & 0 \end{pmatrix}, \tag{IV.15}$$

where $\mathbf{T}^{\uparrow\uparrow}$, $\mathbf{T}^{\downarrow\downarrow}$ and **G** are the matrices defined above, while the elements of the last column (or row) vector

$$\bar{\phi}_i = \begin{pmatrix} \bar{\phi}^{\uparrow} \\ \bar{\phi}^{\downarrow} \end{pmatrix}_i = \varphi(\mathbf{r}_i) = \sum_{q}^{Q} l_q \psi_q(\mathbf{r}_i) \qquad i \in N_e \tag{IV.16}$$

represent the values of a molecular orbital, defined through the variational parameters l_q, occupied by each of the N_e electron in the system.

IV.1.4 Triplet Geminal Power (TGP)

The Pfaffian wave function and antisymmetrized geminal power (AGP) wave function[25,26,27] are related. This can be understood by considering that for an even number of electrons, by removing in the **P** matrix in eq. IV.12 the coupling between the electrons with parallel spin ($\mathcal{Z}^{\uparrow} = \mathcal{Z}^{\downarrow} = 0$), the Pfaffian reduces to the determinant of the geminal matrix **G** [24,20],

$$\mathcal{F}(\mathbf{x}) = \mathrm{pf}\begin{pmatrix} 0 & \mathbf{G} \\ -\mathbf{G}^{\top} & 0 \end{pmatrix} = \det\left[\mathbf{G}\right], \tag{IV.17}$$

which corresponds to the AGP wave function[25,26,27] for closed shell systems.
In the same way, if one suppresses the correlation between different spin electrons $\Lambda = 0$, the singlet geminal matrices will be zero and the full Pfaffian will reduce to the product of two smaller Pfaffian matrices:

$$\mathcal{F}(\mathbf{x}) = \mathrm{pf}\begin{pmatrix} \mathbf{T}^{\uparrow\uparrow} & 0 \\ 0 & \mathbf{T}^{\downarrow\downarrow} \end{pmatrix} = \mathrm{pf}[\mathbf{T}^{\uparrow\uparrow}]\mathrm{pf}[\mathbf{T}^{\downarrow\downarrow}], \tag{IV.18}$$

We can thus define the antisymmetrized triplet geminal power (TGP) wave function as the product of the Pfaffians associated with the populations of spin-up and spin-down electrons

$$\mathcal{F}(\mathbf{x}) = \mathrm{pf}[\mathbf{T}^{\uparrow\uparrow}]\mathrm{pf}[\mathbf{T}^{\downarrow\downarrow}]. \tag{IV.19}$$

The two triplet geminal matrices $\mathbf{T}^{\uparrow}$ and $\mathbf{T}^{\downarrow}$ of respectively $N_e^{\uparrow 2}$ and $N_e^{\downarrow 2}$ elements are skew-symmetric and are in general defined as the matrix products and for all open-shell systems we must point out that the coefficients of the two matrices $\mathcal{Z}^{\uparrow} \neq \mathcal{Z}^{\downarrow}$ must be different. Thus the wave function in general is not an eigenstate of the $\hat{S}^2$ operator.

IV.2 Electronic Jastrow factor

The bosonic Jastrow factor that includes the many-body interactions of the quantum Monte Carlo wave functions is the exponent of the linear combination of various terms. In general, we can separate the Jastrow factor into four terms

$$\mathcal{J}(\mathbf{x}) = e^{\mathrm{J}(\mathbf{x})} = e^{\mathrm{J}_c^{1b}(\mathbf{r})+\mathrm{J}_c^{2b}(\mathbf{x})+\mathrm{J}_d^{1b}(\mathbf{r})+\mathrm{J}_d^{2b}(\mathbf{r})}, \tag{IV.20}$$

that are respectively the one-body term describing the electron-nucleus cusp, the homogeneous two-body term describing the electron-electron cusps, the dynamical one-body term that is the linear combination of non-normalized atomic orbitals, and finally, the dynamical two-body term describing the correlation between electronic pairs in the field of the nuclei.
In the following sections, we will describe the basic functional forms of the Jastrow factor in the QMeCha code and its implementations.

IV.2.1 Electron-Nucleus cusp condition

The first one-body term describes the electron-nucleus cusp condition, and its most general form will be written as the sum of $N_e \times M_n$ terms:

$$\mathrm{J}_c^{1b}(\mathbf{r}) = \sum_{i=1}^{N_e}\sum_{a=1}^{M_n} \mathcal{F}_a(r_{ia}), \tag{IV.21}$$

and will depend only on a set of functions of the electron-nucleus distances. The functions $\mathcal{F}_a(r_{ia})$ only depend on the atom on which they are centred and are spherically symmetrical. In the QMeCha code, the variational parameters are linked automatically for symmetry of identical atoms and can assume three main forms. The first two types of cusp expansions are written as

$$\mathcal{F}_a(r_{ia}) = Z_a f_a(r_{ia}) + \sum_{m=1}^{M} \beta_m e^{-\zeta_m r_{ia}^2}. \tag{IV.22}$$

The cusp functions $f_a(r_{ia})$ are explicitly dependent on the nuclear charge of the ath nucleus (used as a rescaling factor) to which they are associated and are written as:

$$f_a(r_{ia}) = \begin{cases} \frac{1}{\mathcal{B}_a\mathcal{A}_a} e^{-\mathcal{B}_a\mathcal{A}_a r_{ia}} \\ \frac{1}{\mathcal{B}_a\mathcal{A}_a(1+\mathcal{B}_a\mathcal{A}_a r_{ia})} \end{cases}. \tag{IV.23}$$

where $\mathcal{A}_a = (2Z_a)^{1/4}$ is a constant factor and $\mathcal{B}_a$ is a variational parameter that must be optimized.
The linear combination of Gaussian functions is used as a dynamical one-body term that can be set to zero by setting $M = 0$ in the basis set file (see Tab. III.2)
The last cusp function is the one proposed by Drummond, Towler and Needs (DTN)[13]

$$\mathcal{F}_a(r_{ia}) = (r_{ia} - \mathcal{B}_0)^3\, \Theta\left(\mathcal{B}_0 - r_{ia}\right) \mathcal{P}_n(r_{ia}; \bar{\mathcal{B}}) \tag{IV.24}$$

where $\bar{\mathcal{B}} = (\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_M)$ is the set of variational parameters and

$$\mathcal{P}_m(r_{ia}; \bar{\mathcal{B}}) = \mathcal{B}_1 + \left(\frac{3\mathcal{B}_1}{\mathcal{B}_0} + \frac{Z_a}{\mathcal{B}_0^3}\right) r_{ia} + \sum_{m=2}^{M} \mathcal{B}_n r_{ia}^m \tag{IV.25}$$

is the polynomial of order M.

IV.2.2 Electron-Electron cusp condition

The part of the two body Jastrow factor necessary for describing the electron-electron cusp condition is written as a sum over the electronic pairs:

$$\mathrm{J}_c^{2b}(\bar{\mathbf{r}}) = \sum_{j>i=1}^{N_e} \mathcal{F}(\mathbf{r}_{ij}), \tag{IV.26}$$

defined through a function of the only electron-electron distance and satisfying the boundary conditions of going to zero as $r_{ij} \to \infty$ and to a constant value as the two electrons approach:

$$\left.\frac{\partial \mathcal{F}(r_{ij})}{\partial r_{ij}}\right|_{r_{ij}=0} = q_i q_j \mu_{ij} \begin{cases} \frac{2}{d+1} & \text{indistinguishable particles} \\ \frac{2}{d-1} & \text{distinguishable particles} \end{cases} \tag{IV.27}$$

where q_i and q_j are the charges of the two particles, μ_{ij} is their reduced mass (that for fermions is always equal to $\mu = 1/2$) and d is the dimensions of the system (in QMeCha it is always $d = 3$). It must be underlined that by introducing this difference in the electronic cusp conditions of parallel and antiparallel electrons, the variance of the wave function tends to diminish, but a small spin contamination is introduced in the wave function[28]. To diminish this effect it is best to introduce independent variational parameters for the two body terms of parallel and antiparallel electron pairs.
The first two types of electron-electron cusp functions are of the form

$$\mathcal{F}(r_{ij}) = A_{ij} f(r_{ij}) + \sum_{m=1}^{M} \beta_m e^{-\zeta_m r_{ij}^2} \tag{IV.28}$$

where $A_{ij} = \frac{2 q_i q_j \mu_{ij}}{d \pm 1}$ is the constant that depends on the distinguishability (or not) of the fermionic pair. The $f(r_{ij})$ functions can have two forms which present a fast (Becker, Broyles and Tucson[29]) and slow decay (Fahy, Wang and Louie[30])

$$f(r_{ij}) = \begin{cases} -\frac{1}{b} e^{-b r_{ij}} \\ -\frac{1}{b(1+b r_{ij})} \end{cases} . \tag{IV.29}$$

As for the nuclear cusps also the two-body is augmented by a linear combination of Gaussian-type functions that depend only on the two-particle distances.
The third type of cusp is again the one proposed by Drummond, Towler and Needs (DTN)[13], that, in order to guarantee continuity of both the gradient and Laplacian, is defined as

$$\mathcal{F}(r_{ij}) = (r_{ij} - \alpha_0)^3 \, \Theta(\alpha_0 - r_{ij}) \, \mathcal{P}_n(r_{ij}; \bar{\alpha}), \tag{IV.30}$$

where $\bar{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_N)$ is the set of variational parameters and

$$\mathcal{P}_n(r_{ij}; \bar{\alpha}) = \alpha_1 + \left(\frac{3\alpha_1}{\alpha_0} - \frac{\Gamma_{\sigma_i,\sigma_j}}{\alpha_0^3} \right) r_{ij} + \sum_{n=2}^{N} \alpha_n r_{ij}^n \tag{IV.31}$$

is the polynomial of order N in which $\Gamma_{\sigma_i,\sigma_j}$ is the spin prefactor defined as

$$\Gamma_{\sigma_i,\sigma_j} = \begin{cases} \frac{1}{4} & \text{for} \quad \sigma_i = \sigma_j \\ \frac{1}{2} & \text{for} \quad \sigma_i = -\sigma_j \end{cases} .$$

In particular, α_0 is a cut-off parameter that defines the radius of action of the Jastrow factor that is zero outside that radius.

IV.2.3 Jastrow factor for Dynamical correlation

The dynamical Jastrow factor is used to recover the non-homogeneous correlation between the electronic pairs in the field of the external nuclei.
Following the procedure introduced by Prof. Sorella in his TurboRVB code, the one body dynamical part is defined as the linear combination of non-normalized atomic orbitals $\chi_q(\mathbf{r})$:

$$\mathrm{J}_d^{1b}(\mathbf{r}) = \sum_{i=1}^{N_e} \sum_{q=1}^{Q} g_q \chi_q(\mathbf{r}_i). \tag{IV.32}$$

It is important to underline that if the Q_m function is used inside the electron-nucleus cusp, this term should be avoided because redundant.

The Three-Four body Jastrow factor is derived from the construction of the geminal function of electron couples, as introduced by prof. Sorella and as previously discussed for the Fermionic part of the wave function[26]. It can be written as the sum:

$$J_d^{2b}(\mathbf{r}) = \sum_{j>i=1}^{N_e} \sum_{q,p=1}^{Q} g_{qp}\chi_q(\mathbf{r}_i)\chi_p(\mathbf{r}_j), \tag{IV.33}$$

with the condition that $g_{qp} = g_{pq}$ in order to guarantee that the term is symmetric with respect to the exchange of the electronic coordinates.

IV.3 Electron-Positron wave function

The most general expression for many-electrons and a positron wave function $\Psi(\mathbf{x}^e, \mathbf{x}^p; \mathbf{R})$ explicitly describes the many-body correlation effects between the $4N_e$ electronic $textbf x^e$ and positronic $\mathbf{x}^p$ Cartesian and spin coordinates in the field of the nuclei $\mathbf{R}$.

A first approximation to this fully correlated state can be built by considering only the explicit correlation between particle pairs. The wave function is thus built as a symmetrized product (or a linear combination of symmetrized products) of two-particle functions, as proposed for example by Bressanini *et al.* in ref. 31, describing the correlation between electron-electron, electron-positron, nucleus-electron and nucleus-positron pairs. Clearly, this ansatz, although very accurate, is more computationally expensive when applied to large systems of many atoms and many positrons.

A way to further simplify the total wave function is to decouple the electronic and positronic degrees of freedom by writing the wave function as the product

$$\Psi(\mathbf{x}^e, \mathbf{x}^p; \mathbf{R}) = \psi_e(\mathbf{x}^e; \mathbf{R})\psi_p(\mathbf{x}^p; \mathbf{R})J(\mathbf{x}^e, \mathbf{x}^p; \mathbf{R}) \tag{IV.34}$$

of an electronic $\psi_e(\mathbf{x}^e; \bar{\mathbf{R}})$ (such as a Slater determinant) and a positronic $\psi_p(\mathbf{x}^p, \bar{\mathbf{R}})$ wave function, times a bosonic Jastrow factor that describes the correlation between the remaining particle pairs, also including three or four body correlation effects.

Assuming that the electronic wave function $\psi_e(\bar{\mathbf{x}}^e; \bar{\mathbf{R}})$ describes the spin and angular symmetries of the electrons in the field of the nuclei, the general positronic function $\psi_p(\mathbf{x}^p; \bar{\mathbf{R}})$ will also only depend on the nuclear coordinates, and can be written as a Slater determinant or Pfaffian[32,33,34,35,36] expanded in atomic orbitals.

Yet, it is important to remember that the positrons do not bind with the nuclei, as do the electrons, but will be attracted by the electronic molecular cloud. For this reason, a more general positronic wave function should also depend explicitly on the electronic coordinates.

In the next section, we will discuss such a case.

IV.4 Positronic wave function

As previously introduced, while the positron forms bound states with the electrons to which it is attracted, it does not form bound states with the atomic nuclei that repel it.

One way to consider this physical behavior in the positronic wave function is to construct the positron's orbital through a positronic basis set[31,37] explicitly describing the bound states between electron-positron pairs.

In fact, it can be easily shown that the ground state of a system of one electron and one positron, *i.e.* the Positronium (Ps), can be exactly described by an exponential function of the electron-positron distance $r^{ep} = |\mathbf{x}^e - \mathbf{x}^p|$:

$$\phi\left(\mathbf{x}^{ep}\right) = r^{ep} R\left(r^{ep}\right) Y_l^m(\theta^{ep}, \phi^{ep}). \tag{IV.35}$$

where $R\left(r^{ep}\right)$ is a radial function normalized with respect to the distance r^{ep} and $Y_l^m(\theta^{ep}, \phi^{ep})$ is a real spherical harmonic (centered on the positron) that is used to introduce an angular momentum.[38]

Through this basis, we can construct a positronic wave function for many electrons and one positron as the product:

$$\psi_p(\mathbf{x}^p; \bar{\mathbf{x}}^e) = \prod_{i=1}^{N_e} \varphi_p(\mathbf{r}_i^{ep}) \tag{IV.36}$$

of identical orbitals (so that the function is symmetric with respect to the exchange of the electronic coordinates), each dependent on the electron-positrons distance $\mathbf{r}_i^{ep}$, thus re-

ferred to as electron-positron orbitals (EPO), that are defined as linear combinations

$$\varphi_p(\mathbf{r}^{ep}) = \sum_{q=1}^{Q} l_q \phi_q(\mathbf{r}^{ep}) \tag{IV.37}$$

of the newly defined positronic orbitals.

IV.5 Electron-Positron Jastrow factor

The bosonic Jastrow factor[13] to describe the explicit correlation between electrons and positrons can be constructed starting from the electronic one described in Sec. IV.2, with the addition of three contributions

$$J(\mathbf{x}^e, \mathbf{x}^p; \mathbf{R}) = \exp\left\{ \mathrm{J}_c^{1b}(\mathbf{r}^p) + \mathrm{J}_c^{2b}(\mathbf{x}^p) + \right. \\ \left. + \mathrm{J}_c^{2b}(\mathbf{x}^e, \mathbf{x}^p) + \mathrm{J}_d^{2b}(\mathbf{r}^e, \mathbf{r}^p; \mathbf{R}) \right\}. \tag{IV.38}$$

which can be classified as one-body terms $\mathrm{J}_c^{1b}(\mathbf{r}^p)$ that are used to describe the positron-nucleus cusps conditions, pure homogeneous two-body terms $\mathrm{J}_c^{2b}(\mathbf{x}^p)$ and $\mathrm{J}_c^{2b}(\mathbf{x}^e, \mathbf{x}^p)$, that describe the pair correlations between positron-positron and electron-positron pairs, and finally a many-body (or inhomogeneous) term $\mathrm{J}_d^{2b}(\mathbf{r}^e, \mathbf{r}^p; \mathbf{R})$ that is used to describe the dynamical correlation between electrons and positrons in the field of the nuclei.

The One and Two body factors are similar to the one discussed previously for the electrons in Secs. IV.2.1 and IV.2.2, so we will not discuss them further here.

The 3/4 body term, on the other hand, follows the form proposed to describe the correlation between electronic pairs[27] with the addition of electron-positron and positron-positron contributions that use different linear combinations of the same Jastrow atomic basis set

$$\mathrm{J}_d^{2b}(\mathbf{r}^e, \mathbf{r}^p; \mathbf{R}) = \sum_{j>i=1}^{N_p} \sum_{q,p=1}^{\mathcal{Q}} \eta_{qp} \chi_q(\mathbf{r}_i^p) \chi_p(\mathbf{r}_j^p) + \sum_{i=1}^{N_e} \sum_{j=1}^{N_p} \sum_{q,p=1}^{\mathcal{Q}} \nu_{qp} \chi_q(\mathbf{r}_i^e) \chi_q(\mathbf{r}_j^p). \tag{IV.39}$$

Here $\chi_q(\mathbf{r})$ and $\chi_q(\mathbf{r})$ are a set of atomic orbitals and η_{qp} and ν_{qp} are a set of coefficients that are fully optimized. Since the Jastrow factor must be symmetric with respect to the exchange of all the electrons, the η_{qp} parameters satisfy the condition $\eta_{qp} = \eta_{pq}$.

IV.6 Approximate wavefunction for the quantum Drude oscillators

At long distances,[39,40,41,42] the solution of the CQDO model can be written as a perturbation of the solutions of two non-interacting QDOs, represented by isotropic Gaussian orbitals which are deformed by additional couplings.

For example, in the CI approach[39] the variational space of the homogeneous CQDO dimer was constructed as the linear combination of a set of single QDO excitations of the two localized oscillators. On the other hand, in the QMC approach[43,40] the trial wave function was built from the product of the solution of non-interacting harmonic oscillators adding a fixed three-body correlation function constructed analytically to reproduce the long-range interactions between the QDOs that are responsible for the reciprocal polarization.

In this code we have a more general form[42], inspired by the exact solution of the QDO model in the dipole interaction limit (eq. **??**)

$$\Psi^{\text{Dip.}}(\bar{\mathbf{r}}) = \exp\{-\bar{\mathbf{d}}^\top \mathbf{A} \bar{\mathbf{d}}\}, \tag{IV.40}$$

where $\mathbf{A}$ is a symmetric $3N \times 3N$ coupling matrix and $\bar{\mathbf{d}}$ is the $3N$ vector of the distances between the drudons and their corresponding centers, *ie* $\{\mathbf{d}_i = \mathbf{r}_i - \mathbf{R}_i \quad \forall \quad i \in [1, N]\}$.[44,42] While the diagonal elements of the parameters' matrix $\mathbf{A}$ contain the Gaussian solutions of the isolated QDOs, the off-diagonal elements represent the coupling of the various oscillators along the three dimensions.

Despite the fact that for distances characteristic of the dispersion interactions this solution is a good approximation, it fails to converge to the proper state for the CQDO model for all values of the QDO positions $\bar{\mathbf{R}}$ since it does not take into account the form of the single-particle potential energies.

In order to construct a more general and accurate solution we should consider that the ith drudon interacts with its center through a harmonic potential, while it interacts with all the other $N-1$ QDO centers via a Coulomb potential, meaning that the wave func-

tion must satisfy the cusp conditions[45]:

$$\frac{1}{\langle\Psi\rangle}\frac{\partial\langle\Psi\rangle}{\partial d_{ij}}\bigg|_{d_{ij}=0} = q_i q_j \mu_j \quad \forall \quad j \neq i, \tag{IV.41}$$

where $\langle\Psi\rangle$ represents the angular average of the wavefunction and $d_{ij} = |\mathbf{r}_i - \mathbf{R}_j|$ is the absolute distance between the ith drudon and the jth QDO center. Thus, around the jth center the single-particle orbital of the ith drudon will have the form of an exponential function, *ie* a Slater type orbital (STO), proportional to $\varphi_j(\mathbf{r}_i) \propto \exp\left(-q_i q_j \mu_j d_{ij}\right)$.
Considering these characteristics, approximate single particle orbitals should be written as a linear combination of Gaussian and Slater-type functions (see Fig. **??**)

$$\Phi^i(\mathbf{r}_i) \approx \alpha_i^i e^{(-\frac{\mu_i\omega_i}{2}d_i^2)} + \sum_{j\neq i}^{N} \alpha_j^i e^{(-q_i q_j \mu_j d_{ij})}, \tag{IV.42}$$

where α_j^i are a set of parameters, with indices $i, j \in [1, N]$. These orbitals can be generalized following the same approach used to define electronic molecular orbitals, and thus in this work, they are written as a linear combination

$$\Phi^i(\mathbf{r}_i) = \sum_{q=1}^{Q} \alpha_q^i \phi_q(\mathbf{r}_i) \tag{IV.43}$$

of Q atomic-like orbitals $\phi_q(\mathbf{r}_i)$ constituting the basis set of the system of QDOs, and are written as a linear combination of Gaussian type orbitals GTO centered on the positions of the various QDOs $\bar{\mathbf{R}}$, whose linear and exponential parameters are fully optimized. In particular, in all our calculations each QDO is represented through a (3s1s)/[1s1s] contracted basis set, where the uncontracted 1s orbital is used to describe the Gaussian solution around the quadratic potential and the contracted (3s)/[1s] orbital is used to describe the Slater solution centered on the drudon–nucleus Coulomb potential.
The use of GTOs in place of exponential functions clearly introduces an error in the description of the one-body cusp in eq. IV.41 that can be eliminated through a Jastrow factor[46,13] that is normally used in quantum Monte Carlo trial wavefunctions, of the form

$$J_1(\bar{\mathbf{r}}) = \exp\left(\sum_{i\neq j}^{N} q_i q_j \mu_j f_{1b}(d_{ij};\boldsymbol{\gamma})\right), \tag{IV.44}$$

where $f_{1b}(d_{ij};\boldsymbol{\gamma})$ is a parametric function that only depends on the distance between two particles (in this case the ith drudon and the jth QDO center) that for $d_{ij} \to 0$ has the property of going linearly to zero, *ie* $f(d_{ij}) \approx d_{ij}$, and decays to zero as the distance increases $f(d_{ij}) \to 0$ as $d_{ij} \to \infty$.
Another property of the wavefunction will be the explicit correlation between the pairs of drudons, which is introduced through two-body Coulomb potential $v_{ij}(\mathbf{r}_i, \mathbf{r}_j)$ in eq. **??**. A first correlation function between drudonic pairs can be introduced considering that in the limit of two overlapping drudons the exact wavefunction has a two-body cusp of the form:

$$\frac{1}{\langle\Psi\rangle}\frac{\partial\langle\Psi\rangle}{\partial r_{ij}}\bigg|_{r_{ij}=0} = q_i q_j \frac{\mu_i\mu_j}{\mu_i+\mu_j} \quad \forall \quad j \neq i. \tag{IV.45}$$

This requisite can be satisfied again through a Jastrow factor of the type

$$J_2(\bar{\mathbf{r}}) = \exp\left(\sum_{i>j}^{N} q_i q_j \frac{\mu_i\mu_j}{\mu_i+\mu_j} f_{2b}(r_{ij};\boldsymbol{\eta})\right), \tag{IV.46}$$

where $f_{2b}(r_{ij};\boldsymbol{\eta})$ is a parametric function with similar properties to the one defined in eq. IV.44 to reproduce the one-body cusp condition.
From these considerations, by combining eqs. IV.43, IV.44 and IV.46 we can write a first approximation to the explicitly correlated wavefunction as

$$\Psi_T^{\text{Mol+J}}(\bar{\mathbf{r}}) = \left[\prod_{i=1}^{N} \Phi^i(\mathbf{r}_i)\right] J_1(\bar{\mathbf{r}}) J_2(\bar{\mathbf{r}}), \tag{IV.47}$$

that will be referred to as Mol.+J in the next sections, where the one-body cusp function is written as

$$f_{1b}(d_{ij};\boldsymbol{\gamma}) = -\frac{e^{-\gamma_0 d_{ij}}}{\gamma_0} + \sum_{m=1}^{M} \gamma_m e^{-\gamma_{M+m} d_{ij}^2} \tag{IV.48}$$

and the two-body has a form inspired by Padé's approximation[47]

$$f_{2b}(r_{ij};\boldsymbol{\eta}) = -\frac{1}{\eta_0(1+\eta_0 r_{ij})} + \sum_{m=1}^{M} \eta_m e^{-\eta_{M+m} r_{ij}^2}. \tag{IV.49}$$

In eqs. IV.48 and IV.49, the two vectors of $2M+1$ parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$ are both optimized. In our calculations, we assume $M = 5$.
In order to assess approximate wavefunctions, two alternative trial wavefunctions will be considered in this work: (i) dipolar wavefunction in eq. IV.40, referred to as Dip.; (ii) the same dipolar wavefunction enhanced with the Jastrow factors defined in eqs. IV.44 and IV.46:

$$\Psi_T^{\text{Dip.+J}}(\bar{\mathbf{r}}) = \exp\{-\bar{\mathbf{d}}^\top \mathbf{A}\bar{\mathbf{d}}\} J_1(\bar{\mathbf{r}}) J_2(\bar{\mathbf{r}}), \tag{IV.50}$$

that will be referred to as Dip.+J.

IV.7 Atomic orbitals

We build the atomic orbital basis set as the product of a radial part, that only depends on the distance between the electron and the nucleus $\mathbf{r}_a$, on which the basis set is centred, and a cubic Cartesian harmonics:

$$\varphi(\mathbf{r}_a) = r_a^l R(r_a) Z_{l,m}(x_a, y_a, z_a) = R(r_a) \bar{Z}_{l,m}(x_a, y_a, z_a) \tag{IV.51}$$

where $\bar{Z}_{l,m}(x_a, y_a, z_a) = r_a^l Z_{l,m}(x_a, y_a, z_a)$ are 'reduced' cubic harmonics.
For a general orbital written as

$$\varphi(\mathbf{r}_a) = r_a^l R(r_a) Z_{l,m}(x_a, y_a, z_a). \tag{IV.52}$$

the radial part of the wave function can be written as a linear combination of p different primitive functions.

$$R(r_a) = \sum_{k=0}^{p} c_k P_p(\zeta_k, r_a). \tag{IV.53}$$

IV.7.1 Gaussian type orbitals

The most commonly used primitive functions are those based on Gaussian functions, due to the fact that these functions and their products are easy to integrate analytically. For these primitives the exponential function is defined as $\alpha(\zeta, r_a) = -\zeta r_a^2$ so that we have

$$P_n(r_a) = \mathcal{N}_{n,l}(\zeta) r_a^n e^{-\zeta r_a^2}. \tag{IV.54}$$

with the normalization factor

$$\mathcal{N}_{n,l}(\zeta) = \left(\frac{2^{4\bar{n}+7}}{\pi}\right)^{1/4} \frac{1}{\sqrt{(2\bar{n}+1)!!}} \zeta^{\frac{2\bar{n}+3}{4}}. \tag{IV.55}$$

in which $\bar{n} = n + l$.

IV.7.2 Slater type orbitals

The simplest form for the primitive radial functions comes from the eigenstates of the hydrogen atom, that are exponential functions multiplied by a polynomial:

$$P_n(r_a) = \mathcal{N}_{n,l}(\zeta) r_a^n e^{-\zeta r_a}, \tag{IV.56}$$

where the normalization factor will be equal to

$$\mathcal{N}_{n,l}(\zeta) = \frac{2^{\bar{n}+3/2}}{\sqrt{(2\bar{n}+2)!}} \zeta^{\bar{n}+3/2}. \tag{IV.57}$$

with again $\bar{n} = n + l$.

IV.7.3 Slater-Gaussian type orbitals

In order to describe the exponential behaviour of s orbital shells, without including the nuclear cusp, we can use the mixed Slater-Gaussian functions introduced in ref. 14,15 which are defined through the exponential of the function $\alpha(\zeta, r_a) = -\frac{(\zeta r_a)^2}{1+\zeta r_a}$.
Here we generalize these functions by adding the parameter η that defines the switch between the Gaussian and exponential type of function $\alpha(\zeta, \eta, r_a) = -\frac{(\zeta r_a)^2}{\eta+\zeta r_a}$. We thus define the primitives as

$$P_n(r_a) = \mathcal{N}_{n,l}(\eta, \zeta) r_a^n e^{-\frac{(\zeta r_a)^2}{\eta+\zeta r_a}}. \tag{IV.58}$$

Table IV.1: *Reduced Cubic Harmonics and their gradients*

l	m	$r^l Z_{l,m}(\theta,\phi)$	$\bar{\nabla}_{\mathbf{r}}\left[r^l Z_{l,m}(\theta,\phi)\right]$
0	0	$\sqrt{\frac{1}{4\pi}}$	$[0,0,0]$
1	-1	$\sqrt{\frac{3}{4\pi}}y$	$[0,1,0]$
1	0	$\sqrt{\frac{3}{4\pi}}z$	$[0,0,1]$
1	+1	$\sqrt{\frac{3}{4\pi}}x$	$[1,0,0]$
2	-2	$\sqrt{\frac{15}{4\pi}}xy$	$[y,x,0]$
2	-1	$\sqrt{\frac{15}{4\pi}}yz$	$[0,z,y]$
2	0	$\sqrt{\frac{5}{16\pi}}(2z^2-x^2-y^2)$	$[-2x,-2y,4z]$
2	+1	$\sqrt{\frac{15}{4\pi}}xz$	$[z,0,x]$
2	+2	$\sqrt{\frac{15}{16\pi}}(x^2-y^2)$	$[2x,-2y,0]$
3	-3	$\sqrt{\frac{35}{32\pi}}(3x^2-y^2)y$	$[6xy,3(x^2-y^2),0]$
3	-2	$\sqrt{\frac{105}{4\pi}}xyz$	$[yz,xz,xy]$
3	-1	$\sqrt{\frac{21}{32\pi}}y(4z^2-x^2-y^2)$	$[-2xy,-3y^2+4z^2-x^2,8yz]$
3	0	$\sqrt{\frac{7}{16\pi}}z(2z^2-3x^2-3y^2)$	$[-6xz,-6yz,3(2z^2-y^2-x^2)]$
3	1	$\sqrt{\frac{21}{32\pi}}x(4z^2-x^2-y^2)$	$[-3x^2+4z^2-y^2,-2xy,8xz]$
3	2	$\sqrt{\frac{105}{16\pi}}(x^2-y^2)z$	$[2xz,-2yz,x^2-y^2]$
3	3	$\sqrt{\frac{35}{32\pi}}(x^2-3y^2)x$	$[3(x^2-y^2),-6xy,0]$
4	-4	$\frac{3}{4}\sqrt{\frac{35}{\pi}}xy(x^2-y^2)$	$[y(3x^2-y^2),x(x^2-3y^2),0]$
4	-3	$\frac{3}{4}\sqrt{\frac{35}{2\pi}}yz(3x^2-y^2)$	$[6xyz,3z(x^2-y^2),y(3x^2-y^2)]$
4	-2	$\frac{3}{4}\sqrt{\frac{5}{\pi}}xy(6z^2-x^2-y^2)$	$[y(6z^2-3x^2-y^2),x(6z^2-3y^2-x^2),12xyz]$
4	-1	$\frac{3}{4}\sqrt{\frac{5}{2\pi}}yz(4z^2-3x^2-3y^2)$	$[-6xyz,z(4z^2-9y^2-3x^2),3y(4z^2-y^2-x^2)]$
4	0	$\frac{3}{16}\sqrt{\frac{1}{\pi}}(8z^4+3x^4+3y^4+6x^2y^2-24x^2z^2-24y^2z^2)$	$[12x(x^2+y^2-4z^2),12y(x^2+y^2-4z^2),8(4z^2-6x^2-6y^2)]$
4	1	$\frac{3}{4}\sqrt{\frac{5}{2\pi}}xz(4z^2-3x^2-3y^2)$	$[z(4z^2-9x^2-3y^2),-6xyz,3x(4z^2-y^2-x^2)]$
4	2	$\frac{3}{8}\sqrt{\frac{5}{\pi}}(x^2-y^2)(6z^2-x^2-y^2)$	$[4x(3z^2-x^2),4y(y^2-3z^2),12z(x^2-y^2]$
4	3	$\frac{3}{4}\sqrt{\frac{35}{2\pi}}(x^2-3y^2)xz$	$[3z(x^2-y^2),-6xyz,x(x^2-3y^2)]$
4	4	$\frac{3}{16}\sqrt{\frac{35}{\pi}}(x^4-6x^2y^2+y^4)$	$[4x(x^2-3y^2),4y(y^2-3x^2),0]$

The normalization of these orbitals can be obtained through a scaling relation[14,15] for which:

$$\mathcal{N}_{n,l}^{-2}(\eta,\zeta)=\int_0^\infty r_a^{2(\bar{n}+1)}e^{-2\frac{(\zeta r_a)^2}{\eta+\zeta r_a}}dr_a= \\ =\frac{1}{\zeta^{2\bar{n}+3}}\int_0^\infty u_a^{2(\bar{n}+1)}e^{-2\frac{u_a^2}{\eta+u_a}}du_a$$

thus:

$$\mathcal{N}_{n,l}(\eta,\zeta)=\mathcal{N}_{n,l}(\eta,1)\zeta^{\bar{n}+3/2} \tag{IV.59}$$

where the normalization constants for $\zeta=1$ are tabulated in ref. 14,15 and summarized in the table IV.2.

Table IV.2: *Normalization factors of the Slater-Gaussian orbitals (For $\eta=1$ they are reported in ref. 14,15) the others are computed through numerical integration using the software Mathematica.*

	$\mathcal{N}_{n,l}(1,\eta)$			
$\bar{n}$	$\eta=1$	$\eta=0.5$	$\eta=0.1$	$\eta=0.01$
0	1.126467421	1.402524252	1.825237677	1.980293982
1	0.576609950	0.769670200	1.049727352	1.143267860
2	0.196581141	0.274074953	0.382737444	0.417455522
3	0.050275655	0.072177594	0.102211707	0.111568758
4	0.010280772	0.015070484	0.021537912	0.023520642

IV.7.4 No-cusp Slater type orbitals

We define this primitive Slater function, multiplied by a polynomial that depends on the exponential parameter

$$P_n(r_a)=\mathcal{N}_{n,l}(\zeta)r_a^n(1+\zeta r_a)e^{-\zeta r_a}. \tag{IV.60}$$

The normalization factor of these orbitals is equal to

$$\mathcal{N}_{n,l}(\zeta)=\zeta^{\bar{n}+3/2}\sqrt{\frac{(2\bar{n}+3)}{(2\bar{n}+7)(2\bar{n}+4)!}}2^{\bar{n}+5/2}. \tag{IV.61}$$

with as usual $\bar{n} = n + l$.

V

Quantum Monte Carlo methods and main input

V.1 Variational Monte Carlo

The variational Monte Carlo (VMC) method[48,49,50,51] is based on the application of the Monte Carlo stochastic integration to the evaluation of the energy functional

$$\mathsf{E}\left[\Psi_T\right] = \frac{\int \Psi_T^*(\bar{\mathbf{r}})\hat{\mathsf{H}}_e\Psi_T(\bar{\mathbf{r}})d\bar{\mathbf{r}}}{\int |\Psi_T(\bar{\mathbf{r}})|^2 d\bar{\mathbf{r}}} \tag{V.1}$$

over a given *trial* wave function $\Psi_T(\bar{\mathbf{r}})$. In order to compute this integral through Monte Carlo, the integrand in eq. V.1 is rewritten as the product between two functions

$$\mathsf{E}\left[\Psi_T\right] = \int E_{loc}(\bar{\mathbf{r}})\Pi(\bar{\mathbf{r}})d\bar{\mathbf{r}}$$

that are respectively the *local energy*

$$\mathsf{E}_{loc}(\bar{\mathbf{r}}) = \frac{\hat{\mathsf{H}}_e\Psi_T(\bar{\mathbf{r}})}{\Psi_T(\bar{\mathbf{r}})}, \tag{V.2}$$

defined as the energy associated to a given electronic configuration $\bar{\mathbf{r}}$, and the probability density

$$\Pi(\bar{\mathbf{r}}) = \frac{|\Psi_T(\bar{\mathbf{r}})|^2}{\int |\Psi_T(\bar{\mathbf{r}})|^2 d\bar{\mathbf{r}}} \tag{V.3}$$

to find the electrons in that particular configuration.

The splitting of the integrand into a density probability and a local quantity is applied in general to compute the averages of physical observables within Monte Carlo.
Thus, for the energy functional, once the integrand is separated as above, it is possible to extract a certain number $\mathcal{N}$ of electronic configurations with probability $\Pi(\bar{\mathbf{r}})$ estimating for each of them, the value of $\mathsf{E}_{loc}(\bar{\mathbf{r}})$, so that the energy functional $\mathsf{E}\left[\Psi_T\right]$ will be approximately equal to:

$$\mathsf{E}\left[\Psi_T\right] \approx \langle \mathsf{E}_{loc}\rangle_{\mathcal{N}} \pm \sqrt{\frac{\left\langle \mathsf{E}_{loc}^2\right\rangle_{\mathcal{N}} - \langle \mathsf{E}_{loc}\rangle_{\mathcal{N}}^2}{\mathcal{N}}}. \tag{V.4}$$

with an error that decreases as the square root of $\mathcal{N}$. Because of the stochastic nature of the QMC methods all averages are affected by a statistical error, so the correct evaluation of the observable requires also the integral of the square of the local quantity to be finite: in order for the variance to be finite and the standard error to be well defined.
One important property of the QMC methods lies in their variationality and it is the so-called *zero variance principle*: the variance of the local energies, *i.e.* $\left\langle \mathsf{E}_{loc}^2\right\rangle_{\mathcal{N}} - \langle \mathsf{E}_{loc}\rangle_{\mathcal{N}}^2$, is exactly null if $\Psi_T(\bar{\mathbf{r}})$ is an eigenfunction of the electronic Hamiltonian[52,53].

V.1.1 Metropolis-Hastings algorithm

The advantage of stochastic integration is that it can be applied to whatever complicated functions, also with explicit correlation between the integration variables, with the only condition that the variance associated with the local energy is finite.
Yet, for complicated wave functions, to generate electronic configurations according to $\Pi(\bar{\mathbf{r}}) \propto |\Psi_T(\bar{\mathbf{r}})|^2$ is a difficult if not impossible task, which also requires the knowledge of the wave functions' normalization factor, that is an integration problem of more or less the same complexity as the evaluation of the energy functional.
For this reason, in order to sample the electronic configurations the most efficient approach is to apply the general purpose sampling method (combining a Markov chain with an accept/reject procedure) which was first introduced my Metropolis *et al.*[54] and later generalized by Hastings[55].
The Metropolis-Hastings algorithm is able to generate $\mathcal{N}$ consecutive electronic configurations through the following steps that describe a random walk:

1. Initialise a *walker* in an electronic configuration $\bar{\mathbf{r}}$;

2. Propose a trial move from $\bar{\mathbf{r}}$ to a new position $\bar{\mathbf{r}}'$ with transition probability $\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$, so that the probability that a *walker* initially in $\bar{\mathbf{r}}$ afterwards transitions in an infinitesimal volume $d\bar{\mathbf{r}}'$ of the configurational space is equal to $d\bar{\mathbf{r}}'\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$;

3. Accepted the trial move with probability

$$\mathcal{A}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}}) = \min\left\{\frac{\Pi(\bar{\mathbf{r}}')\mathcal{T}(\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}')}{\Pi(\bar{\mathbf{r}})\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})}, 1\right\}; \tag{V.5}$$

4. If the trial move is accepted $\bar{\mathbf{r}}'$ becomes the new position of the *walker* and a new transition is proposed from there, else a new transition is proposed from the previous position $\bar{\mathbf{r}}$.

This procedure is repeated until $\mathcal{N}$ configurations (some of them repeated) are sampled. An important property of this *random walk* procedure is that it satisfies the *detailed balance principle* after an equilibration period. To prove this and to show its implications, let's assume to have a distribution of independent walkers according to a general function $\mathcal{P}(\bar{\mathbf{r}})$, so that the number of walkers in an infinitesimal volume of space is $d\bar{\mathbf{r}}\mathcal{P}(\bar{\mathbf{r}})$. Once the random walk of each walker is progressed for a sufficient amount of time we should reach an equilibrium in which the number of walkers transitioning between two volumes of space $d\bar{\mathbf{r}}'$ and $d\bar{\mathbf{r}}$ in one time step should be the same. This equilibrium condition is expressed as:

$$d\bar{\mathbf{r}}'\mathcal{A}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})\mathcal{P}(\bar{\mathbf{r}})d\bar{\mathbf{r}} = d\bar{\mathbf{r}}\mathcal{A}(\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}')\mathcal{T}(\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}')\mathcal{P}(\bar{\mathbf{r}}')d\bar{\mathbf{r}}'$$

so that:

$$\frac{\mathcal{P}(\bar{\mathbf{r}}')}{\mathcal{P}(\bar{\mathbf{r}})} = \frac{\mathcal{A}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})}{\mathcal{A}(\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}')\mathcal{T}(\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}')}.$$

By substituting the acceptance probability defined in eq. V.5 we have that

$$\frac{\mathcal{P}(\bar{\mathbf{r}}')}{\mathcal{P}(\bar{\mathbf{r}})} = \frac{\Pi(\bar{\mathbf{r}}')}{\Pi(\bar{\mathbf{r}})}$$

and this guarantees that the random walk is actually sampling the correct probability density $\Pi(\bar{\mathbf{r}})$ and so that the **detailed balance principle** is satisfied.

One important comment that has to be made, is that the random walk does not need the probability density to be normalized since in the definition of $\mathcal{A}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$ only the ratio between probability densities appears, canceling out the normalization factor. This leaves us with the advantage of having to compute only the square modulus of the wave function $|\Psi_T(\bar{\mathbf{r}})|^2$ (a local quantity) and not its integral.
The main drawback of *random walks* is that the configurations that are generated, are correlated since the new configurations have *memory* of the space from which they have transitioned. A discussion about correlation and the common approaches to mitigating these biases can be found in section **??**. For now, in the following section, we want to discuss the importance of the role played be the transition probabilities.

V.1.2 Transition probabilities

Within the Metropolis-Hastings algorithm the transition probability $\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$ plays a fundamental role in ensuring both that there is low correlation between consecutive Monte Carlo steps, which is essential for the estimation of the observable's variance and a high acceptance rate, that ensures lesser time lost in proposing moves that are then rejected.
In the following paragraphs, we will discuss the most common transition probabilities and how they are implemented.

V.1.2.A. Uniform or Gaussian transition probabilities. The simplest and most common transition probability is that of proposing a move in a $3N_e$ uniform space of volume $(2\delta\tau)^{3N_e}$ centered around the initial position $\bar{\mathbf{r}}$:

$$\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}}) = \frac{1}{(2\delta\tau)^{3N_e}}. \tag{V.6}$$

This is effectively done by updating the electronic position according to the rule:

$$\bar{\mathbf{r}}' = \bar{\mathbf{r}} + \bar{\eta}\delta\tau \tag{V.7}$$

where $\bar{\eta}$ is a vector of $3N_e$ independent *pseudo*-random numbers generated uniformly in the closed interval $[-1:1]$.
Clearly, this transition probability has the limit of proposing moves that are at most of length $\delta\tau$, thus limiting the configurational space that is sampled in short periods and

giving high correlation. Another aspect is that the uniform probability disregards the fact that the $\Pi(\bar{\mathbf{r}})$ usually varies greatly in space, also depending on its dimensionality.
A way to partially overcome these difficulties is to use continuous functions to define $\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$. The first, more intuitive choice, which also comes from the analogy with diffusion Monte Carlo, is that of a Gaussian distribution

$$\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}}) = \frac{1}{(2\pi\delta\tau)^{3N_e/2}} e^{-\frac{|\bar{\mathbf{r}}'-\bar{\mathbf{r}}|^2}{2\delta\tau}} \tag{V.8}$$

centered on the initial position $\bar{\mathbf{r}}$ and variance $\delta\tau$. These steps are proposed according to the rule:

$$\bar{\mathbf{r}}' = \bar{\mathbf{r}} + \bar{\eta}\sqrt{\delta\tau} \tag{V.9}$$

where $\bar{\eta}$ is a vector of $3N_e$ independent *pseudo*-random numbers generated according to a Gaussian distribution centered in 0 and with standard deviation $\sqrt{\delta\tau}$.
Even though this function in principle is able to propose large random steps and to connect all the configurational space, reducing correlation, it is still unable to account for the variation of $\Pi(\bar{\mathbf{r}})$. In order to do so, different $\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}})$ have been proposed, to take for example into account the higher density around the nuclear positions[56,57], or based on *damped*[58,57,59] or *undamped Langevin dynamics*[60].

V.1.3 Overdamped Langevin dynamics

The traditional sampling methods reported in the previous section are still not optimal since they still have a high rejection rate. It has been empirically shown that a good value Δ for these sampling procedures is the one that gives an acceptance rate of around 50%. One method to improve the sampling is based on the so-called *importance sampling* which takes into account the form of the density probability through its first-order derivative.
The importance sampling corresponds to overdamped Langevin dynamics in which the move Euler scheme (with the *space/time* discretization) is proposed as

$$\bar{\mathbf{r}}' = \bar{\mathbf{r}} + \sqrt{\Delta}\bar{\eta} + \Delta\bar{\mathbf{v}}_D(\bar{\mathbf{r}}) \tag{V.10}$$

where $\bar{\mathbf{v}}_D = \nabla_{\bar{\mathbf{r}}} \ln\left[\Psi_T(\bar{\mathbf{r}})\right]$ is the drift velocity associated to the gradient of the logarithm of the wave function, and $\bar{\eta}$ is a vector of $3N_e$ independent *pseudo*-random numbers generated according to a Gaussian distribution centered in 0 and with standard deviation $\sqrt{\Delta}$. The transition probability in this case is defined as

$$\mathcal{T}(\bar{\mathbf{r}}' \leftarrow \bar{\mathbf{r}}) = \frac{1}{(2\pi\Delta)^{3N_e/2}} e^{-\frac{|\bar{\mathbf{r}}'-\bar{\mathbf{r}}-\Delta\bar{\mathbf{v}}_D(\bar{\mathbf{r}})|^2}{2\Delta}}. \tag{V.11}$$

If $\Delta \to 0$ the acceptance probability goes to 1 and all the moves are always accepted. Due to the discretization of the move's amplitude, we have to reintroduce an acceptance/rejection step, as defined in eq. V.5. In this specific case, an empirical way to choose the value of Δ is to converge to an acceptance rate lower than 90%.

V.1.4 Two level sampling

A way to further reduce correlation effects in the sampling, enhancing the accuracy for a fixed sample size, can be obtained through the two-step sampling proposed by M. Dewing in ref. 61.
The idea of the author was to split the acceptance probability in eq. V.5 into two parts: one associated with the determinant part and the one-body Jastrow terms

$$\mathcal{A}^{(1)}(\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}}) = \min\left\{\frac{|\mathcal{F}(\bar{\mathbf{x}}')|^2\mathcal{T}(\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}}')}{|\mathcal{F}(\bar{\mathbf{x}})|^2\mathcal{T}(\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}})} e^{2\left[\mathrm{J}_c^{1b}(\bar{\mathbf{r}}') - \mathrm{J}_c^{1b}(\bar{\mathbf{r}})\right]}, 1\right\}; \tag{V.12}$$

and a second one associated with the two-body and few-body terms in the Jastrow factor

$$\mathcal{A}^{(2)}(\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}}) = \min\left\{e^{2\left[\mathrm{J}^{2b}(\bar{\mathbf{x}}') - \mathrm{J}^{2b}(\bar{\mathbf{x}})\right]}, 1\right\}. \tag{V.13}$$

Each acceptance probability is verified separately. The advantage of this procedure lies, not only in the fact that it reduces the correlation between configurations, but also in the fact that the computation of the one-body is usually faster than those of the many-body terms in the Jastrow, which are now computed only if $\mathcal{A}^{(1)}(\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}})$ is satisfied.

V.2 Variational Monte Carlo calculations

The simplest quantum Monte Carlo run that we want to describe is the single point energy evaluation on a trial wave function that has already been optimized: we refer to this as Variational Monte Carlo.
All variational Monte Carlo calculations require the presence (inside the running directory), of the saved wave function directory `wvfn.save` that has to be read.
In the next section, we will describe the files that are contained in this directory, for now, we just assume that the optimized wave function has been stored inside this directory.
The variational Monte Carlo sampling is regulated by three parameters that are: `n_bra`, `bin_l` and `n_bin` corresponding respectively to the number of consecutive single electron Monte Carlo steps that are executed 'before' the observable (local energy and derivatives) are stored; length of the block (or bin) in which the observable are accumulated, and number of blocks (or bins) used to compute the quantities.

Table V.1: *List of basic variables contained in the* `qmcsmp` *namelist. Additional variables can be found in Tab. V.2.*

Variable	Type	Descrition
`qmc_mthd`	char(3)	Monte Carlo method used for the sampling: `vmc` (default), `dmc` and in the future also `lrdmc`.
`dt`	real64	Length of the Monte Carlo step for VMC. In DMC it corresponds to the time step discretization.
`n_trm`	int32	Number of thermalization steps (default is 10).
`n_wlk`	int32	Number of walkers (default is 1 per MPI task).
`bin_l`	int32	Length of the blocks in the reblocking procedure. The real length is actually the product of this parameter and the `n_bra` one, which for default is set to twice the number of electrons.
`n_bin`	int32	Number of blocks in the reblocking procedure.
`n_bra`	int32	Number of steps before branching for DMC otherwise it is an automatic rescaling of the total sampling that depends on the number of electrons. (Default is $2N_e$).

This means that the total sampling used to compute the observable in `bin_l`$\times$`n_bin` and that between each move `n_bra` quantities are discarded. The default value of `n_bra` is equal to two times the number of electrons (it can be larger) that is to say that, after $2N_e$ Monte Carlo moves with 50% acceptance rate (90% for Langevin dynamics), we assume that all electrons have been moved, the correlation between the Monte Carlo steps is low, and we can compute the full derivatives on this new configuration. Naturally, correlation and variance are computed on `n_bin` quantities.
The other important variable in the VMC run is the number of walkers `n_wlk` that specifies the number of independent Monte Carlo samplings that run in parallel. `n_wlk` is set to 1 by default, meaning that each MPI task executes only one simultaneous Monte Carlo sampling. A schematic representation of the total sampling accumulated during a VMC run can be seen in Fig. **??**. The partial list of the basic variables for a VMC run can be found in Tab. V.1.

V.2.1 Input file for a VMC energy calculation

An all-electron variational Monte Carlo calculation of electronic systems requires the address of the molecular input file (Chapter II) and of the basis set file (Chapter III), and the definition of the `bin_l` and `n_bin` variables that define the sampling. We assume to run the code in four MPI tasks, meaning that by default `n_wlk=1` and the total walkers will be 4 so that the total sampling will be of $4\times$`bin_l`$\times$`n_bin`.
There is no need to specify the method since `qmc_mthd='vmc'` is the default, and there is no need to specify the time step `dt` of the Monte Carlo step's amplitude, since this is automatically rescaled at the beginning of the run, with the condition that the acceptance rate is approximately 50%. Finally, the thermalization steps are by default `n_trm=10` which is more than sufficient to thermalize the run. Since no averages are computed `n_trm=1` is also acceptable. The thermalization is repeated until the acceptance rate is 50% so that equilibration is surely reached, and at least one step is repeated. In this case, the basic input file for variational Monte Carlo is written as:

Input for the Variational Monte Carlo calculation for a H2O molecule

```
&qmcsmp
     bin_l = 2000
     n_bin = 100
/
&basset
 fle_basis = 'h2o_qmecha.bas'
 fle_pspot = 'pseudo.dat'
/
&molsys
 fle_coord = 'h2o_qmecha.xyz'
/
```

Notice that the specification of the molecular file `h2o_qmecha.xyz`, the basis set file `h2o_qmecha.bas` and the pseudopotential file are optional if the code is run specifying the flags `-m h2o_qmecha.xyz -b h2o_qmecha.bas -p pseudo.dat` in the list of input arguments.

V.2.2 Output file for the VMC energy calculation

Let us now analyze the output of the VMC run described in the previous session. After the first lines that describe the system, the VMC run starts with the following information regarding the type of sampling, the initial value of the time step, it sets the type step as a tunable variable (`var_dt=T`) and it all the main variable regarding the characteristics of the sampling. In particular the total number of samplings used for the estimation of the observable will be `n_tot_wlk*n_bin*bin_l`.

Start of the variational Monte Carlo run

```
                        VARIATIONAL MONTE CARLO

                    --------------------------------
                    Variational Monte Carlo sampling

 3D Gaussian Transition prob. (dt*I covariance).
 Two step sampling activated

 Initial Metropolis step for electrons           (dt_e) :  0.507000E+00
 Variable metropolis step                       (var_dt) :             T
 Total number of walkers                    (n_tot_wlk) :            10
 Number of independent walkers per MPI task     (n_wlk) :             1
 Monte Carlo sampling per core             (n_tot_smpl) :        200000
 Particle sweeps before derivatives computat(n_bra_vmc) :             2
 Number of moves before matrix recomputation    (n_scr) :             8
```

After these initial information the equilibration steps are executed, with an automatic evaluation of the best time step satisfying the condition that the acceptance rate is around 50%.

Thermalization and Δt evaluation

```
                Variational Monte Carlo thermalization

          Block N°      Acc. rate. [%]      Time step [au]
            1              22.196           0.24991812E-01
            2              67.599           0.44652148E-01
            3              59.856           0.67610540E-01
```

```
                7              50.005          0.84374152E-01
                8              50.099          0.84844228E-01
                9              50.021          0.84944452E-01
```

Having converged to a time step of 0.084944 Bohr within 9 blocks, the code starts the sampling to evaluate the VMC observable, assuming that the walkers are now thermalized,

Sampling and final acceptance rate

```
                    Variational Monte Carlo sampling

  Block Num.          1 sampled in                [sec.] :  0.665000E+00
  Block Num.          2 sampled in                [sec.] :  0.667000E+00
  Block Num.          3 sampled in                [sec.] :  0.669000E+00
                        .
                        .
                        .
  Block Num.        100 sampled in                [sec.] :  0.669000E+00

  Acceptance rate of Monte Carlo moves for elec.      [%] :  0.498761E+02
```

and finishes by printing the effective acceptance rate computed during the sampling. After this is done the code automatically starts the computation of the physical observable through the reblocking procedure, which is printed in the `vmc_eval.dat` file, or if the name of the system is specified through the input argument `-sn {name_of_run}` the output file will be `vmc_eval_{name_of_run}.dat`
The code first computes the correlated energy and variance (`bin_l=1`) computed with VMC, so that the errors are typically underestimated due to correlation.

Correlated values of the energy and variance (`bin_l=1`).

```
              Correlated estimation of energy and variance

  Energy           [Eh] : -0.17219895657409E+02 +/- 0.38269619654366E-03
  Variance       [Eh^2] :  0.29291275769679E+00 +/- 0.29291283092559E-03
```

After this first estimation, the code generates a table of the energy's standard error (and its associated error) as a function of the block lengths used in the reblocking procedure to decorrelate the error, and from these it estimates the best block length to be used in the observables' error estimations

Energy error per block size

```
                Energy error variation per block size

     Block length       Error [Eh]       Stnd. error of error [Eh]
                1       0.3822550E-03            0.6043998E-06
                2       0.4301557E-03            0.9618622E-06
                4       0.4784239E-03            0.1512924E-05
                5       0.4923102E-03            0.1740601E-05
                8       0.5177853E-03            0.2315653E-05
               10       0.5286657E-03            0.2643394E-05
               16       0.5468700E-03            0.3458848E-05
               20       0.5554721E-03            0.3927978E-05
               25       0.5625020E-03            0.4447247E-05
               32       0.5643364E-03            0.5047982E-05
```

where the error of the error bar is estimated as $\sqrt{\text{Var}[E_b]/(2N_b(N_b-1))}$ where N_b are the number of blocks and $Var[E_b]$ is the variance of the blocked local energies. In this example, the error converges with a block length of around 32 and it subsequently prints the final evaluations of the various quantities, in this simple case only the energy, and energy components.

Final observable estimation

```
  Block len. / Num. of Blocks                              32/      6250

  Energy           [Eh] : -0.17219895657409E+02 +/- 0.56433638469335E-03
  Kin.Ene.         [Eh] :  0.13795568683029E+02 +/- 0.13557983619447E-01
  Pot.Ene.         [Eh] : -0.37994972231571E+02 +/- 0.13591819031958E-01
```

The sampling presented in this example has been done with the two-step sampling described in the previous section, which is the recommended approach set as the default sampling method. (Additional variables of the `qmcsmp` name list are listed in Table V.2)

Table V.2: *List of additional variables contained in the* `qmcsmp` *namelist.*

Variable	Type	Descrition
`dt_e`	real64	Alternative varible to set the length of the VMC step for for the electrons. Overwrites `dt`.
`dt_p`	real64	Alternative varible to set the length of the VMC step for the positrons. Overwrites `dt`.
`dt_d`	real64	Alternative varible to set the length of the VMC step for for drudons. Overwrites `dt`.
`n_scr`	int32	Number of Monte Carlo steps before the matrices are recomputed by scratch. (Default is N_e).
`smp_mthd`	char(3)	Sets the different types of sampling that can be used. (Default is Gaussian 3D two-step dynamics `g3d`). It can also be replaced by the Langevin dynamics that is used in DMC. (`l3d`)
`two_step_sampling`	logical	If `.true.` (Default is `.true.`) uses the two step sampling approach described in ref. 61 and Sec. V.1.4.
`r_seed`	int32	Initial seed for the random number generator.
`nuc_corr`	logical	If `.true.` (Default is `.false.`) uses Umrigar, Nightingale and Runge scheme[57] to better sample the nuclear region in the Langevin dynamics (`smp_mthd='l3d'`) in all-electron calculations.
`a_drft`	real64	(Default is `0.5`) Parameter for the cut-off of the drift velocity[57] (See sec. V.3.6 for details).

V.3 Diffusion Monte Carlo

As previously discussed, in the VMC method the quality of the description of the ground state and of its energy, strictly depends on the parametrization of the trial wave function $\Psi_T(\bar{\mathbf{r}};\bar{\alpha})$, and so on the dimension of the variational space. To overcome this limitation Diffusion Monte Carlo (DMC) methods[62,63] have been introduced. These methods are based on a very simple idea: that is the transformation of the time-dependent Schrödinger equation in a diffusion equation in imaginary time. To describe the basic concepts of this method, let's consider a system of N_e electrons described through the Hamiltonian $\hat{H}_e$ **??** which at time t_0 is in an initial state $\varphi(\bar{\mathbf{r}},t_0)$, that can be expressed as a linear combination of a complete orthonormal basis in the Hilbert space, corresponding to the stationary eigenstates of the same Hamiltonian:

$$\varphi(\bar{\mathbf{r}},t_0)=\sum_{k=0}^{\infty}a_k\Psi_k(\bar{\mathbf{r}}). \tag{V.14}$$

Its evolution in time will satisfy the time-dependent Schrödinger equation[64]

$$i\hbar\frac{\partial}{\partial t}\varphi(\bar{\mathbf{r}},t)=\hat{\mathrm{H}}_e\varphi(\bar{\mathbf{r}},t). \tag{V.15}$$

who's solutions are the sum over all the stationary eigenstates, each multiplied by a phase that depends on the associated eigenvalue:

$$\varphi(\bar{\mathbf{r}},t)=\sum_{k=0}^{\infty}a_k\left(t\right)\Psi_k(\bar{\mathbf{r}})\qquad a_k\left(t\right)=a_k\left(t_0\right)e^{-i\mathrm{E}_k(t-t_0)/\hbar}. \tag{V.16}$$

The fundamental idea behind the DMC methods is to transform eq. V.15, in a diffusion equation, substituting the real time variable t with the imaginary time $t=-i\tau$ and adding a constant shift in the energies E_R, so that eq. V.15 assumes the following form:

$$\hbar\frac{\partial\varphi(\bar{\mathbf{r}},\tau)}{\partial\tau}=\frac{\hbar^2}{2m_e}\nabla^2\varphi(\bar{\mathbf{r}},\tau)-\left(V(\bar{\mathbf{r}})-\mathrm{E}_R\right)\varphi(\bar{\mathbf{r}},\tau). \tag{V.17}$$

The solutions in imaginary time of this new equation will still be linear combinations of the stationary states of the time-independent Schrodinger equation

$$\varphi(\bar{\mathbf{r}},\tau)=\sum_{k=0}^{\infty}a_k e^{-(\mathrm{E}_k-\mathrm{E}_R)\tau/\hbar}\Psi_k(\bar{\mathbf{r}}), \tag{V.18}$$

yet, in this case, each state will be multiplied by an exponential function that decays as $\tau\to\infty$ according to the energy differences $\mathrm{E}_k-\mathrm{E}_R$.
Now, the role of the constant energy shift E_R, becomes evident[62,65]. If $\mathrm{E}_R\approx\mathrm{E}_0$, where E_0 is the ground state's energy, by letting the solution evolve for a reasonable amount

of time, the only component that will survive in the expansion V.18 will be the one corresponding to $\Psi_0(\vec{\mathbf{r}})$.
As a matter of fact, all the excited states with $\mathrm{E}_k > \mathrm{E}_R$ will decay faster in time, while all the states with $\mathrm{E}_k \leq \mathrm{E}_R$ will gain more and more weight.
In general we must add that whatever the choice of E_R the weight of the ground state with respect to the others, will always grow exponentially in the imaginary time τ, so that for $\tau \to \infty$ the algorithm will extract the ground state component $\Psi_0(\bar{\mathbf{r}})$ from a general initial wave function $\varphi(\bar{\mathbf{r}}, \tau_0)$, given that latter is not orthogonal to the former.

V.3.1 Simple diffusion Monte Carlo

In order to derive the simple diffusion Monte Carlo (DMC) algorithm[62] we can start by rewriting the solution in eq. V.18 in the integral form

$$\varphi(\bar{\mathbf{r}}', \tau) = \int_V G(\bar{\mathbf{r}}', \tau; \bar{\mathbf{r}}, \tau_0)\varphi(\bar{\mathbf{r}}, \tau_0)d\bar{\mathbf{r}} \tag{V.19}$$

where the kernel of the integral is the Green function that defines the time propagation of the solution from $\bar{\mathbf{r}}$ to $\bar{\mathbf{r}}'$ in the time interval $\tau - \tau_0$, and has the form

$$G(\bar{\mathbf{r}}', \tau; \bar{\mathbf{r}}, \tau_0) = \langle \bar{\mathbf{r}}' | e^{-(\tau-\tau_0)\left(\hat{\mathrm{T}}+\hat{\mathrm{V}}-\mathrm{E}_R\right)/\hbar} | \bar{\mathbf{r}} \rangle. \tag{V.20}$$

It can be shown that the Green function is the solution of the time-dependent Schrödinger equation in $\bar{\mathbf{r}}$ e τ,

$$\hbar \frac{\partial}{\partial \tau} G(\bar{\mathbf{r}}, \tau; \bar{\mathbf{r}}_0, \tau_0) = (\hat{\mathrm{H}}_e - \mathrm{E}_R) G(\bar{\mathbf{r}}, \tau; \bar{\mathbf{r}}_0, \tau_0), \tag{V.21}$$

with the initial conditions

$$\lim_{\tau \to \tau_0} G(\bar{\mathbf{r}}, \tau; \bar{\mathbf{r}}_0, \tau_0) = \delta(\bar{\mathbf{r}} - \bar{\mathbf{r}}_0). \tag{V.22}$$

The exponential that appears in eq. V.20 is not separable since the operators $\hat{\mathrm{T}}$ and $\hat{\mathrm{V}}$ do not commute. Yet, a way to factorize the exponential operator is through the use of the Trotter-Suzuki decomposition[66], for which the time interval of the propagation is discretized in $\mathcal{M}$ intervals of length $\delta\tau = (\tau - \tau_0)/\mathcal{M}$ so that

$$e^{-(\tau-\tau_0)(\hat{\mathrm{T}}+\hat{\mathrm{V}}-\mathrm{E}_R)/\hbar} = \lim_{\mathcal{M}\to\infty} (e^{-(\hat{\mathrm{V}}-\mathrm{E}_R)\frac{\delta\tau}{2\hbar}} e^{-\hat{\mathrm{T}}\frac{\delta\tau}{\hbar}} e^{-(\hat{\mathrm{V}}-\mathrm{E}_R)\frac{\delta\tau}{2\hbar}})^{\mathcal{M}}. \tag{V.23}$$

For finite small intervals the full operator is approximated by the product

$$e^{-\delta\tau(\hat{\mathrm{T}}+\hat{\mathrm{V}}-\mathrm{E}_R)} = e^{-(\hat{\mathrm{V}}-\mathrm{E}_R)\frac{\delta\tau}{2\hbar}} e^{-\hat{\mathrm{T}}\frac{\delta\tau}{\hbar}} e^{-(\hat{\mathrm{V}}-\mathrm{E}_R)\frac{\delta\tau}{2\hbar}} + \mathcal{O}(\delta\tau^3), \tag{V.24}$$

with an error of the order of the third power of $\delta\tau$. By applying this discretization to the Green function V.20 we obtain that the evolution in an infinitesimal time $\delta\tau$ between to general configurations $\bar{\mathbf{r}}$ and $\bar{\mathbf{r}}'$ will be written as:

$$\begin{aligned} G(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) = \langle \bar{\mathbf{r}}' | e^{-\frac{\delta\tau}{\hbar}(\hat{\mathrm{T}}+\hat{\mathrm{V}}-\mathrm{E}_R)} | \bar{\mathbf{r}} \rangle \approx \\ \approx e^{-(\mathrm{V}(\bar{\mathbf{r}}')+\mathrm{V}(\bar{\mathbf{r}})-2\mathrm{E}_R)\frac{\delta\tau}{2\hbar}} \langle \bar{\mathbf{r}}' | e^{-\hat{\mathrm{T}}\frac{\delta\tau}{\hbar}} | \bar{\mathbf{r}} \rangle \end{aligned} \tag{V.25}$$

Since $\langle \bar{\mathbf{r}}' | e^{-\hat{\mathrm{T}}\frac{\delta\tau}{\hbar}} | \bar{\mathbf{r}} \rangle$ is the Green function for the non-interacting systems, defined as

$$\langle \bar{\mathbf{r}}' | e^{-\hat{\mathrm{T}}\frac{\delta\tau}{\hbar}} | \bar{\mathbf{r}} \rangle = \left(\frac{m_e}{2\pi\hbar\delta\tau}\right)^{3N_e/2} e^{-\frac{m_e}{2\hbar\delta\tau}|\bar{\mathbf{r}}'-\bar{\mathbf{r}}|^2} \tag{V.26}$$

the final Green function form will be approximated by the product

$$G(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) \approx P(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) W(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau)$$

of a diffusion function

$$P(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) = \left(\frac{m_e}{2\pi\hbar\delta\tau}\right)^{3N_e/2} e^{-\frac{m_e}{2\hbar\delta\tau}|\bar{\mathbf{r}}'-\bar{\mathbf{r}}|^2} \tag{V.27}$$

and a *weighting* or *growth/decay* function

$$W(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) = e^{-\frac{\delta\tau}{2\hbar}(V(\bar{\mathbf{r}}')+V(\bar{\mathbf{r}})-2E_R)}, \tag{V.28}$$

If we apply this decomposition to the time evolution in eq. V.19, we obtain the integral solution

$$\begin{aligned} \varphi(\bar{\mathbf{r}}', \tau) = \int_V \Bigg[\prod_{m=1}^{\mathcal{M}} d\bar{\mathbf{r}}^{m-1} \delta(\bar{\mathbf{r}}' - \bar{\mathbf{r}}^{\mathcal{M}}) \times \\ \times P(\bar{\mathbf{r}}^m, \bar{\mathbf{r}}^{m-1}; \delta\tau) W(\bar{\mathbf{r}}^m, \bar{\mathbf{r}}^{m-1}; \delta\tau) \Bigg] \varphi(\bar{\mathbf{r}}^0, \tau_0), \end{aligned} \tag{V.29}$$

which propagates $\varphi(\bar{\mathbf{r}}^0, \tau_0)$ in time through $\mathcal{M}$ approximated infinitesimal steps.

Because of the discretization introduced in the time evolution, DMC calculations always suffer of the so called *time-step* error that affects the estimation of the various observable, and that is compensated by extrapolating the values obtained for different time steps $\delta\tau$ to the limit of continuum $\delta\tau \to 0$.
To understand the basic procedure to stochastically solve integral V.29, let us assume to have $\bar{\mathbf{r}}^{n,0}$ with $n = 1, 2, ..., \mathcal{N}$ configurations initially distributed according to $\varphi(\bar{\mathbf{r}}, \tau_0)$. Each configuration, is assumed to be the initial position of an independent random walk that evolves in the time interval $\tau - \tau_0$ from $\bar{\mathbf{r}}^{n,0}$ to the final one, $\bar{\mathbf{r}}^{n,\mathcal{M}}$.
For each walker the total probability of a certain path will be defined as the product of the single diffusion probabilities

$$\mathcal{P}(\bar{\mathbf{r}}^{n,\mathcal{M}}, \tau; \bar{\mathbf{r}}^{n,0}, \tau_0) = \prod_{m=1}^{\mathcal{M}} P(\bar{\mathbf{r}}^{n,m}, \bar{\mathbf{r}}^{n,m-1}; \delta\tau), \tag{V.30}$$

and each walker will accumulate a *weight* given by the product

$$\mathcal{W}(\bar{\mathbf{r}}^{n,\mathcal{M}}, \tau; \bar{\mathbf{r}}^{n,0}, \tau_0) = \prod_{m=1}^{\mathcal{M}} W(\bar{\mathbf{r}}^{n,m}, \bar{\mathbf{r}}^{n,m-1}; \delta\tau). \tag{V.31}$$

At the end of the time evolution the wave function $\varphi(\bar{\mathbf{r}}, \tau)$ will be represented through the distribution of the walkers, each multiplied by its weight:

$$\varphi(\bar{\mathbf{r}}', \tau) = \sum_{i=1}^{\mathcal{N}} \mathcal{W}(\bar{\mathbf{r}}^{n,\mathcal{M}}, \tau; \bar{\mathbf{r}}^{n,0}, \tau_0)\delta(\bar{\mathbf{r}}^{n,\mathcal{M}} - \bar{\mathbf{r}}'). \tag{V.32}$$

Within this simple procedure, the time dependent wave function $\varphi(\bar{\mathbf{r}}, \tau_0)$ clearly acts as a probability density, and thus it must be a positive function in all the volume. This limitation excludes the possibility to apply this basic algorithm to our systems of interest, systems of N_f interacting fermions, which thus suffer for a sign-problem. This problem is cured by introducing the **fixed-node approximation**.

V.3.2 Fixed-node approximation in diffusion Monte Carlo

As anticipated in the previous section, for fermionic systems the wave function changes sign and the simple diffusion Monte Carlo algorithm cannot be used. The region in which the function changes sign is an hypersurface of $3N_e - 1$ dimensions, that includes all the configurations where the wave function is zero, the so called *nodal surface*. This surface has nearly the same complexity of the exact ground state[67] so that direct knowledge is impossible.
Despite this, it is possible to demonstrated that by approximating the exact nodal surface with that of a trial wave function $\Psi_T(\bar{\mathbf{r}})$ we recover a variational principle for which the approximate energy is an upper bound of the exact ground state eigenvalue E_0[65,68].
As a matter of fact, the nodal surface of the trial wave function $\Psi_T(\bar{\mathbf{r}})$ divides the space in a certain number of volumes v_α in which the sign of $\Psi_T(\bar{\mathbf{r}})$ is constant. In each of these regions there will be one and only one bosonic ground state $\psi_\alpha(\bar{\mathbf{r}})$, satisfying the equations:

$$\begin{cases} \hat{\mathsf{H}}_e\psi_\alpha(\bar{\mathbf{r}}) = \mathsf{E}_\alpha\psi_\alpha(\bar{\mathbf{r}}) & \bar{\mathbf{r}} \in v_\alpha \\ \psi_\alpha(\bar{\mathbf{r}})\Psi_T(\bar{\mathbf{r}}) > 0 \end{cases} \tag{V.33}$$

that are zero at the borders of the regions, *i.e.* at the nodal surface. Starting from the fundamental state of a volume α, if we antisymmetrize the wave function $\psi_\alpha(\bar{\mathbf{r}})$

$$\tilde{\psi}_\alpha(\bar{\mathbf{r}}) = \frac{1}{N_P}\sum_P (-1)^P \psi_\alpha(\hat{P}\bar{\mathbf{r}}), \tag{V.34}$$

the permutation operator $\hat{P}$ will map the electronic coordinates in the same v_α or in an equivalent volume, while outside the nodal pocket the function will be null. In this way we define a wave function $\tilde{\psi}_\alpha(\bar{\mathbf{r}})$ in all the space V with the same nodal surface of the trial wave function, with a variational energy of

$$\frac{\int_V d\bar{\mathbf{r}}\tilde{\psi}^*_\alpha(\bar{\mathbf{r}})\hat{\mathsf{H}}_e\tilde{\psi}_\alpha(\bar{\mathbf{r}})}{\int_V d\bar{\mathbf{r}}\tilde{\psi}^*_\alpha(\bar{\mathbf{r}})\tilde{\psi}_\alpha(\bar{\mathbf{r}})} = \mathsf{E}_\alpha \geq \mathsf{E}_0. \tag{V.35}$$

This condition guarantees that the probability density $f_\alpha(\bar{\mathbf{r}}) = \tilde{\psi}_\alpha(\bar{\mathbf{r}})\Psi_T(\bar{\mathbf{r}})$ is always non negative, so that the terms that appear in the sum of eq. V.34 must have all the same sign and for this reason $\tilde{\psi}_\alpha(\bar{\mathbf{r}})$ cannot be null in v_α.
Thus, a way to obtain the ground state energy of a single pocket is to sample the function $f_\alpha(\bar{\mathbf{r}}, \tau \to \infty)$, estimating the energy through the mean value of the corresponding

local energies:

$$\langle \mathsf{E}_{loc} \rangle_{\mathcal{N}} = \frac{\int_V d\bar{\mathbf{r}} f_\alpha(\bar{\mathbf{r}}, \tau \to \infty) \frac{\hat{\mathsf{H}}_e \Psi_T(\bar{\mathbf{r}})}{\Psi_T(\bar{\mathbf{r}})}}{\int_V d\bar{\mathbf{r}} f_\alpha(\bar{\mathbf{r}}, \tau \to \infty)} = \mathsf{E}_\alpha \geq \mathsf{E}_0. \tag{V.36}$$

In the limit of $\tau \to \infty$ in each volume the initial wave functions $\varphi_\alpha(\bar{\mathbf{r}}, \tau_0)$ will converge towards the ground state function $\psi_\alpha(\bar{\mathbf{r}})$

$$f(\bar{\mathbf{r}}, \tau \to \infty) = \sum_\alpha c_\alpha \tilde{\psi}_\alpha(\bar{\mathbf{r}}) \Psi_T(\bar{\mathbf{r}}) \tag{V.37}$$

where c_α are the positive coefficients, proportional to the initial population in v_α.
At the end the Schrödinger equation is solved exactly with the boundary conditions that the wave function vanishes on the nodes of the trial wave function $\Psi_T(\bar{\mathbf{r}})$ instead of those of the unknown exact ground state eigenfunction: this idea is at the basis of the **fixed-node approximation**.
In order to derive the fixed-node DMC algorithm, let's consider the wave function

$$f(\bar{\mathbf{r}}, \tau) = \varphi(\bar{\mathbf{r}}, \tau) \Psi_T(\bar{\mathbf{r}}) : \tag{V.38}$$

the equation that describes the evolution of $f(\bar{\mathbf{r}}, \tau)$, can be obtained by multiplying the right and left sides of the Schrödinger equation V.17 by $\Psi_T(\bar{\mathbf{r}})$:

$$\hbar \frac{\partial}{\partial \tau} \varphi(\bar{\mathbf{r}}, \tau) \Psi_T(\bar{\mathbf{r}}) = \left[\frac{\hbar^2}{2m_e} \nabla^2 \varphi(\bar{\mathbf{r}}, \tau) - (\mathsf{V}(\bar{\mathbf{r}}) - \mathsf{E}_R) \varphi(\bar{\mathbf{r}}, \tau) \right] \Psi_T(\bar{\mathbf{r}}). \tag{V.39}$$

Since $\Psi_T(\bar{\mathbf{r}})$ is time independent, we can explicit the Laplacian as a function of $f(\bar{\mathbf{r}}, \tau)$ rewriting the equation as[65]:

$$\hbar \frac{\partial}{\partial \tau} f(\bar{\mathbf{r}}, \tau) = \frac{\hbar^2}{2m_e} \nabla^2 f(\bar{\mathbf{r}}, \tau) - \frac{\hbar^2}{m_e} \nabla \left[f(\bar{\mathbf{r}}, \tau) \bar{\mathbf{v}}_D(\bar{\mathbf{r}}) \right] - \\ - (\mathsf{E}_{loc}(\bar{\mathbf{r}}) - \mathsf{E}_R) f(\bar{\mathbf{r}}, \tau), \tag{V.40}$$

where we find two quantities that only depend on the trial wave function, that are respectively the *local energy* defined in VMC

$$\mathsf{E}_{loc}(\bar{\mathbf{r}}) = \frac{\hat{\mathsf{H}}_e \Psi_T(\bar{\mathbf{r}})}{\Psi_T(\bar{\mathbf{r}})}, \tag{V.41}$$

and the *drift velocity* (also known as half the *quantum force*)[65]

$$\bar{\mathbf{v}}_D(\bar{\mathbf{r}}) = \frac{\nabla \Psi_T(\bar{\mathbf{r}})}{\Psi_T(\bar{\mathbf{r}})}, \tag{V.42}$$

which has the effect of pushing the walkers away from the regions with low probability, according to the form of the trial wave function (and not according to the sole potential).
The solution of eq. V.40 will still be of the form

$$f(\bar{\mathbf{r}}', \tau) = \int_V \tilde{G}(\bar{\mathbf{r}}', \tau; \bar{\mathbf{r}}, \tau_0) f(\bar{\mathbf{r}}, \tau_0) d\bar{\mathbf{r}} \tag{V.43}$$

where the importance sampling Green function, connected to the original one through the equation

$$\tilde{G}(\bar{\mathbf{r}}', \tau; \bar{\mathbf{r}}, \tau_0) = \Psi_T(\bar{\mathbf{r}}') G(\bar{\mathbf{r}}', \tau; \bar{\mathbf{r}}, \tau_0) \frac{1}{\Psi_T(\bar{\mathbf{r}})}, \tag{V.44}$$

can still be approximated by the product of a diffusion (with the addition of a drift component) and reweighting process through the Trotter-Suzuki decomposition. For an infinitesimal time variation $\delta\tau$ we will have again that

$$\tilde{G}(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) \approx P(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) W(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau)$$

with the diffusion process defined by the function

$$P(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) = \left(\frac{m_e}{2\pi\hbar\delta\tau} \right)^{3N_e/2} e^{-\frac{m_e}{2\hbar\delta\tau} \left[\bar{\mathbf{r}}' - \bar{\mathbf{r}} - \frac{\hbar\delta\tau}{m_e} \bar{\mathbf{v}}_D(\bar{\mathbf{r}}) \right]^2} \tag{V.45}$$

and the reweighting

$$W(\bar{\mathbf{r}}', \bar{\mathbf{r}}; \delta\tau) = e^{-\frac{\delta\tau}{2\hbar}(\mathsf{S}(\bar{\mathbf{r}}') + \mathsf{S}(\bar{\mathbf{r}}))}, \qquad \mathsf{S}(\bar{\mathbf{r}}) = \mathsf{E}_{loc}(\bar{\mathbf{r}}) - \mathsf{E}_R \tag{V.46}$$

which differently from that of the simple DMC will now depend on the values of trial function's local energy.
The more $\Psi_T(\bar{\mathbf{r}})$ resembles the exact eigenfunction of the system, the more the local energy will become independent on the configuration $\bar{\mathbf{r}}$ on which it is calculated.
Before discussing the algorithm used, some considerations have to be made.

V.3.3 Basic fixed-node DMC algorithm

The algorithm used to execute the simple fixed-node diffusion Monte Carlo process to find the solution V.43 can be summarized in the following steps.

I. First, a number of $\mathcal{N}$ walkers is distributed according to the square modulus of the trial wave function $|\Psi_T(\bar{\mathbf{r}})|^2 = f(\bar{\mathbf{r}}, \tau_0)$ through a VMC random walk, which also gives a first estimation of the Energy of the system. through the mean value of the local energies. The initial weight of each of the nth walker is set to $W^{n,0} = 1$.

II. Each of the $\mathcal{N}$ configurations becomes a walker that moves with a probability $P(\bar{\mathbf{r}}^{n,m+1}, \bar{\mathbf{r}}^{n,m}; \delta\tau)$ to a new position which is proposed through the drift/diffusion equation

$$\bar{\mathbf{r}}^{n,m+1} = \bar{\mathbf{r}}^{n,m} + \frac{\hbar\delta\tau}{m_e}\bar{\mathbf{v}}_D(\bar{\mathbf{r}}^{n,m}) + \sqrt{\frac{\hbar\delta\tau}{m_e}}\bar{\rho}, \tag{V.47}$$

where $\bar{\rho}$ is a vector of random variables extracted with a Gaussian distribution with zero mean value and unitary variance.

III. Once all the $\mathcal{N}$ walkers are moved, their weights are updated by the factor:

$$W(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau) = e^{-\frac{\delta\tau}{2\hbar}\left(\mathsf{S}(\bar{\mathbf{r}}^n_{m+1})+\mathsf{S}(\bar{\mathbf{r}}^n_m)\right)}, \tag{V.48}$$

so that the evolved weight at step m' will be equal to

$$W^n_{m'} = \prod_{m=1}^{m'} W(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau) W^n_0. \tag{V.49}$$

IV. After each reweighting process the local quantities are accumulated to compute the weighted averages of each observable.

The steps from II to IV are repeated until the projection has evolved for a chosen time $\tau = \mathcal{M}\delta\tau$.

V.3.4 Acceptance probability in FN-DMC

The definition of the propagator in eq. V.45 is a good approximation in those regions where the derivative of trial wave function doesn't vary too much[65]. In the regions in which the potential has singularities (the nodal surface of the wave function), the variations of the local energy and of the drift velocity can become quite large, together with the error in the estimation of the energy. One way to alleviate this problem is to reduce the time step $\delta\tau$. Unfortunately, this has the drawback of increasing the correlation between the steps of the random walks, reducing the efficiency of the sampling.
A more effective way is to introduce, after the drift/diffusion step[65] in eq. V.47, an acceptance probability defined as

$$\mathcal{A}^n \Rightarrow \mathcal{A}(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau) = \min\left[1, \frac{|\Psi_T(\bar{\mathbf{r}}^n_{m+1})|^2 P(\bar{\mathbf{r}}^n_m, \bar{\mathbf{r}}^n_{m+1}; \delta\tau)}{|\Psi_T(\bar{\mathbf{r}}^n_m)|^2 P(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau)}\right], \tag{V.50}$$

that satisfies the *detailed balance principle*[65]. As a matter of fact, when $\delta\tau \to 0$, the number of walkers on the nodal surface vanishes, the ratio that appears in the acceptance probability tends to one and all the moves that are proposed become accepted. For $\delta\tau > 0$ some walkers will cross the nodal surface due to the approximated nature of the Green function. The moves for which after the diffusion process $\frac{\Psi_T(\bar{\mathbf{r}}_{m+1})}{\Psi_T(\bar{\mathbf{r}}_m)} < 0$ are thus rejected automatically. Some algorithms use to kill the walkers that cross the nodal surface, though this violates the detail balance principle and introduce a $\sqrt{\delta\tau}$ dependence in the *growth estimator* described in the following sections[57].
The acceptance step can be applied after moving all the particles in a configuration-by-configuration manner, or after moving every single particle in a particle-by-particle manner. The two approaches are equivalent from an algorithmic point of view since the acceptance probability of a configuration move is equivalent to the product of the acceptance probabilities of each particle move. Yet, it has been shown that the latter is always more efficient, since the overlap between two configurations after a full particle sweep is usually lower than the overlap of two configurations that differ only for one particle move, thus the probability to accept the configuration update in the latter case is higher, lowering the correlation effects during the dynamics. For this reason, the QMeCha code only applies the particle-by-particle procedure.

V.3.5 Weight variation

To take into account the change in the diffusion rate, Reynolds *et al.*[65] introduced an effective time step computed for each walker after an all N_p particle sweep and defined as

$$\delta\tau^n_{\text{eff}} = \delta\tau \frac{\sum_{i=1}^{N_p} \mathcal{A}_i^n \cdot (\Delta \mathbf{r}_i^{n,m})^2}{\sum_{i=1}^{N_p} (\Delta \mathbf{r}_i^{n,m})^2}. \tag{V.51}$$

where $\Delta\mathbf{r}_i^{n,m}$ is the diffusion distance of the electronic move, that is defined as $\Delta\mathbf{r}_i^{n,m} = \sqrt{\frac{\hbar\delta\tau}{m_i}}\bar{\rho}_i$ for the ith particle. Through the definition of this effective time step the reweighting factor V.28 becomes

$$W(\bar{\mathbf{r}}^{n,m+1}, \bar{\mathbf{r}}^{n,m}; \delta\tau) = e^{-\frac{\delta\tau_{\text{eff}}}{2\hbar}\left(\mathsf{S}(\bar{\mathbf{r}}^{n,m+1}) + \mathsf{S}(\bar{\mathbf{r}}^{n,m})\right)}. \tag{V.52}$$

V.3.6 Singularities near the nodal surface

This simple fixed-node approach is still not suitable for direct application to fermionic problems for two reasons, directly related to the behaviour of the local energy and of the drift velocity for a configuration approaching the nodal surface. For $\delta\tau \neq 0$ there is always a non zero density of walkers on the nodal surface, and as a configuration approaches the nodal surface of the trial wave function, the local energy diverges as the inverse of the distance from the surface. Since the nodal surface is $3N_e - 1$ dimensional we have that also the variance of the local energy diverges for any finite $\delta\tau$. At the same time, a configuration that has reached the nodel surface will in the following move be pushed away from it by the drift velocity for a length that is proportional to the inverse of the distance from the nodal surface. This results in a charge distribution (walker's distribution) that decays as the square of the distance and not as an exponential.
One way to address these problems is to introduce cut-offs in the values of both the local energy and the drift velocity that depend on the time step $\delta\tau$. In particular DePasquale *et al.*[69] proposed the following conditions:

$$\mathsf{E}_{loc}(\bar{\mathbf{r}}) = \begin{cases} \bar{\mathsf{E}}_{loc} + \frac{2}{\sqrt{\delta\tau}}\text{sgn}\left[\mathsf{E}_{loc}(\bar{\mathbf{r}}) - \bar{\mathsf{E}}_{loc}\right] & \text{for} \quad |\mathsf{E}_{loc}(\bar{\mathbf{r}}) - \bar{\mathsf{E}}_{loc}| > \frac{2}{\sqrt{\delta\tau}} \\ \mathsf{E}_{loc}(\bar{\mathbf{r}}) & \text{otherwise} \end{cases} \tag{V.53}$$

and

$$\bar{\mathbf{v}}_{D(i)}(\bar{\mathbf{r}}) = \begin{cases} \frac{1}{\tau}\text{sgn}\left[\bar{\mathbf{v}}_{D(i)}(\bar{\mathbf{r}})\right] & \text{for} \quad |\bar{v}_{D(i)}(\bar{\mathbf{r}})| > \frac{1}{\tau} \\ \bar{v}_{D(i)}(\bar{\mathbf{r}}) & \text{otherwise} \end{cases} \tag{V.54}$$

which do not affect the extrapolation limit for $\delta\tau \to 0$.
Another, more efficient way, was proposed by Umrigar *et al.*[57] and is based on the rescaling of the drift velocity and of the branching probability.
By considering the approximated dynamics of the electrons near the nodal surface the rescaled velocity is written as

$$\tilde{\mathbf{v}}_{D(i)} = \frac{-1 + \sqrt{1 + 2a|\mathbf{v}_{D(i)}|^2\delta\tau}}{a|\mathbf{v}_{D(i)}|^2\delta\tau}\mathbf{v}_{D(i)} \tag{V.55}$$

depending on the time-step and on a parameter a which close to a node should be equal to 1.0. In general for open systems the value of 0.5 is used as default and does not change the results of the sampling. The parameter can be changed through the (`a_drft`) variable in the `qmcsmp` namelist (See Tab. V.2).
In the same work[57] the authors also proposed a cut-off for the local energy in the reweighting factor, that was later shown by Zen *et al.*[70] to be not size-consistent at finite time step, introducing an error in the computation of the binding energies.
In QMeCha the cut-off are chosen through the `encrr_type` variable (see Table V.3). In the code we implement three different cut-off that remodulate the values of $\mathsf{S}(\bar{\mathbf{r}}^{n,m+1})$ and $\mathsf{S}(\bar{\mathbf{r}}^{n,m})$ essentially according to the equations

$$\begin{cases} \mathsf{S}(\bar{\mathbf{r}}^{n,m}) = \mathsf{E}_R^m - \mathsf{E}_{\text{MA}}^m - \text{sgn}\left\{\Delta\mathsf{E}^{n,m}\right\}\frac{\min\{|\Delta\mathsf{E}^{n,m}|, \mathsf{E}_{\text{cut}}\}}{C^{n,m}} \\ \mathsf{S}(\bar{\mathbf{r}}^{n,m+1}) = \mathsf{E}_R^m - \mathsf{E}_{\text{MA}}^m - \text{sgn}\left\{\Delta\mathsf{E}^{n,m+1}\right\}\frac{\min\{|\Delta\mathsf{E}^{n,m+1}|, \mathsf{E}_{\text{cut}}\}}{C^{n,m+1}} \end{cases} \tag{V.56}$$

where $\Delta\mathsf{E}^{n,m} = \mathsf{E}_l(\bar{\mathbf{r}}^{n,m}) - \mathsf{E}_{\text{MA}}^m$ and $\Delta\mathsf{E}^{n,m+1} = \mathsf{E}_l(\bar{\mathbf{r}}^{n,m+1}) - \mathsf{E}_{\text{MA}}^m$ and E_{MA}^m is the mixed average estimation of the energy that is accumulated during the dynamics, and E_R^m is the reference energy, both estimated at the previous step m.
The first two cut-offs implemented are both size-extensive and they fix the value of $C^{n,m} = C^{n,m+1} = 1$. In the cut-off proposed by Zen *et al.*[70] (`encrr_type=1`) $\mathsf{E}_{\text{cut}} = g_e\sqrt{\frac{N_p}{\delta\tau}}$ depends on a parameter g_e which is by default set to 0.2 and on the number

of particles in the system. An alternative approach proposed by Ye Luo (unpublished) (`encrr_type=2`) is to set the cut-off to $\mathrm{E}_{\mathrm{cut}} = \sqrt{\frac{\mathrm{Var}[\mathrm{E}^m_{\mathrm{MA}}]}{\delta\tau}}$ depending on the square root of the variance of the local energies computed at step m. In QMeCha to avoid that both cut-offs diverge as $\delta\tau \to 0$, we impose that for $\delta\tau \leq 0.01$ $\mathrm{E}_{\mathrm{cut}} = 10\sqrt{\mathrm{Var}\left[\mathrm{E}^m_{\mathrm{MA}}\right]}$.
The final cut-off (`encrr_type=3`) proposed by Anderson and Umrigar[71,72] which is not fully size-extensive fixes the value $\mathrm{E}_{\mathrm{cut}} = 10\sqrt{\mathrm{Var}\left[\mathrm{E}^m_{\mathrm{MA}}\right]}$ and uses the reweighting constant $C^{n,m} = 1 + \left(\frac{|\bar{\mathbf{v}}_D(\bar{\mathbf{r}}^{n,m})|^2\delta\tau}{N_p}\right)^2$ that depends on the square modulus of the vector of the drift velocities at step m.

V.3.7 Reference energy

In the reweighting process the weight of the population of walkers largely fluctuates according to the value of the reference energy E_R that appears in the Green function. To keep the total weight approximately constant the reference energy is updated during the DMC run through the general expression

$$\mathrm{E}^{m+1}_R = \mathrm{E}^m_{MA} - g_r \frac{\delta\tau}{\delta\tau^m_{\mathrm{eff}}} \log \frac{W_m}{W_0} \tag{V.57}$$

where W_0 is the total initial weight (usually equal to the number of initial walkers), W_m is the total weight at the step m and g_r is a constant that is usually set to 1, that determines the response of the reference energy to the change in weights. This quantity can be changed in the `dmc` namelist (See Tab. V.3) through the input variable `g_r`. Finally, $\mathrm{E}^m_{\mathrm{MA}}$ and $\delta\tau^m_{\mathrm{eff}}$ corresponds respectively to the *mixed average* (discussed in the following sections) of the energy and of the effective time step at the mth time evolution step. At the beginning of the run it is set as the VMC energy of the trial wavefunction $\Psi_T(\bar{\mathbf{r}})$.

V.3.8 Branching procedure

As discussed in section V.3.1 after an equilibration time the FN-DMC algorithm will be sampling a wave function $f(\bar{\mathbf{r}}, \tau) = \varphi(\bar{\mathbf{r}}, \tau)\Psi_T(\bar{\mathbf{r}})$ through the distribution of $\mathcal{N}$ finite walkers each multiplied by its weight

$$f(\bar{\mathbf{r}}, \tau) \approx \sum_{i=1}^{\mathcal{N}} W^i_{\mathcal{M}} \delta(\bar{\mathbf{r}}^i_{\mathcal{M}} - \bar{\mathbf{r}}), \tag{V.58}$$

where the weights at the final time τ are equal to the the consecutive products of the weight variations

$$W^n_{\mathcal{M}} = \prod_{m=1}^{\mathcal{M}} W(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau) W^n_0, \tag{V.59}$$

being $W^n_0 = 1$ for all n.
This approach is highly inefficient since the accumulation of weights during the random walk might introduce large fluctuations for which only few walkers will dominate the overall distribution and thus will be significant in the evaluation of the observable.
An alternative and more efficient approach is to sue a *branching* or *death/birth* procedure for at each time step, or at every few time steps, the accumulated weights are used as probabilities the regulate their death or their replication.
This can be done in various ways, determined in QMeCha by the input variable `brnch_type` in namelist `dmc` (See Tab. V.3).
The most common approach (`brnch_type=1`) consists in creating a number of replicas

$$r^{n,m+1} = \mathrm{int}\left[W(\bar{\mathbf{r}}^{n,m+1}, \bar{\mathbf{r}}^{n,m}; \delta\tau) + \eta\right] \tag{V.60}$$

for each walker at a time m which are proportional to their total weight, where η is a random number in the uniform interval $(0, 1)$. Clearly, this branching procedure introduces a fluctuation in the number of walkers, some of which are destroyed while others are replicated.
In order to limit the population variation, and the increase of correlation in the sample, due to a large number of replicas, one way is to put a cut-off on the maximum number of replicas that can be created (which in QMeCha is fixed to 5) or to use the *split-join* algorithm introduced by Umrigar *et al.*[57].
In the *split-join* approach each walker with weight larger then 2 is split into $r^n_{m+1} = \mathrm{int}\left[W(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau)\right]$ replicas each with a new weight of $W^n_{m+1} = \frac{W(\bar{\mathbf{r}}^n_{m+1}, \bar{\mathbf{r}}^n_m; \delta\tau)}{r^n_{m+1}}$, at the same time, if two walkers of indexes i and j have both a weight lower than $1/2$ then they are joined in a new walker with total weight $W(\bar{\mathbf{r}}^i_{m+1}, \bar{\mathbf{r}}^i_m; \delta\tau) + W(\bar{\mathbf{r}}^j_{m+1}, \bar{\mathbf{r}}^j_m; \delta\tau)$ with a configuration chosen as i with probability $\frac{W(\bar{\mathbf{r}}^i_{m+1}, \bar{\mathbf{r}}^i_m; \delta\tau)}{W(\bar{\mathbf{r}}^i_{m+1}, \bar{\mathbf{r}}^i_m; \delta\tau) + W(\bar{\mathbf{r}}^j_{m+1}, \bar{\mathbf{r}}^j_m; \delta\tau)}$ and as j

otherwise. This algorithm conserves the total weight of the walkers while varying the number of walkers.
The advantage of the first type of algorithm with respect to the second type is in the fact that the first is "local" while the second is "non-local" in the sense that the second requires information on the overall distribution of walkers while the first only refers to the values of the weights.
An alternative approach, to the *split-join* algorithm is to keep fixed the total number of walkers by varying the weights as done in the *stochastic reconfiguration* algorithm first proposed and later generalized by Calandra and Sorella[73,74], and improved by Assaraf *et al.*[75]. This approach can avoid the memory fluctuations in large calculations and thus might be more easy to implement in an efficient way.
The idea behind the *stochastic reconfiguration* algorithm, which is selected in QMeCha with (`brnch_type=2`), is the following: at a certain step in the time evolution m, given the $\mathcal{N}$ weights of the corresponding set of walkers, we define the renormalized weights as

$$\tilde{W}^{n,m} = \frac{W^{n,m}}{\sum_{n=1}^{\mathcal{N}} W^{n,m}}. \tag{V.61}$$

so that $\sum_{n=1}^{\mathcal{N}} \tilde{W}^{n,m} = 1$. We now divide the interval $[0:1]$ in $\mathcal{N}$ segments of lengths $\tilde{W}^{n,m}$, each defined through the interval

$$l^n = [\sum_{i=0}^{n-1} \tilde{W}^{i,m} : \sum_{i=0}^{n} \tilde{W}^{i,m}] \text{ with } \tilde{W}^{0,m} = 0 \qquad \forall n \in [1:\mathcal{N}]. \tag{V.62}$$

Afterwards, $\mathcal{N}$ random numbers are extracted in the uniform interval $[0:1]$ and the number of random numbers that fall in each interval are counted, leading to the array of replicas that have to be created. Some intervals will be empty, and thus the walkers will be deleted, other intervals (the largest) will count more then one random number, thus the corresponding walkers will be replicated.
At the end, we will have $\mathcal{N}$ new walkers chosen randomly from the previous walkers with probabilities $\tilde{W}^{n,m}$. Each walker will have a new weight equal to $W^{n,m'} = \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} W^{n,m}$ which will be accumulated until the next *stochastic reconfiguration*.
The choice of $\mathcal{N}$ random numbers are extracted in the uniform interval $[0:1]$ has the disadvantage of leading to a loss of information if some large intervals end up counting zero random numbers within them. An alternative approach will be to consider z^n numbers distributed uniformly in the interval $[0:1]$ with a random shift

$$z^n = \frac{\eta}{\mathcal{N}} + \frac{n-1}{\mathcal{N}} \qquad \forall n \in [1:\mathcal{N}]. \tag{V.63}$$

where η is a random number extracted uniformly in the interval $[0:1]$. In this way all the space of the probability densities is visited in a uniform way, reducing the possible loss of information coming from the pseudo-random number extractions.
Also this procedure is actually non-local, thus it is convenient to do the branching only when the distribution of the walkers is such that it is best to drop or split some configurations. A way to do so, which is implemented in the QMeCha code, is to do the branching only if at a time m one of the weights is either $0.5 < W^{n,m}$ or $W^{n,m} > 1.5$. This check is rather fast within the code and can be done every each drift-diffusion step without significant computational cost. If the condition is true then the branching is done. Clearly, if the sampling is stable, the result is that as the discretization $\delta\tau$ diminishes, so thus the frequency of the branching which is thus proportional to the inverse of $\delta\tau$.
This type of approach is selected in QMeCha by setting `n_bra=0` (default for DMC, see Tab. V.1), otherwise for `n_bra` ≥ 1 the code branches every `n_bra` drift-diffusion steps of all the particles.

V.3.9 Singularities near the nuclear cusps

The introduction of the rescaling factor in the *drift velocities* to correct the divergence in the nodal surfaces as the effect of introducing a bias in the regions near the nuclear cusps in which the drift velocity is at most of the order of the nuclear charge. In order to avoid this bias the a constant in eq. V.55 can be made position dependent[57], and defined as:

$$a(\mathbf{r}) = \frac{1}{2}\left(1 + \frac{\mathbf{v}_{D(i)}}{|\mathbf{v}_{D(i)}|} \cdot \frac{\mathbf{r} - \mathbf{R}_{min}}{|\mathbf{r} - \mathbf{R}_{min}|}\right) + \frac{Z_{min}^2 (\mathbf{r} - \mathbf{R}_{min})^2}{10\left[4 + Z_{min}^2 (\mathbf{r} - \mathbf{R}_{min})^2\right]} \tag{V.64}$$

where $\mathbf{R}_{min}$ is the position of the nearest nucleus and Z_{min} is its nuclear charge. This last term corresponds essentially to a space-mapping that is nucleus dependent.

When an electron is close to the nucleus the drift velocity is always directed towards it, yet an electron should not be able while drifting to move through the nucleus itself. Thus to avoid overshooting of the nucleus it is possible to redefine the drift process in the follwing way: We define the distance from the nucleus to the electron as

$$\mathbf{r} - \mathbf{R}_{min} = r\hat{\mathbf{r}}_{\parallel} \tag{V.65}$$

and decompose the *drift velocity* in the component parallel to the electron-nucleus direction $\hat{\mathbf{r}}_{\parallel}$ and orthogonal $\hat{\mathbf{r}}_{\perp}$ to it:

$$\tilde{\mathbf{v}}_{D(i)} = v_{\parallel}\hat{\mathbf{r}}_{\parallel} + v_{\perp}\hat{\mathbf{r}}_{\perp}. \tag{V.66}$$

The displacement caused by the remodulated drift will be compute in cilindrical coordinates as:

$$\begin{aligned} \delta r'_{\parallel} &= \max\left[r + v_{\parallel}\delta\tau, 0\right] \\ \delta r'_{\perp} &= \frac{2v_{\perp}\delta\tau\delta r'_{\parallel}}{r+\delta r'_{\parallel}} \end{aligned} \tag{V.67}$$

and the new electronic position will be:

$$\mathbf{r}' = \mathbf{R}_{min} + \delta r'_{\parallel}\hat{\mathbf{r}}_{\parallel} + \delta r'_{\perp}\hat{\mathbf{r}}_{\perp}. \tag{V.68}$$

This new position still needs to be diffused according to the gaussian in the Green function. Yet in the case of bare nucleus the Green function should have at the nuclear position. When an electron crosses the nucleus we want it to diffuse according to a probability density that is related to the 1s cusped orbital, approximated as $^{-2Z|\eta|}$ (where $|\bar{\eta}|$ is a three dimensional vector of random displacements). The probability that an electron drifts to the nucleus and diffuses across it is given by:

$$\tilde{q} = 1 - \tilde{p} = \frac{1}{2}\mathrm{erfc}\left(\frac{r + v_{\parallel}\delta\tau}{\sqrt{2\delta\tau}}\right), \tag{V.69}$$

thus with probability $\tilde{p}$ we choose to sample the diffusion according to the Gaussian move

$$g_1(\delta\mathbf{r}_1) = \left(\frac{1}{2\pi\delta\tau}\right)^{3/2} e^{-\frac{|\delta\mathbf{r}_1|^2}{2\delta\tau}} \tag{V.70}$$

so that the new position will be

$$\mathbf{r}' = \mathbf{R}_{min} + \delta r'_{\parallel}\hat{\mathbf{r}}_{\parallel} + \delta r'_{\perp}\hat{\mathbf{r}}_{\perp} + \delta\mathbf{r}_1 \tag{V.71}$$

and with probability $\tilde{p}$ we sample the exponential distribution[76]:

$$g_2(\delta\mathbf{r}_2) = \frac{\zeta^3}{\pi}e^{-2\zeta|\delta\mathbf{r}_2|} \qquad \zeta = \sqrt{Z^2_{min} + \frac{1}{\delta\tau}} \tag{V.72}$$

and the new diffused position will be

$$\mathbf{r}' = \mathbf{R}_{min} + \delta\mathbf{r}_2. \tag{V.73}$$

By considering the two step approach the Green function of the diffusion part has to be modified by mixing the two probability densities according two the equation:

$$G(\mathbf{r}' \leftarrow \mathbf{r}) = \tilde{p}g_1(\delta\mathbf{r}_1) + \tilde{q}g_2(\delta\mathbf{r}_2) \tag{V.74}$$

and in the acceptance probability we will need to compute the ratio $G(\mathbf{r} \leftarrow \mathbf{r}')/G(\mathbf{r}' \leftarrow \mathbf{r})$ where the numerator is the Green function of the inverse process, for which we will need to compute the probability of diffusing randomly to $\mathbf{r}$ from an intermediate position drifted from $\mathbf{r}'$.

V.4 Diffusion Monte Carlo calculations

A DMC calculation is specified in the `qmcsmp` namelist by setting `qmc_mthd='dmc'`. The calculation requires the definition of some of the same variables used for VMC, and in particular the number of walkers per mpi task `n_wlk`, the thermalization steps `n_dmc_trm` the number of blocks `n_bin` in which we divide the time evolution and the length of the blocks `bin_l`. In DMC the last three quantities are particularly important during the thermalization process, since during the thermalization the averages are not allocated and each block is used independently to update quantities such as the energy average, the reference energy and the effective time step. Finally the `dt` is used as the variable that regulates the discretization of time. The total time of the projection is thus `dt`$\times$`bin_l` $\times$(`n_trm` $+$`n_bin`) where the first `n_trm` are discarded and not used for the computation of the observables.

The dmc calculations can require some optional variables that are specified in the **dmc** namelist (see Tab. V.3) which are, in particular, the branching method **brnch_type** and the energy cut-off method **encrr_type** discussed in respectively in sec. V.3.8 and sec. V.3.6.
In the following examples we will address some particular cases for DMC calculations.

V.4.1 Diffusion Monte Carlo for pseudo H_2O

Let us consider at first the all-electron Oxygen atom. The minimal input to run a DMC calculation with the variable population branching executed after each drift-diffusion step and Zen's cut-off will be written as

Input for a Diffusion Monte Carlo calculation

```
&qmcsmp
    qmc_mthd ='dmc'
    n_wlk = 80
    bin_l = 50
    n_bra = 0
    n_trm = 10
    n_bin  = 100
    dt = 0.100
/
&dmc
  n_trm_dmc = 25
        g_e = 0.2
 brnch_type = 2
 encrr_type = 1
/
&basset
  fle_basis = 'h2o_qmecha.bas'
  fle_pspot = 'pseudo.dat'
/
&molsys
  fle_coord = 'h2o_qmecha.xyz'
/
```

where we also specify the value of the Zen's energy cut-off constant `g_e = 0.2`.
At the start of the DMC run refers QMeCha prints the details of the fermionic dynamics which is fixed to the 3d Langevin dynamics with a fixed time step of 0.10.
These details are followed by the sampling properties of which in particular we need to check the number of thermalization steps (which now refer to the DMC thermalization steps) and the total number of walkers that refer to the population that samples the electronic distribution at each time step.
The third section refers to the parameters of the DMC run defined by input or during the initialization in which the code summarizes the total time of the projection, the discretization, the type of branching and of energy cut-off.

Diffusion Monte Carlo parameters

```
                Diffusion Monte Carlo parameters

 Total number of thermalization steps (n_trm_dmc*bin_l) :          1250
 Total number of sampling steps        (bin_l*n_blocks) :          5000
 Time discretization step                       (dt)[au] :  0.100000E+00
 Thermalization time                               [au] :  0.125000E+03
 Sampling time                                     [au] :  0.500000E+03
 Total target weight of the configurations           :  0.128000E+05
 Reference energy rescaling factor                [Eh] :  0.100000E+01
 Stochastic Reconfiguration with fixed popu(brnch_type) :             2
 Number of configurational updates between branc(n_bra) :             0
 ZSGMA 2016 energy cut-off                (encrr_type) :             1
 Cut-off rescaling factor                        (g_e) :  0.200000E+00
```

At first the code does a VMC calculation to thermalize the electronic configuration according to the trial wave function, this step is common to the VMC algorithm with the only difference that the time step is kept fixed.

Variational Thermalization

```
          Variational Monte Carlo thermalization

      Block Num      Acc. rate. [%]      Time step [au]
        1              91.079          0.10000000E+00
        2              91.063          0.10000000E+00
        .                .                   .
        .                .                   .
        .                .                   .
```

Table V.3: *Complete List of variables in the* `dmc` *namelist.*

Variable	Type	Descrition
`n_trm_dmc`	int32	Number of DMC thermalization steps (default is 10).
`w_chk`	real64	Value that defines the interval $[1.0 - \texttt{w\_chk} : 1.0 + \texttt{w\_chk}]$ outside which the code branches if `n_bra`=0.
`g_e`	real64	α factor for energy cut-off from Zen et al.[70]
`brnch_type`	int32	Type of branching method (Default is 2 Stochastic reconfiguration with fixed walker population). 1 uses the traditional method with variable population and 0 is no branching.
`encrr_type`	int32	Methods for energy cut-offs in branching factor. If 1 use Zen's corrections[70] for which $E_{cut} = \texttt{g\_e}\sqrt{\frac{N}{dt}}$. If 2 (default) uses Ye Lou's correction that defines the cut-off energy as $E_{cut} = \frac{\sigma_E}{\sqrt{dt}}$, where σ_E is the standard deviation of the mixed Energy. If 3 uses Umrigar's 2021 cut-off[71,72]. If 0 no corrections.(See sec. V.3.6 for details)

After the run the code computes the VMC energy which is used as the initial reference energy of the run, and gives a correlated estimation of the variance of the wave function.

Reference energy and initial VMC guess

```
E_VMC            [Eh] : -0.17222350322485E+02 +/- 0.65182924297773E-03
Var[E_VMC]     [Eh^2] :  0.27192407168059E+00 +/- 0.48069876317905E-03

Initialized reference Energy                   [Eh] : -0.172338E+02
```

From now on the code starts the true DMC run, with the first 25 thermalization steps and for each block it prints in particular the energy estimation in the block sampling `E_b` the corresponding variance `Var_b` and the rescaled total weight (in units of 1).

Thermalization Blocks

```
Thermalization block                                :             1

E_b              [Eh] : -0.17241694945551E+02 +/- 0.65012839312579E-03
Var[E_b]       [Eh^2] :  0.26972064839090E+00 +/- 0.47749801339761E-03

Number of branchings                                :            21
Reference Energy                                [Eh] : -0.172364E+02
Average effective time step                     [au] :  0.100000E+00
Total number of configurations                      :         12800
Rescaled total weight                               :  0.999910E+00
Acceptance rate of Monte Carlo moves for elec.   [%] :  0.910336E+02
Block time                                    [sec.] :  0.408000E+00
```

After these first `n_trm` steps are executed, the algorithm starts accumulating the mixed averages of the observable that are also used in the evaluation/stabilization of the reference energy and of the effective time step.

Accumulation Blocks

```
Accumulation block                                  :           100

E_b              [Eh] : -0.17244390601797E+02 +/- 0.65650368461555E-03
Var[E_b]       [Eh^2] :  0.27514209164451E+00 +/- 0.48700231215072E-03
E_MA             [Eh] : -0.17242808345201E+02 +/- 0.64922814123953E-04
Var[E_MA]      [Eh^2] :  0.26869188058302E+00 +/- 0.47592619504275E-04

Number of branchings                                :            24
Reference Energy                                [Eh] : -0.172387E+02
Average effective time step                     [au] :  0.861165E-01
Total number of configurations                      :         12800
Rescaled total weight                               :  0.100199E+01
Acceptance rate of Monte Carlo moves for elec.   [%] :  0.910504E+02
Block time                                    [sec.] :  0.177000E+00
```

Notice that these first steps conserve the total number of configurations, in order to avoid population explosions in this phase of the integration, while the total weight is varied.

Please also notice that the number of branchings executed in the block may vary if `n_bra`=0 in input (default value). This is because the code branches according to an

internal threshold `w_chk` (default value is 0.5) that sets the minimum value of a weight $1.0 - $ `w_chk` and the maximum value of a weight $1.0 +$ `w_chk` within which the weights are evolved without branching. If even one configuration falls outside these limits the branching is activated. This also means that for small time steps the branching is nearly never attempted.
In this case the length of a block was `bin_l=50` and the number of branching was only 21 for a time step of 0.1 au.
This automatic branching frequency can be controlled through the `n_bra` input variable, by setting `n_bra>0`. With `n_bra=1` the code attempts to branch every time.
At the end of the run, the code generates automatically the `dmc_eval.dat` file that collects the error analysis for the mixed average estimations with the reblocking procedure, which is the same as the one discussed for the VMC calculations.

V.5 Wave function optimization

As anticipated in the section dedicated to VMC, given the variational wavefunction $\Psi_T(\mathbf{x};\boldsymbol{\alpha})$ it is possible to optimize the $\boldsymbol{\alpha}$ set of $N_{\boldsymbol{\alpha}}$ parameters through the *variational principle* of the energy functional defined in eq. V.1 or by exploiting the *zero variance principle*, thus minimizing respectively the energy or the variance or a combination of both[77,78,79,80,81,82,83,84,85,86,87,88,89].
Although, the sole variance minimization is particularly indicated in the case in which only the parameters of the Jastrow factor have to be optimized (see Section IV.2) it has been shown that wavefunctions fully optimized by variance minimization lack multiplicative separability[88], and when applied to excited states' optimization the procedures might lead to divergences[89].
For this reason, in this work, we only consider energy minimization within the VMC framework, where the basic quantities that have to be computed are the *'generalized forces'*, *ie* the set of partial derivatives

$$\mathbf{f}_{\boldsymbol{\alpha}} = -\frac{\partial}{\partial \boldsymbol{\alpha}} E\left[\Psi_T(\boldsymbol{\alpha})\right]. \tag{V.75}$$

of eq. **??** with respect to the vector $\boldsymbol{\alpha}$. Considering that the expectation value of the first derivative with respect to a real parameter of the local value of any Hermitian operator, such as the Hamiltonian, is always zero[80], *ie* $\int \frac{\partial E_l(\mathbf{x};\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \Pi(\mathbf{x};\boldsymbol{\alpha}) d\mathbf{x} = 0$, the full expression of the generalized forces is given by

$$\mathbf{f}_{\boldsymbol{\alpha}} = -2\left\{ \int E_l(\mathbf{x};\boldsymbol{\alpha}) \mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) \Pi(\mathbf{x};\boldsymbol{\alpha}) d\mathbf{x} - \right. \\ \left. - \int E_l(\mathbf{x};\boldsymbol{\alpha}) \Pi(\mathbf{x};\boldsymbol{\alpha}) d\mathbf{x} \int \mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) \Pi(\mathbf{x};\boldsymbol{\alpha}) d\mathbf{x} \right\} \tag{V.76}$$

which can be rewritten as

$$\mathbf{f}_{\boldsymbol{\alpha}} = -2\left\{ \langle E_l(\mathbf{x};\boldsymbol{\alpha}) \mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) \rangle_{\Pi(\boldsymbol{\alpha})} - \right. \\ \left. - \langle E_l(\mathbf{x};\boldsymbol{\alpha}) \rangle_{\Pi(\boldsymbol{\alpha})} \langle \mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) \rangle_{\Pi(\boldsymbol{\alpha})} \right\}, \tag{V.77}$$

where $\mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) = \frac{1}{\Psi_T(\mathbf{x};\boldsymbol{\alpha})} \frac{\partial \Psi_T(\mathbf{x},\boldsymbol{\alpha})}{\partial \alpha_k} = \frac{\partial}{\partial \alpha_k} \ln[\Psi_T(\mathbf{x},\boldsymbol{\alpha})]$ is the vector of $N_{\boldsymbol{\alpha}}$ components that can be also expressed through the local operators $\hat{\mathbf{O}}_{\boldsymbol{\alpha}} = \delta_{\mathbf{x},\mathbf{x}'} \frac{1}{\Psi_T(\mathbf{x}';\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\alpha}}$ in bra-ket formalism as $\mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha}) = \langle \mathbf{x} | \hat{\mathbf{O}}_{\boldsymbol{\alpha}} | \Psi_T(\boldsymbol{\alpha}) \rangle$.
The knowledge of the generalized forces allows one to minimize the energy functional through the simple steepest-descent (SD) method[90] where the parameters are progressively updated as

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \boldsymbol{\delta\alpha} \qquad \boldsymbol{\delta\alpha} = \delta\tau \mathbf{f}_{\boldsymbol{\alpha}} \tag{V.78}$$

being $\delta\tau$ an arbitrary small damping parameter used to guarantee the stability of the evolution. In the case of all 'equivalent' parameters, such as the coefficients of a linear combination, for sufficiently small values of $\delta\tau$, the SD method always converges towards the variational minimum. Yet, when the wavefunction contains highly non-linear parameters, such as the exponentials of the basis set or the parameters of the Jastrow factor (see Section **??**) the SD method becomes highly inefficient since small changes in some parameters could correspond to large changes in the total energy and vice versa. A way to solve this issue is to define a new metric that takes into account the non-linearity of the parameters, as done for example by the Stochastic Reconfiguration (SR) method that introduces the metric of the variational wavefunction[74,81,26].

V.5.1 Stochastic reconfiguration

The Stochastic Reconfiguration (SR) method was first introduced by S. Sorella in refs. 74,81 to cure the sign problem in the $J_1 - J_2$ Heisenberg model, and was later generalized as an optimization method for atomic and molecular wave functions in refs. 26,27. In order to obtain the derivation of the SR method, we can recall that in lattice dynamics[90,74,81], the projection in imaginary time of an initial eigenstate $|\Psi_T(\boldsymbol{\alpha})\rangle = |\Psi_T(\tau)\rangle$ of the Hamiltonian, with non-zero overlap with the ground state, is obtained through the repeated application of the linear projection operator $\left[\hat{\mathbb{I}} - \delta\tau\left(\hat{\mathrm{H}} - \hat{\mathbb{I}}E_R\right)\right]$

$$|\Psi_T(\tau+\delta\tau)\rangle = \left[\hat{\mathbb{I}}(1+\delta\tau E_R) - \delta\tau\hat{\mathrm{H}}\right]|\Psi_T(\tau)\rangle\,, \tag{V.79}$$

where $\hat{\mathbb{I}}$ is the identity operator, $\delta\tau$ is the imaginary time discretization, that must be kept small enough to stabilize the diffusion process, and E_R is the reference energy, that is updated during the run and is used to conserve the normalization of the wave function, exactly as in the DMC algorithm[49]. For sufficiently long time-projections this procedure converges to the exact ground state of the system, at least for bosonic systems.
If we now consider the initial parametric wave function $\Psi_T(\mathbf{x};\boldsymbol{\alpha})$, we can write its first order update for a small change $\boldsymbol{\delta\alpha}$ of the parameter vector $\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \boldsymbol{\delta\alpha}$ as

$$\Psi_T(\mathbf{x};\boldsymbol{\alpha}') = \Psi_T(\mathbf{x};\boldsymbol{\alpha}) + \\ + \boldsymbol{\delta\alpha}^\top\left(\frac{\partial\Psi_T(\mathbf{x};\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}}\right) + \mathcal{O}(\boldsymbol{\delta\alpha}^2), \tag{V.80}$$

that can be written in bra-ket formalism through the local operators $\hat{\mathbf{O}}_{\boldsymbol{\alpha}}$ defined in sec. V.5 as

$$\left|\Psi_T(\boldsymbol{\alpha}')\right\rangle = \left[\hat{\mathbb{I}} + \boldsymbol{\delta\alpha}^\top\hat{\mathbf{O}}_{\boldsymbol{\alpha}}\right]|\Psi_T(\boldsymbol{\alpha})\rangle\,. \tag{V.81}$$

The basic idea behind the SR method is to map the projection of the wave function during the time step $\delta\tau$ over the direction given by the application of the $\hat{\mathbf{O}}_{\boldsymbol{\alpha}}$ operators on the initial state. This can be obtained by imposing the constraints

$$\frac{\langle\Psi_T(\boldsymbol{\alpha})|\,\hat{O}_{\alpha_k}\,|\Psi_T(\tau+\delta\tau)\rangle}{\langle\Psi_T(\boldsymbol{\alpha})|\,\Psi_T(\tau+\delta\tau)\rangle} = \\ = \frac{\langle\Psi_T(\boldsymbol{\alpha})|\,\hat{O}_{\alpha_k}\,|\Psi_T(\boldsymbol{\alpha}')\rangle}{\langle\Psi_T(\boldsymbol{\alpha})|\,\Psi_T(\boldsymbol{\alpha}')\rangle} \quad \forall \quad k \in [1, N_{\boldsymbol{\alpha}}] \tag{V.82}$$

for each of the $N_{\boldsymbol{\alpha}}$ parameters, which reduce to the linear system[81,91]

$$\mathbf{S}\boldsymbol{\delta\alpha} = \frac{\delta\tau}{2}\mathbf{f}_{\boldsymbol{\alpha}} \tag{V.83}$$

where $\frac{\delta\tau}{2}$ is a constant factor that, the vector $\mathbf{f}_{\boldsymbol{\alpha}}$ represents the generalized forces defined in eq. V.77, and $\mathbf{S}$ is the covariance matrix of the $\mathbf{O}_{\boldsymbol{\alpha}}(\mathbf{x};\boldsymbol{\alpha})$ local quantities, of elements

$$\mathbf{S}_{k,q} = \left\langle \mathrm{O}_{\alpha_k}(\mathbf{x};\boldsymbol{\alpha})\mathrm{O}_{\alpha_q}(\mathbf{x};\boldsymbol{\alpha})\right\rangle_{\Pi(\boldsymbol{\alpha})} - \\ - \left\langle \mathrm{O}_{\alpha_k}(\mathbf{x};\boldsymbol{\alpha})\right\rangle_{\Pi(\boldsymbol{\alpha})}\left\langle \mathrm{O}_{\alpha_q}(\mathbf{x};\boldsymbol{\alpha})\right\rangle_{\Pi(\boldsymbol{\alpha})}. \tag{V.84}$$

Thus, including the factor 2 into the arbitrary value of $\delta\tau$ we can see that the parameter variation for the SR method becomes

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \delta\tau\mathbf{S}^{-1}\mathbf{f}_{\boldsymbol{\alpha}} \tag{V.85}$$

which is similar to the SD case in eq. V.78, with the only difference that, while the SD method works in the Cartesian metric $\delta_{\boldsymbol{\alpha}} = \boldsymbol{\delta\alpha}^\top\boldsymbol{\delta\alpha}$ the SR method works in the metric of the wavefunction's parametric space $\delta_{\boldsymbol{\alpha}} = \boldsymbol{\delta\alpha}^\top\mathbf{S}\boldsymbol{\delta\alpha}$ determined by the covariance of the local $\hat{\mathbf{O}}_{\boldsymbol{\alpha}}$ operators, that is related to Fisher's information matrix[92] and the Fubini-Study metric of Hermitian operators[93,94].

V.5.2 Regularization of the optimizations

The bare variance of the forces defined in eq. V.77 is unbounded since the integrand diverges as the electronic configuration approaches a nodal point of the wave function. Many different regularization schemes have been introduced in the literature to cure

this problem and give finite variances and covariances[95,96,97]. Here we use the simple approach proposed by S. Pathak and L. K.Wagner that consists of introducing a regularization function in the estimation of the local quantities that constitute the generalized forces[98] with a constant fixed at $\epsilon = 0.01$.
Furthermore, in order to guarantee a stable solution of the linear systems obtaining a well-conditioned value of the parameters' variations it is essential to introduce some regularizations in the correlation or covariance matrices. First, it has been extensively discussed how the matrix **S** that appears in the SR has to be regularized[81,91,99] in order to guarantee invertibility. This happens because two or more rows and columns of these positive and symmetric matrices might not be fully independent, especially in the case of redundant parameters. Second, since the eigenvalues of the matrices can be very small, this might lead to additional instability in the inversion procedure or in the parallel Conjugate Gradient schemes that are used to solve the linear systems of equations. To solve both issues a simple way[81,91], following the approach used in the Levenberg-Marquardt method[100,99], is to add a positive shift to the diagonal elements of the matrices $\mathbf{S} \rightarrow \mathbf{S} + \epsilon\mathbf{I}$, where ϵ is a small constant and **I** is the identity matrix. As ϵ grows the direction of the parameters' variation $\boldsymbol{\delta\alpha}$ rotates towards that given by the SD method. Traditionally for the SR method the value of the shift can be set to $\epsilon = 0.001$, yet it also depends on the energy scales of the matrix eigenvalues that depend on the type of parameters that are being optimiaed, so in general, be careful.
Another issue comes from the fact that the scale of the eigenvalues can be quite different due to the non-linearity of the parameters. For this reason, it is useful to precondition the matrices before inversion, as suggested in ref. 91.

V.6 Wave function optimization calculations

The QMeCha code implements only the Stochastic reconfiguration optimization method, that can be done with and without Correlated Sampling. The wave function optimization is implemented only in the framework of VMC, since the optimization with DMC is usually unstable.
QMeCha automatically activates the optimization process if the namelist `&optmet` is present in the main input.

In Tab. V.4 we report all the parameters that are included in that namelist, and explain their action.
In the next sections we will describe the minimal parameters that must be set to optmize the wave function of the pseudo water molecule without entering further specific details.

V.6.1 Wave function optimization for the H_2O molecule

The simplest input for wave function optmization of some of the Jastrow parameters is written as

Minimal input for an optimization

```
&qmcsmp
         n_wlk = 1
         bin_l = 10
         n_bin = 400
/
&optmet
 lin_solv_mthd = 'cg'
            da = 0.6d-1
           eps = 0.1d-2
    n_opt_step = 50
    n_crs_step = 50
       jc1_opt = .true.
       jd2_opt = .true.
        je_opt = .true.
       restart = .true.
/
```

The first namelist (`&qmcsmp`) refers to the VMC sampling used to compute the quantities necessary for the optimiation process, at each step. The explnation of these variables can be found in the VMC chapter. Through the namelist `&optmet` optimization is automatically activated.
The first variable `lin_solv_mthd = 'cg'` is used to define the procedure to solve the SR linear system. If the system is small, it is ideal to use `lin_solv_mthd = 'lu'` since usually the MPI communication of the covariance matrices **S** to the rank 0 core is usually

a fast procedure and the LU decomposition of the preconditioned matrix is fast.
The values here set for the dampling paramter `da` and the matrix regularization shift `eps` are also reasonable values if the parameters are far from the minimum, and when Correlated sampling is activated.
In this case, the code runs `n_opt_step = 50` total optimization steps, attempting `n_crs_step = 50` consecutive Correlated Sampling optimization steps. That is to say that the code attempts to run one single VMC sampling at the beginning, and to use CS to estimate the new statistical quantities using the same sample during the full run. This is automatically handle by the code, as we will explain when explaining the output.
The variables `jc1_opt = .true.`, `jd2_opt = .true.` and `je_opt = .true.` identify which set of parameters have to be optimized: The parameters of the one body cusp and the linear coefficients of the Gaussian expansion, the linear parameters of the dynamical Jastrow matrix and the exponents and linear coefficients of the basis set of the Jastrow factor.
The `restart = .true.` flag is essential to force the code to read the `wvfn.save` otherwise the code reinitializes the wave function from scratch overwriting the initial optimal parameters.

V.6.2 Output of the optimization

In the output file of the optimization process, after the details about the initial parameters of the VMC sampling used to compute the various statistical quantities

Sampling details

```
                    Variational Monte Carlo sampling

 3D Gaussian Transition prob. (dt*I covariance).
 Two step sampling activated

 Initial Metropolis step for electrons          (dt_e) :  0.507000E+00
 Variable metropolis step                      (var_dt) :             T
```

the code prints the details about the main optimization parameters, and in particular of the damping parameter `da` and the matrix diagonal shift `eps`.

Parameters of the optimization

```
 Parameter step amplitude                              (da) :  0.600000E-01
 Parameter for matrix regularization                  (eps) :  0.100000E-02
 Parameter for force regularization        (eps_force_reg) :  0.100000E-02
 Maximum force considered in optimization       (force_cut) :  0.100000E-07
 Normalization limit                             (norm_cut) :  0.100000E+00
 Minimum Normalization limit                     (norm_min) :  0.100000E-02
 Alpha normalization mix                        (alpha_norm) :  0.000000E+00
```

Of the other parameters, the most important are the cut-off for the regularization of the forces near the nodes `eps_force_reg`, please refer to refs. 98,97, and the force cut-off `force_cut` that is a threshold below which the forces are considered null. Finally, `norm_cut` is a parameter introduces to avoid that between two consecutive optimization steps the norm of the wave function is changed more than a certain threshold, slowing down the optimization otherwise. This section is followed by the list that includes the types and number of parameters to optimize

Wave function parameters to optimize

```
 Optimizing One Body cusp                        (jc1_opt) :             T
 Optimizing dynamical two body Jastrow terms     (jd2_opt) :             T
 Optimizing Jastrow orbitals coefficients         (jc_opt) :             T
 Optimizing Jastrow orbitals exponentials         (je_opt) :             T

 Number of elec. determinant/pfaffian parameters           :             0
 Number of posi. determinant/pfaffian parameters           :             0
 Number of elec.-posi. geminal parameters                  :             0
 Number of atomic orbitals parameters                      :             0
 Number of positronic orbitals parameters                  :             0

 Number of cusp parameters                                 :            12
 Number of dynamical Jastrow parameters                    :           312
 Number of Jastrow orbitals parameters                     :            24

 Total number of ferimionic parameters                     :             0
 Total number of Jastrow parameters                        :           348

 Total number of parameters to optimize                    :           348
 Total number of parameters after symmetrization           :           138
```

In this section, it is clear that the total number of parameters are automatically reduced by symmetry, so only 138 independent parameters will be included in the optimization. All derivatives are computed, the symmetrization is a process that is done after the parameters' derivatives are computed for each configuration. Afterwards, the code starts the VMC thermalization in which the optimal time step is initialized.

Variational Monte Carlo thermalization

```
            Variational Monte Carlo thermalization

      Block N°      Acc. rate. [%]      Time step [au]
        1              34.125           0.13264961E+00
        2              58.750           0.19418551E+00
        .                 .                  .
        .                 .                  .
```

After the initialization of the sampling, the optimization starts with a new sampling and by printing for each optimization step with independent MC sampling te section

Single Optimization step

```
 Optimization step                                    :            1

 Energy           [Eh] : -0.16778974022951E+02 +/- 0.12545039095730E-01
 Variance       [Eh^2] :  0.14285425980583E+01 +/- 0.44353965502239E-04
 Number of signal active parameters                   :          114
 Maximum force deviation                              :  0.788682E+02
 CG solver: accuracy set to                           :  0.100000E-08
 CG solver: number of steps to convergence            :          312
 CG solver: residual error                            :  0.855158E-09
 CG solver: execution time                     [sec.] :  0.550000E+00
 WARNING!!! Slowing down to preserve norm. change     :  0.469474E-01
 Wavefunction norm change                             :  0.100000E+00
 Projected energy change                              : -0.218027E+00
 Signal-to-noise ratio                                :  0.593510E+00
 Final parameter step used                      (da)  :  0.469474E-01
 Acceptance rate of Monte Carlo moves for elec.  [%]  :  0.501505E+02
 Changing Time step for next sampling for elec.       :  0.249918E+00
 Optimization step time                        [sec.] :  0.150700E+01
```

At first the code writes the energy and the variance for the given set of parameters. Afterwards it prints 'the number of active parameters' that are the number of parameters above the threshold `force_cut`, and the 'maximum force deviation' defined as $\max\left[\frac{|\mathbf{f}_\alpha|}{\sigma_{\mathbf{f}_\alpha}}\right]$ that as rule of the thumb during the optimization should converge below ≈ 4 to signal convergence.

These information are followed by the details of the convergence of the CG procedure (this does not appear when using LU decomposition).

Notice that if, as in this case, the optimization step would exceed the change in the norm given by the `norm_cut` `da` is reduced to slow down the optimization of the wave function.

Afterwards, some details regarding the sampling are printed. We can see that the code still tries to optimize the MC time step to converge the acceptance rate to 50% which is the rule of the thumb to reduced correlation.

In this optimization procedure, we have used Correlated Sampling, thus, after a first step with traditional sampling, the code reuses the previous MC sampled configurations to estimate the statistical MC quantities on the wave function with the new set of parameters.

In the Correlated Sampling optimization step

Single Correlated Sampling Optimization step

```
 Correlated sampling optimization step                :            2

 Wave functions overlap                               :  0.995344E+00
 Energy           [Eh] : -0.16957426379147E+02 +/- 0.96529495726056E-02
 Variance       [Eh^2] :  0.86896849901954E+00 +/- 0.68257544557496E-02
 Number of signal active parameters                   :          130
 Maximum force deviation                              :  0.688534E+02
 CG solver: accuracy set to                           :  0.100000E-08
 CG solver: number of steps to convergence            :          347
 CG solver: residual error                            :  0.736366E-09
 CG solver: execution time                     [sec.] :  0.611000E+00
 Wavefunction norm change                             :  0.904502E-01
 Projected energy change                              : -0.140891E+00
 Signal-to-noise ratio                                :  0.508344E+00
 Final parameter step used                      (da)  :  0.600000E-01
 Optimization step time                        [sec.] :  0.139200E+01
```

the code first writes the overlap between the two wave functions, that is an important measure to understand if resampling of the configurations is needed. The code automatically resamples if the overlap is < 0.99.
After this additional detail is given the details that are given afterwards are the same of the normal optimization step.
In Tab. V.4 we report all the main variables that can be configured for the optimization.

Table V.4: *Complete List of variables in the* `optmet` *namelist.*

Variable	Type	Descrition
`n_opt_step`	int32	(Default `0_int32`) Total number of optimization steps.
`n_crs_step`	int32	(Default `0_int32`) Number of consecutive correlated sampled steps.
`n_avg_step`	int32	(Default `1_int32`) Number of last steps on which the parameters can be averaged.
`da`	real64	(Default `0.005_dp`) Damping parameter for the optimization process.
`eps`	real64	(Default `0.001_dp`) Diagonal matrix shift for regularization.
`restart`	logical	(Default `.false.`) Flag for restart, imposes to read the wave function in the `wvfn.save` directory.
`jc1_opt`	logical	(Default `.false.`) Optimization of the one-body cusp main parameters and linear parameters of the Gaussian expansion.
`jc2_opt`	logical	(Default `.false.`) Optimization of the two-body cusp main parameters and linear parameters of the Gaussian expansion.
`jce_opt`	logical	(Default `.false.`) Optimization of the non-linear parameters of the Gaussian expansions in the one- and two-body cusps.
`jd_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the one and -three -four body Jastrow factors. (Corresponds to `jd1_opt = .true.` and `jd2_opt = .true.`).
`jd1_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the one body Jastrow factor.
`jd2_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the -three -four body Jastrow factor.
`je_opt`	logical	(Default `.false.`) Optimization of the linear and non linear parameters of the Jastrow basis sets.
`jc_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the Jastrow basis sets.
`de_opt`	logical	(Default `.false.`) Optimization of the linear and non linear parameters of the atomic basis sets in the determinantal part.
`dc_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the atomic basis sets in the determinantal part.
`dl_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the determinants.
`ql_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the QDO wave functions (or those of the dipole wave function).
`qc_opt`	logical	(Default `.false.`) Optimization of the linear parameters of the QDO basis set (for the Hartree product).
`qe_opt`	logical	(Default `.false.`) Optimization of the non-linear parameters of the QDO basis set (for the Hartree product).
`jqe_opt`	logical	(Default `.false.`) Optimization of the dipole coupling between electronic subsystem and QDO environment.
`el_opt`	logical	(Default `.true.`) Optimization of the electronic part of the wave function.
`po_opt`	logical	(Default `.true.`) Optimization of the positronic part of the wave function.
`norm_cut`	real64	(Default `0.1_dp`) Maximum value for the variation of the norm of the wave function set to avoid jumps.
`force_cut`	real64	(Default `.false.`) Minimum value of the forces on the parameters, otherwise the dimension is disregarded.
`lin_solv_mthd`	char(2)	(Default `'lu'`) Linear solver for the SR method, LU decomposition (`'lu'`) or conjugate gradients (`'cg'`).
`eps_force_reg`	real64	(Default `0.001_dp`) Regularization of the forces to avoid divergences near the nodes.
`cg_acc`	real64	(Default `10.0d-8_dp`) Precision of the CG iterations.

Bibliography

(1) Burkatzki, M.; Filippi, C.; Dolg, M. Energy-consistent pseudopotentials for quantum Monte Carlo calculations. *J. Chem. Phys.* **2007**, *126*, 234105.

(2) Burkatzki, M.; Filippi, C.; Dolg, M. Energy-consistent small-core pseudopotentials for 3d-transition metals adapted to quantum Monte Carlo calculations. *J. Chem. Phys.* **2008**, *129*, 164115.

(3) Trail, J. R.; Needs, R. J. Norm-conserving Hartree–Fock pseudopotentials and their asymptotic behavior. *J. Chem. Phys.* **2005**, *122*.

(4) Trail, J. R.; Needs, R. J. Smooth relativistic Hartree-Fock pseudopotentials for H to Ba and Lu to Hg. *J. Chem. PZur Quantentheorie der Molekelnhys.* **2005**, *122*.

(5) Trail, J. R.; Needs, R. J. Comparison of Smooth Hartree-Fock Pseudopotentials. *J. Chem. Theory Comput.* **2014**, *10*, 2049–2053.

(6) Trail, J. R.; Needs, R. J. Correlated electron pseudopotentials for 3d-transition metals. *J. Chem. Phys.* **2015**, *142*, 064110.

(7) Krogel, J. T.; Santana, J. A.; Reboredo, F. A. Pseudopotentials for quantum Monte Carlo studies of transition metal oxides. *Phys. Rev. B* **2016**, *93*, 075143.

(8) Trail, J. R.; Needs, R. J. Shape and energy consistent pseudopotentials for correlated electron systems. *J. Chem. Phys.* **2017**, *146*, 204107.

(9) Bennett, M. C.; Melton, C. A.; Annaberdiyev, A.; Wang, G.; Shulenburger, L.; Mitas, L. A new generation of effective core potentials for correlated calculations. *The Journal of Chemical Physics* **2017**, *147*, 224106.

(10) Bennett, M. C.; Wang, G.; Annaberdiyev, A.; Melton, C. A.; Shulenburger, L.; Mitas, L. A new generation of effective core potentials from correlated calculations: 2nd row elements. *J. Chem. Phys.* **2018**, *149*, 104108.

(11) Annaberdiyev, A.; Wang, G.; Melton, C. A.; Bennett, M. C.; Shulenburger, L.; Mitas, L. A new generation of effective core potentials from correlated calculations: 3d transition metal series. *J. Chem. Phys.* **2018**, *149*, 134108.

(12) Wang, G.; Annaberdiyev, A.; Melton, C. A.; Bennett, M. C.; Shulenburger, L.; Mitas, L. A new generation of effective core potentials from correlated calculations: 4s and 4p main group elements and first row additions. *J. Chem. Phys.* **2019**, *151*, 144110.

(13) Drummond, N. D.; Towler, M. D.; Needs, R. J. Jastrow correlation factor for atoms, molecules, and solids. *Phys. Rev. B* **2004**, *70*, 235119.

(14) Petruzielo, F. R.; Toulouse, J.; Umrigar, C. J. Compact and flexible basis functions for quantum Monte Carlo calculations. *J. Chem. Phys.* **2010**, *132*, 094109.

(15) Petruzielo, F. R.; Toulouse, J.; Umrigar, C. J. Basis set construction for molecular electronic structure theory: Natural orbital and Gauss–Slater basis for smooth pseudopotentials. *J. Chem. Phys.* **2011**, *134*, 064104.

(16) Morales, M. A.; McMinis, J.; Clark, B. K.; Kim, J.; Scuseria, G. E. Multideterminant Wave Functions in Quantum Monte Carlo. *J. Chem. Theory Comput.* **2012**, *8*, 2181–2188.

(17) Zen, A.; Luo, Y.; Sorella, S.; Guidoni, L. Molecular Properties by Quantum Monte Carlo: An Investigation on the Role of the Wave Function Ansatz and the Basis Set in the Water Molecule. *J. Chem. Theory Comput.* **2013**, *9*, 4332–4350.

(18) Barborini, M.; Coccia, E. Investigating Disjoint Non-Kekulé Diradicals with Quantum Monte Carlo: The Tetramethyleneethane Molecule through the Jastrow Antisymmetrized Geminal Power Wave Function. *J. Chem. Theory Comput.* **2015**, *11*, 5696–5704.

(19) Barborini, M.; Guidoni, L. Ground State Geometries of Polyacetylene Chains from Many-Particle Quantum Mechanics. *J. Chem. Theory Comput.* **2015**, *11*, 4109–4118.

(20) Bajdich, M.; Mitas, L.; Wagner, L. K.; Schmidt, K. E. Pfaffian pairing and backflow wavefunctions for electronic structure quantum Monte Carlo methods. *Phys. Rev. B* **2008**, *77*, 115112.

(21) Marchi, M.; Azadi, S.; Casula, C.; Sorella, S. Resonating valence bond wave function with molecular orbitals: Application to first-row molecules. *J. Chem. Phys.* **2009**, *131*, 154116.

(22) Cayley, A. Sur les déterminants gauches. (Suite du Mémoire T. XXXII. p. 119). *J. Reine Angew. Math.* **1849**, *38*, 93–96.

Bibliography

(23) Rubow, J.; Wolff, U. A factorization algorithm to compute Pfaffians. *Comp. Phys. Commun.* **2011**, *182*, 2530 – 2532.

(24) Bajdich, M.; Mitas, L.; Drobný, G.; Wagner, L. K.; Schmidt, K. E. Pfaffian Pairing Wave Functions in Electronic-Structure Quantum Monte Carlo Simulations. *Phys. Rev. Lett.* **2006**, *96*, 130201.

(25) (a) Coleman, A. J. Structure of Fermion Density Matrices. *Rev. Mod. Phys.* **1963**, *35*, 668–687; (b) Coleman, A. J. Structure of Fermion Density Matrices II. Antisymmetrized Geminal Powers. *J. Math. Phys* **1965**, *6*, 1425–1431.

(26) Casula, M.; Sorella, S. Geminal wave functions with Jastrow correlation: A first application to atoms. *J. Chem. Phys.* **2003**, *119*, 6500–6511.

(27) Casula, M.; Attaccalite, C.; Sorella, S. Correlated geminal wave function for molecules: An efficient resonating valence bond approach. *J. Chem. Phys.* **2004**, *121*, 7110.

(28) Huang, C.-J.; Filippi, C.; Umrigar, C. J. Spin contamination in quantum Monte Carlo wave functions. *J. Chem. Phys.* **1998**, *108*, 8838–8847.

(29) Becker, M. S.; Broyles, A. A.; Dunn, T. A Parametric Approach to the Ground-State Energy of an Electron Gas. *Phys. Rev.* **1968**, *175*, 224–228.

(30) Fahy, S.; Wang, X. W.; Louie, S. G. Variational quantum Monte Carlo nonlocal pseudopotential approach to solids: Formulation and application to diamond, graphite, and silicon. *Phys. Rev. B* **1990**, *42*, 3503–3522.

(31) Bressanini, D.; Mella, M.; Morosi, G. Stability and positron annihilation of positronium hydride $L = 0, 1, 2$ states: A quantum Monte Carlo study. *Phys. Rev. A* **1998**, *57*, 1678–1685.

(32) Kita, Y.; Maezono, R.; Tachikawa, M.; Towler, M.; Needs, R. J. Ab initio quantum Monte Carlo study of the positronic hydrogen cyanide molecule. *J. Chem. Phys.* **2009**, *131*, 134310.

(33) Charry, J.; Romero, J.; Varella, M. T. d. N.; Reyes, A. Calculation of positron binding energies of amino acids with the any-particle molecular-orbital approach. *Phys. Rev. A* **2014**, *89*, 052709.

(34) Charry, J.; Varella, M. T. d. N.; Reyes, A. Binding Matter with Antimatter: The Covalent Positron Bond. *Angewandte Chemie Int. Ed.* **2018**, *57*, 8859–8864.

(35) Reyes, A.; Moncada, F.; Charry, J. The any particle molecular orbital approach: A short review of the theory and applications. *Int. J. Quantum Chem.* **2019**, *119*, e25705.

(36) Ito, S.; Yoshida, D.; Kita, Y.; Tachikawa, M. First-principles quantum Monte Carlo studies for prediction of double minima for positronic hydrogen molecular dianion. *J. Chem. Phys.* **2020**, *153*, 224305.

(37) Bressanini, D.; Mella, M.; Morosi, G. Positronium chemistry by quantum Monte Carlo. I. Positronium-first row atom complexes. *J. Chem. Phys.* **1998**, *108*, 4756–4760.

(38) Brown, R.; Prigent, Q.; Swann, A. R.; Gribakin, G. F. Effective radius of ground- and excited-state positronium in collisions with hard walls. *Phys. Rev. A* **2017**, *95*, 032705.

(39) Sadhukhan, M.; Manby, F. R. Quantum mechanics of Drude oscillators with full Coulomb interaction. *Phys. Rev. B* **2016**, *94*, 115106.

(40) Jones, A. P.; Crain, J.; Sokhan, V. P.; Whitfield, T. W.; Martyna, G. J. Quantum Drude oscillator model of atoms and molecules: Many-body polarization and dispersion interactions for atomistic simulation. *Phys. Rev. B* **2013**, *87*, 144103.

(41) Cipcigan, F. S.; Crain, J.; Sokhan, V. P.; Martyna, G. J. Electronic coarse graining: Predictive atomistic modeling of condensed matter. *Rev. Mod. Phys.* **2019**, *91*, 025003.

(42) Ditte, M.; Barborini, M.; Sandonas, L. M.; Tkatchenko, A. Molecules in Environments: Toward Systematic Quantum Embedding of Electrons and Drude Oscillators. *Phys. Rev. Lett.* **2023**, *0*, 0.

(43) Jones, A.; Thompson, A.; Crain, J.; Müser, M. H.; Martyna, G. J. Norm-conserving diffusion Monte Carlo method and diagrammatic expansion of interacting Drude oscillators: Application to solid xenon. *Phys. Rev. B* **2009**, *79*, 144119.

(44) Bade, W. L. Drude-Model Calculation of Dispersion Forces. I. General Theory. *J. Chem. Phys.* **1957**, *27*, 1280–1284.

(45) Kato, T. On the eigenfunctions of many-particle systems in quantum mechanics. *Commun. Pure Appl. Math.* **1957**, *10*, 151–177.

(46) Jastrow, R. Many-Body Problem with Strong Forces. *Phys. Rev.* **1955**, *98*, 1479–1484.

(47) Padé, H. Sur la représentation approchée d'une fonction par des fractions ra-

tionnelles. *Ann. Sci. Éc. Norm. Supér.* **1892**, *9*, 3–93.

(48) Kolorenč, J.; Mitas, L. Applications of quantum Monte Carlo methods in condensed systems. *Rep. Prog. Phys.* **2011**, *74*, 026502.

(49) Foulkes, W. M. C.; Mitas, L.; Needs, R. J.; Rajagopal, G. Quantum Monte Carlo simulations of solids. *Rev. Mod. Phys.* **2001**, *73*, 33–83.

(50) Umrigar, C. J. In *Quantum Monte Carlo Methods in Physics and Chemistry*; Nightingale, M. P., Umrigar, C. J., Eds.; NATO ASI Series, Series C, Math. & Phys. Sciences; Kluwer Academic Publishers: Boston, 1999; Vol. C-525.

(51) Ceperley D. M. In *Solving quantum many-body problems with random walks in Comp. Phys. , Proc. Ninth Physics Summer School, Australian National University*, world scientific pub. co. ed.; Gardner, E. H. J., Savage, C. M., Eds.; 1997.

(52) Umrigar, C. J.; Wilson, K. G.; Wilkins, J. W. Optimized trial wave functions for quantum Monte Carlo calculations. *Phys. Rev. Lett.* **1988**, *60*, 1719–1722.

(53) Umrigar, C. J.; Filippi, C. Energy and Variance Optimization of Many-Body Wave Functions. *Phys. Rev. Lett.* **2005**, *94*, 150201.

(54) Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **1953**, *21*, 1087–1092.

(55) Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **1970**, *57*, 97–109.

(56) Everett, C.; Cashwell, E. Third Monte Carlo sampler. Revision and extension of samplers I and II. **1983**,

(57) Umrigar, C. J.; Nightingale, M. P.; Runge, K. J. A diffusion Monte Carlo algorithm with very small time-step errors. *J. Chem. Phys.* **1993**, *99*, 2865–2890.

(58) Ceperley, D.; Chester, G. V.; Kalos, M. H. Monte Carlo simulation of a many-fermion study. *Phys. Rev. B* **1977**, *16*, 3081–3099.

(59) Umrigar, C. J. Accelerated Metropolis method. *Phys. Rev. Lett.* **1993**, *71*, 408–411.

(60) Scemama, A.; Lelièvre, T.; Stoltz, G.; Cancès, E.; Caffarel, M. An efficient sampling algorithm for variational Monte Carlo. *J. Chem. Phys.* **2006**, *125*, 114105.

(61) Dewing, M. Improved efficiency with variational Monte Carlo using two level sampling. *The Journal of Chemical Physics* **2000**, *113*, 5123–5125.

(62) Kosztin, I.; Faber, B.; Schulten, K. Introduction to the diffusion Monte Carlo method. *Am. J. Phys.* **1996**, *64*, 633–644.

(63) Mitas, L. In *Quantum Monte Carlo Methods in Physics and Chemistry*; Nightingale, M. P., Umrigar, C. J., Eds.; NATO ASI Series, Series C, Math. & Phys. Sciences; Kluwer Academic Publishers: Boston, 1999; Vol. C-525.

(64) Sakurai J. J. In *Meccanica Quantistica Moderna*, ii ed. ed.; Zanichelli, Ed.; 1996.

(65) Reynolds P. J.; Ceperley D. M.; Alder J. B.; Lester W. A. Fixed-node quantum Monte Carlo for molecules. *J. Chem. Phys.* **1982**, *77*, 5593–5603.

(66) Fye R. M. Study of Trotter-like Approximations. *J. Stat. Phys.* **1986**, *43*, 827–843.

(67) Bressanini D.; Ceperley D. M.; Reynolds P. In *What do we know about wave function nodes?, in Recent Advances in Quantum Monte Carlo Methods*, ii ed. ed.; S. Rothstein, W. S., Ed.; 2001.

(68) Ceperley D. M. Fermion Nodes. *J. Stat. Phys.* **1991**, *63*, 1237–1266.

(69) DePasquale, M. F.; Rothstein, S. M.; Vrbik, J. Reliable diffusion quantum Monte Carlo. *J. Chem. Phys.* **1988**, *89*, 3629–3637.

(70) Zen, A.; Sorella, S.; Gillan, M. J.; Michaelides, A.; Alfè, D. Boosting the accuracy and speed of quantum Monte Carlo: Size consistency and time step. *Phys. Rev. B* **2016**, *93*, 241118.

(71) Anderson, T. A.; Umrigar, C. J. Nonlocal pseudopotentials and time-step errors in diffusion Monte Carlo. *J. Chem. Phys.* **2021**, *154*, 214110.

(72) Anderson, T. A.; Umrigar, C. J. Erratum: "Nonlocal pseudopotentials and time-step errors in diffusion Monte Carlo". *J. Chem. Phys.* **2021**, *155*, 079901.

(73) M. Calandra Buonaura; Sorella S. Numerical study of the two-dimensional Heisenberg model using a Green function Monte Carlo technique with a fixed number of walkers. *Phys. Rev. B* **1998**, *57*, 11446.

(74) Sorella, S.; Capriotti, L. Green function Monte Carlo with stochastic reconfiguration: An effective remedy for the sign problem. *Phys. Rev. B* **2000**, *61*, 2599–2612.

(75) Assaraf, R.; Caffarel, M.; Khelif, A. Diffusion Monte Carlo methods with a fixed number of walkers. *Phys. Rev. E* **2000**, *61*, 4566–4575.

(76) Kalos, M. H.; Whitlock, P. A. *Monte Carlo Methods*; John Wiley & Sons, Ltd, 2009; Chapter 3, pp 35–76.

(77) Bianchi, R.; Bressanini, D.; Cremaschi, P.; Mella, M.; Morosi, G. Wave-function optimization by least-squares fitting of the exact wave function sampled by quantum

Monte Carlo. *Int. J. Quantum Chem.* **1996**, *57*, 321–325.
(78) Harju A.; Barbiellini B.; Siljam aki S.; Nieminen R. M. Stochastic Gradient Approximation: An Efficient Method to Optimize Many-Body Wave Functions. *Phys. Rev. Lett.* **1997**, *79*, 1173.
(79) Kent P. R. C.; Needs R. J.; Rajagopal G. Monte Carlo energy and variance-minimization techniques for optimizing many-body wave functions. *Phys. Rev. B* **1999**, *59*, 12344.
(80) Lin, X.; Zhang, H.; Rappe, A. M. Optimization of quantum Monte Carlo wave functions using analytical energy derivatives. *J. Chem. Phys.* **2000**, *112*, 2650–2654.
(81) Sorella, S. Generalized Lanczos algorithm for variational quantum Monte Carlo. *Phys. Rev. B* **2001**, *64*, 024512.
(82) Nightingale, M. P.; Melik-Alaverdian, V. Optimization of Ground- and Excited-State Wave Functions and van der Waals Clusters. *Phys. Rev. Lett.* **2001**, *87*, 043401.
(83) Bressanini, D.; Morosi, G.; Mella, M. Robust wave function optimization procedures in quantum Monte Carlo methods. *J. Chem. Phys.* **2002**, *116*, 5345–5350.
(84) Drummond N. D.; Needs R. J. Variance-minimization scheme for optimizing Jastrow factors. *Phys. Rev. B* **2005**, *72*, 85124.
(85) Sorella, S. Wave function optimization in the variational Monte Carlo method. *Phys. Rev. B* **2005**, *71*, 241103.
(86) Scemama, A.; Filippi, C. Simple and efficient approach to the optimization of correlated wave functions. *Phys. Rev. B* **2006**, *73*, 241101.
(87) Umrigar, C. J.; Toulouse, J.; Filippi, C.; Sorella, S.; Hennig, R. G. Alleviation of the Fermion-Sign Problem by Optimization of Many-Body Wave Functions. *Phys. Rev. Lett.* **2007**, *98*, 110201.
(88) Toulouse J.; Umrigar C. J. Optimization of quantum Monte Carlo wave functions by energy minimization. *J. Chem. Phys.* **2007**, *126*, 84102.
(89) Cuzzocrea, A.; Scemama, A.; Briels, W. J.; Moroni, S.; Filippi, C. Variational Principles in Quantum Monte Carlo: The Troubled Story of Variance Minimization. *J. Chem. Theory Comput.* **2020**, *16*, 4203–4212.
(90) Koonin, S. *Computational Physics: Fortran Version*; CRC Press, 1990.
(91) Becca, F.; Sorella, S. *Quantum Monte Carlo Approaches for Correlated Systems*; Cambridge University Press, 2017.
(92) Fisher, R. A. In *Breakthroughs in Statistics: Methodology and Distribution*; Kotz, S., Johnson, N. L., Eds.; Springer New York: New York, NY, 1992; pp 66–70.
(93) Fubini, G. Sulle metriche definite da una forma Hermitiana. *Atti del Reale Istituto Veneto di Scienze, Lettere ed Arti* **1904**, *63*, 501–513.
(94) Study, E. Kürzeste Wege im komplexen Gebiet. *Mathematische Annalen* **1905**, *60*, 321–378.
(95) Attaccalite, C.; Sorella, S. Stable Liquid Hydrogen at High Pressure by a Novel Ab Initio Molecular-Dynamics Calculation. *Phys. Rev. Lett.* **2008**, *100*, 114501.
(96) Trail, J. R. Heavy-tailed random error in quantum Monte Carlo. *Phys. Rev. E* **2008**, *77*, 016703.
(97) van Rhijn, J.; Filippi, C.; De Palo, S.; Moroni, S. Energy Derivatives in Real-Space Diffusion Monte Carlo. *J. Chem. Theory Comput.* **2022**, *18*, 118–123.
(98) Pathak, S.; Wagner, L. K. A light weight regularization for wave function parameter gradients in quantum Monte Carlo. *AIP Advances* **2020**, *10*, 085213.
(99) Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Indust. Appl. Math.* **1963**, *11*, 431–441.
(100) Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.* **1944**, *2*, 164–168.