openLOWDIN

Release 0.1

openLOWDIN

Apr 16, 2025

CONTENTS:

openLOWDIN is a computational program that implements the Any Particle Molecular Orbital (APMO) method to study systems containing any type and number of quantum species, such as electrons, positrons, quantum nuclei, muons, or drude oscillators.

This manual is still in early construction stage! Thanks for your patience.

CHAPTER

ONE

ABOUT

openLOWDIN is a computational program that implements the Any Particle Molecular Orbital (APMO) method to study systems containing any type and number of quantum species, such as electrons, positrons, quantum nuclei, muons, or drude oscillators.

At present, openLOWDIN code is publicly available at https://github.com/efposadac/openLOWDIN

1.1 Capabilities

What can we do in Lowdin? The current version of the code encompasses the following quantum chemistry methods:

- HF
- DFT
- MP2
- CI (CIS, CISD, CIST, CISDTQ, FCI, CIPSI)
- PT (PT2, PT3, PP3, RENPP3, OVGF)
- NOCI

Check more details in the *Code* and *Tutorials*

1.2 Documentation

The online manual of openLOWDIN is available at https://github.com/openLOWDIN/openLOWDIN_manual A compiled pdf version of the manual can be found here `openlowdin.pdf`

1.3 How to cite:

Please cite the code as:

R.i Flores-Moreno, E. Posada, F. Moncada, J. Romero, J. Charry, M. Díaz-Tinoco, S.A. González, N.F. Aguirre, A. Reyes, LOWDIN: The any particle molecular orbital code. Int. J. Quantum Chem.. 114. (1), 50–56 (2014).

1.4 Acknowledgements:

This code results from an evolution of the APMO and LOWDIN software packages, both mainly developed at the Universidad Nacional de Colombia. Currently, the code is under an open-source initiative thanks to the support of some initial developers, who are presently spread around the world!

CHAPTER

TWO

DEVELOPERS AND CONTRIBUTORS

2.1 Active developers (alphabetical order)

- Andrés Reyes
- Edwin Posada
- Jorge Charry
- Félix Moncada

2.2 Contributors (alphabetic order)

- Jhonathan Romero (APMO/Propagators)
- Sergio Gonzalez (APMO: Older versions)
- Nestor Aguirre (APMO: Older versions)
- Danilo González Forero (APMO/COSMO)
- Jose Mauricio Rodas Rodriguez (APMO/(QM/MM))
- Carlos Ortiz-Mahecha (APMO/CC)
- Alejandro Peña Torres (APMO/CC)
- Laura Pedraza-González (APMO/core)
- Manuel Diaz (APMO/Propagators)
- Teresa Tamayo-Mendoza (Development of higher order propagator methods)
- Roberto Flores-Moreno (ADFT/ADPT/Propagators)

CHAPTER

THREE

INSTALLATION

Compile: (see below for a step-by-step example)

- run *./configure* in LOWDIN root directory. Be sure that you have permissions to write in the installation directory and have properly exported the *$PATH* environment.
- run *make*

Install:

- run *make install*

Uninstall

- run *make uninstall*

Documentation

- run *make doc*

The *make doc* command produces both latex and html documentation using doxygen program. Be sure you have installed doxygen, for instance in a debian-based distribution run:

 # apt-get install doxygen graphviz

To use latex documentation in doc/latex folder, run command:

pdflatex refman.tex

To visualize the html documentation use:

<web browser> doc/html/index.html

Clean the project

- run *make clean* and then *make distclean*

—

Step-by-step installation example: (replace apt-get with your preferred package manager)

sudo apt-get update sudo apt-get -y install wget git build-essential liblapack-dev libblas-dev libgsl0-dev autotools-dev automake libtool gfortran python3 gawk libeigen3-dev libgmp-dev libboost-all-dev # Define ENV Variables export WORKDIR=$PWD/dependencies export PATH=$PATH:$WORKDIR/bin export C_INCLUDE_PATH=$C_INCLUDE_PATH:$WORKDIR/include:$WORKDIR/include/libint2:/usr/include/eigen3 export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:$WORKDIR/include:$WORKDIR/include/libint2:/usr/include/e export LIBRARY_PATH=$LIBRARY_PATH:$WORKDIR/lib export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WORKDIR/lib # Create work directories mkdir $WORKDIR mkdir $WORKDIR/bin mkdir $WORKDIR/lib cd $WORKDIR

Libint2 # If you have Ubuntu, you can get this precompiled Libint2 library wget https://www.dropbox.com/s/d3d44j238lkfwcr/libint-master-SEP052019.tgz tar xzvf libint-master-SEP052019.tgz

Otherwise, download and compile with minimal (default am), G12, fPIC options (libint2 commit 668b10c4bdca5876984058742d4212675eb93f3f) # git clone https://github.com/evaleev/libint.git # cd libint # git checkout 668b10c4bdca5876984058742d4212675eb93f3f # ./autogen.sh # mkdir ../build # cd ../build # ../libint/configure –prefix=$WORKDIR –with-max-am=6 –enable-g12=4 –with-g12-max-am=4 –with-cxxgen-optflags # make -j 4 # make install # ../libint/configure –prefix=$WORKDIR

cd -

Libint1 git clone https://github.com/evaleev/libint.git cd libint git checkout v1 aclocal -I lib/autoconf autoconf ./configure –prefix=$WORKDIR make -j 4 make install make clean make distclean cd -

Libxc cd $WORKDIR # If you have Ubuntu, you can get this precompiled Libxc library wget https://www.dropbox.com/s/6cja3zzhl1cq46i/libxc-master-MAY242023.tgz tar xzvf libxc-master-MAY242023.tgz # Otherwise, download and compile with default options (libxc commit 4bd0e1e36347c6d0a4e378a2c8d891ae43f8c951) # git clone https://gitlab.com/libxc/libxc.git # cd libxc # git checkout 4bd0e1e36347c6d0a4e378a2c8d891ae43f8c951 # autoreconf -i # ./configure –enable-shared –prefix=$WORKDIR # make -j 4 # make install

cd ..

Configure Lowdin ./configure -p $WORKDIR/bin -s /tmp -l “-lblas -llapack” # Build Lowdin make -j 4 # Install Lowdin make install # Run Tests make test

CHAPTER

FOUR

GETTING STARTED

Let's get ready to run openLOWDIN. Here you can find the basic information about the input and how to run the code. A more exhaustive description of all code keywords and files can be found in section

4.1 Input file

The code requires an text `input` file with extension `.lowdin`. Here is an example of a minimum `input` file

```
GEOMETRY
e-[O]   6-31G                0.0000         0.0000         0.1173 multiplicity=2␣
↪addParticles=-1
e-[H]   6-31G                0.0000         0.7572        -0.4692
e-[H]   6-31G                0.0000        -0.7572        -0.4692
u-        13S.ET.O.u.TF        0.0000         0.0000         0.1173
O        dirac                0.0000         0.0000         0.1173
H-a_1   DZSPDN                0.0000         0.7572        -0.4692 m = 1836.15
H-b_2        DZSPDn                0.0000        -0.7572         0.4692 m = 1836.15
END GEOMETRY

TASKS
        method = "UHF"
        propagatorTheoryCorrection = 2
END TASKS

CONTROL
        iterationScheme = 3
        convergenceMethod = 1
        readCoefficients = F
        hartreeProductGuess = T
END CONTROL
```

The minimum required blocks to run a calculation are `GEOMETRY}`, `TASKS`, and `CONTROL`.

The `GEOMETRY` block provides the information needed to build the molecular system. The first column declares the type of the quantum species. As shown in the above example, `e-[H]` and `e-[O]` define the electrons of a Hydrogen and a Oxygen atom respectively; `U-` defines a negative muon, `O\_{16}`, `H\_1` and `H\_2` define ^{16}O, ^{1}H and ^{2}H nuclei respectively.

The second column declares the basis sets. When the `dirac` basis is chosen, the particle is treated as a classical point charge. The third, fourth and fifth columns declare the x,y,z coordinates of the particle basis set center.

The sixth column provides additional information via keywords `addParticles` and `multiplicity`. These keywords are used to change the default values. `addParticles` is used to modify the number of particles of a quantum species. As shown in the provided example, one electron is removed from the system. `multiplicity` defines the multiplicity for open shell calculations. In the example, an electronic multiplicity of 2 was chosen.

4.2 How to run

To run LOWDIN

```
lowdin2 -i inputname.lowdin
```

This will generate a plain text output file called `inputname.out`

CHAPTER

FIVE

CODE

This sections summarizes the multicomponent version of the common quantum chemistry methods and capabilities implemented in openLOWDIN, as well as of all available input code keywords.

CHAPTER

SIX

INPUT

CHAPTER

SEVEN

CONTROL

CHAPTER

EIGHT

SCRATCH

CHAPTER

NINE

OUTPUTS

CHAPTER

TEN

LIB

CHAPTER

ELEVEN

INTEGRALS

CHAPTER

TWELVE

HARTREE-FOCK, HF

CHAPTER

THIRTEEN

SCF

CHAPTER

FOURTEEN

PROPERTIES

CHAPTER

FIFTEEN

DENSITY FUNCTIONAL THEORY, DFT

CHAPTER

SIXTEEN

MANY-BODY PERTURBATION THEORY, MBPT

CHAPTER

SEVENTEEN

PROPAGATHOR THEORY, PT

CHAPTER

EIGHTEEN

CONFIGURATION INTERACTION, CI

The APMO/CI wave function is written as a linear combination of CI configurations between all quantum species

$$|\Phi_0\rangle = c_0|\Psi_0\rangle + \sum_{\alpha}\sum_{ia\in\alpha} c_i^a|\Psi_i^a\rangle + \sum_{\alpha,\beta}\sum_{\substack{ia\in\alpha\\ jb\in\beta}} c_{ij}^{ab}|\Psi_{ij}^{ab}\rangle + \sum_{\alpha,\beta}\sum_{\substack{ia\in\alpha\\ jb\in\alpha\\ kc\in\beta}} c_{ijk}^{abc}|\Psi_{ijk}^{abc}\rangle + \cdots \tag{18.1}$$

CHAPTER

NINETEEN

NON-ORTHOGONAL CONFIGURATION INTERACTION, NOCI

CHAPTER

TWENTY

TUTORIALS

In this section you can find multiples examples for openLOWDIN

CHAPTER

TWENTYONE

POSITRONIC SYSTEMS

This is an example on how to run a *Configuration Interaction, CI* calculations for PsH

```
GEOMETRY
        e-(H)   SHARON-E-6S2P         0.00        0.00         0.00 addParticles=1
        H        dirac                        0.00        0.00         0.00
        e+        SHARON-E+6S2P        0.00        0.00        0.00
END GEOMETRY

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
        !configurationInteractionLevel ="CISD"
END TASKS

CONTROL
readCoefficients=F
numberOfCIstates=3
CINaturalOrbitals=T
        CIStatesToPrint = 1
        !CIdiagonalizationMethod = "DSYEVX"
        CIdiagonalizationMethod = "JADAMILU"
        !CIPrintEigenVectorsFormat = "NONE"
        CIPrintEigenVectorsFormat = "OCCUPIED"
        !CIPrintEigenVectorsFormat = "ORBITALS"
        CIPrintThreshold = 5e-2
        buildTwoParticlesMatrixForOneParticle=T
END CONTROL

INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
        species="E+" core=0 active=0
END INPUT_CI
```

CHAPTER

TWENTYTWO

POSITRON COVALENT BOND

This is an example on how to compute the binding energy of a dihydride positron-bound system, using *Configuration Interaction, CI* calculations, as was done in https://doi.org/10.1002/anie.201800914

$$H^- e^+ H^- --> PsH + H^-$$

This input computes the energy of the dihydride system

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(e+H2^2-): -1.279680 a.u.

SYSTEM_DESCRIPTION='e+H2^2- from Charry 2018 (10.1002/anie.201800914)'

!add two electrons (one for each hydrogen anion)
!remove one positron
GEOMETRY
        e-(H)   AUG-CC-PVDZ         0.00        0.00         -1.6 addParticles=1
        e-(H)   AUG-CC-PVDZ         0.00        0.00         1.6 addParticles=1
        e+        E+-H-AUG-CC-PVDZ        0.00        0.00        -1.6
        e+        E+-H-AUG-CC-PVDZ        0.00        0.00        1.6 addParticles=-1
        H        dirac                0.00        0.00         -1.6
        H        dirac                0.00        0.00         1.6
END GEOMETRY

!method to solve the SCF - CI only works for unrestricted reference
!CI level strings chooses the desired excitations to be included. FCI is all possible␣
↪excitations

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
        !configurationInteractionLevel ="CIS","CISD","CISDT","CISDTQ"
END TASKS

!Compute only the "numberOfCIstates" states. Here we only need the ground state
!Compute the density matrix for "CIstatesToPrint" states, for density outputs
!Generate the natural orbitals, for visualization in molden files
!The Davidson diagonalization implemented in JADAMILU is the recomended method.
!For small systems, full matrix diagonalization with DSYEVX is possible
!CI EigenVectors with coefficient higher than "CIPrintThreshold" are printed
```

(continues on next page)

(continued from previous page)

```
!Printing format "OCCUPIED" shows the coefficients, "ORBITALS" shows the strings, "NONE"␣
↪skips printing
!Strict SCF convergence improves the quality of the CI results (not required for the FCI)

CONTROL
        numberOfCIstates=1
        CIStatesToPrint=1
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
        !CIdiagonalizationMethod = "DSYEVX"
        CIPrintEigenVectorsFormat = "OCCUPIED"
        !CIPrintEigenVectorsFormat = "NONE","ORBITALS"
        CIPrintThreshold = 5e-2
        !totalEnergyTolerance=1E-12
END CONTROL

!INPUT_CI block help us define the frozen core and active virtuals orbitals. Here we are␣
↪not restricting the excitation space
INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
        species="POSITRON" core=0 active=0
END INPUT_CI

!With CI, moldenFiles, 1D and 2D density slices and density cubes are good ways to␣
↪visualize the density results
OUTPUTS
        moldenFile state=1
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1
        densityPlot dimensions=3 point1=0.0 -6.0 -6.0 point2=0.0 -6.0 6.0 point3=0.0 6.0␣
↪-6.0 state=1
END OUTPUTS
```

Then, we have to subtract the energy obtained from calculations of the dissoc

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(PsH): -0.734559

SYSTEM_DESCRIPTION='PsH from Charry 2018 (10.1002/anie.201800914)'

GEOMETRY
        e-(H)    AUG-CC-PVDZ          0.00         0.00          0.00 addParticles=1
        e+         E+-H-AUG-CC-PVDZ         0.00         0.00         0.00
        H         dirac                 0.00         0.00          0.00
END GEOMETRY
```

(continues on next page)

(continued from previous page)

```
!method to solve the SCF - CI only works for unrestricted reference
!CI level strings chooses the desired excitations to be included. FCI is all possible␣
↪excitations

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
        !configurationInteractionLevel ="CIS","CISD","CISDT","CISDTQ"
END TASKS

!Compute only the "numberOfCIstates" states. Here we only need the ground state
!Compute the density matrix for "CIstatesToPrint" states, for density outputs
!Generate the natural orbitals, for visualization in molden files
!The Davidson diagonalization implemented in JADAMILU is the recomended method.
!For small systems, full matrix diagonalization with DSYEVX is possible
!CI EigenVectors with coefficient higher than "CIPrintThreshold" are printed
!Printing format "OCCUPIED" shows the coefficients, "ORBITALS" shows the strings, "NONE"␣
↪skips printing
!Strict SCF convergence improves the quality of the CI results (not required for the FCI)

CONTROL
        numberOfCIstates=1
        CIStatesToPrint=1
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
        !CIdiagonalizationMethod = "DSYEVX"
        !CIdiagonalizationMethod = "ARPACK"
        CIPrintEigenVectorsFormat = "OCCUPIED" !"NONE","ORBITALS"
        CIPrintThreshold = 5e-2
        !totalEnergyTolerance=1E-12
END CONTROL

!INPUT_CI block help us define the frozen core and active virtuals orbitals. Here we are␣
↪not restricting the excitation space
INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
        species="POSITRON" core=0 active=0
END INPUT_CI

!With CI, moldenFiles, 1D and 2D density slices and density cubes are good ways to␣
↪visualize the density results
OUTPUTS
        moldenFile state=1
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1
END OUTPUTS
```

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(H-): -0.524029

SYSTEM_DESCRIPTION='H- from Charry 2018 (10.1002/anie.201800914)'

GEOMETRY
        e-(H)   AUG-CC-PVDZ         0.00        0.00         0.00 addParticles=1
        H        dirac                0.00        0.00         0.00
END GEOMETRY

!method to solve the SCF - CI only works for unrestricted reference
!CI level strings chooses the desired excitations to be included. FCI is all possible␣
↪excitations

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
        !configurationInteractionLevel ="CIS","CISD","CISDT","CISDTQ"
END TASKS

!Compute only the "numberOfCIstates" states. Here we only need the ground state
!Compute the density matrix for "CIstatesToPrint" states, for density outputs
!Generate the natural orbitals, for visualization in molden files
!The Davidson diagonalization implemented in JADAMILU is the recomended method.
!For small systems, full matrix diagonalization with DSYEVX is possible
!CI EigenVectors with coefficient higher than "CIPrintThreshold" are printed
!Printing format "OCCUPIED" shows the coefficients, "ORBITALS" shows the strings, "NONE"␣
↪skips printing
!Strict SCF convergence improves the quality of the CI results (not required for the FCI)

CONTROL
        numberOfCIstates=1
        CIStatesToPrint=1
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
        !CIdiagonalizationMethod = "DSYEVX"
        !CIdiagonalizationMethod = "ARPACK"
        CIPrintEigenVectorsFormat = "OCCUPIED" !"NONE","ORBITALS"
        CIPrintThreshold = 5e-2
        !totalEnergyTolerance=1E-12
END CONTROL

!INPUT_CI block help us define the frozen core and active virtuals orbitals. Here we are␣
↪not restricting the excitation space
INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
END INPUT_CI

!With CI, moldenFiles, 1D and 2D density slices and density cubes are good ways to␣
↪visualize the density results
```

(continues on next page)

(continued from previous page)

```
OUTPUTS
        moldenFile state=1
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1
END OUTPUTS
```

CHAPTER

TWENTYTHREE

QUANTUM NUCLEI

CHAPTER

TWENTYFOUR

NEGATIVE MUONS

CHAPTER

TWENTYFIVE

QUANTUM DRUDE OSCILLATORS, QDOS

CHAPTER

TWENTYSIX

EXTERNAL POTENTIALS

CHAPTER

TWENTYSEVEN

INDICES AND TABLES

- genindex
- modindex
- search