NDELOC 1.2.17

a Program to Compute Multicenter Electron Delocalization Indices

Author:	e-mail:
Nicolás Otero Martínez	`nom05@uvigo.es`
Marcos Mandado Alonso	`mandado@uvigo.es`

Nicolás Otero Martínez   Marcos Mandado Alonso

May 12, 2022

Universidade Vigo

Contents

1 NDELOC scheme and software

This document is a short guide to use NDELOC, a program developed as a parallelized computation of N-center electron (de)localization (localization and delocalization) indices, in which $N \geq 2$, and the "centers" represent atoms or regions obtained from a molecular real space partitioning (such as QTAIM, ELF and Hirshfeld-based approaches). These indices are currently employed as a criterion to study chemical challenges related to concepts like bond orders, pericyclic reactions, 3-center 2-electron bonds (such as the B_2H_6 "banana bonds") and, especially, aromaticity.

1.1 Scheme of the program

NDELOC is a software designed to perform computations of multicenter (or N-center) electron localization and delocalization indices in a parallelized and efficient way from real space partitionings, such as Quantum Theory of Atoms In Molecules (QTAIM),[1] electron localization function (ELF)[2,3] and Hirshfeld-based[4–6] approaches. However, the program could compute any atomic partitioning in which an atomic overlap matrices (AOMs) were computable by means of the combination with Fukui program. Among the different features of the program, we highlight:

- It supports several wave function file formats to carry out the calculations: wfn, fchk and molden.
- It is parallelized with different strategies, choosing the most adequate one according to the kind of calculation.
- Together with a detailed output file, the program can print a .xyz file with the N-delocalization indices[*] compatible with Chemcraft, a graphical program for visualization of quantum chemistry computations (`https://chemcraftprog.com/`).
- It is able to compute a specific set of molecular orbitals (MOs) defined by the user, as well as several predefined presets (π, σ and outer shell MOs).
- Multideterminental AOM are supported via natural orbitals to obtain approximate N-center (de)localization indices.[†]
- Easily expandable to increase the computable order of the N-center (de)localization indices. The program includes up to 14.[‡]

[*]This option is only available when the index order is $N > 2$

[†]Keep in mind the property of idempotence is not accomplished in the case of post-HF calculation.

[‡]Do not forget the computational resources scaling, explained in Section 3.

- It is able to use a basic ring perception algorithm.
- Several AOM formats are supported: Fukui, STOCK, AIMPAC, AIMAll and Multiwfn.

1.2 Program details

- Programming language: FORTRAN 77 and Fortran 90.
- Operating systems: Any with Fortran (compilers), CMake and OpenMP.
- List of source files:
  - In root ("./") directory: ndeloc.f, goon.f
  - In *calc* directory: ndelocma.f, scaleovermat.f90
  - In *combinatorics* directory: allnr.f, ft.f, nbinom.f, nextp.f, permut.f
  - In *common* directory: atomlabel.f, calcdist.f, hmnumb.f, ncolumn.f, normaliz.f90, openfile.f, prepnumb.f, procnumb.f, showprog.f90
  - In *loopdir* directory: choose.f, looporb3.f, looporb4.f, looporb5.f, looporb6.f, looporb7.f, looporb8.f, looporb9.f, looporb10.f, looporb11.f, looporb12.f, looporb13.f, looporb14.f, looptree.f, ndeloc1.f, ndeloc2.f
  - In *modules* directory: mo_kind.f90 mo_quicksort.f90 mo_utils.f90
  - In *openmp* directory: init_par.f, strategy.f
  - In *output* directory: ringhost.f, results.f
  - In *read* directory: atomovma.f, eloc.f, fchkinfo.f, moldeninfo.f90, mwfn.f, p1fromgauss.f90, readfch.f, readmold.f90, readsom.f, readwfn.f, sudgfchk.f, sudgfdim.f, wfn-info.f, atovmall.f
  - In *ring* directory: connect.f, ringdtct.f
  - In *sort* directory: indexxabs.f90
- List of parameter files: cnstants.h, descr.h, elements.h, ndeloc.param.h
- List of CMake files: CMakeLists.txt, calc/CMakeLists.txt, combinatorics/CMake-Lists.txt, common/CMakeLists.txt, loopdir/CMakeLists.txt, modules/quicksort.f90 openmp/CMakeLists.txt, output/CMakeLists.txt, read/CMakeLists.txt, ring/C-MakeLists.txt, sort/CMakeLists.txt

1.3 Structure of the program

The structure and steps of the program are summarized in Figure 1.1.

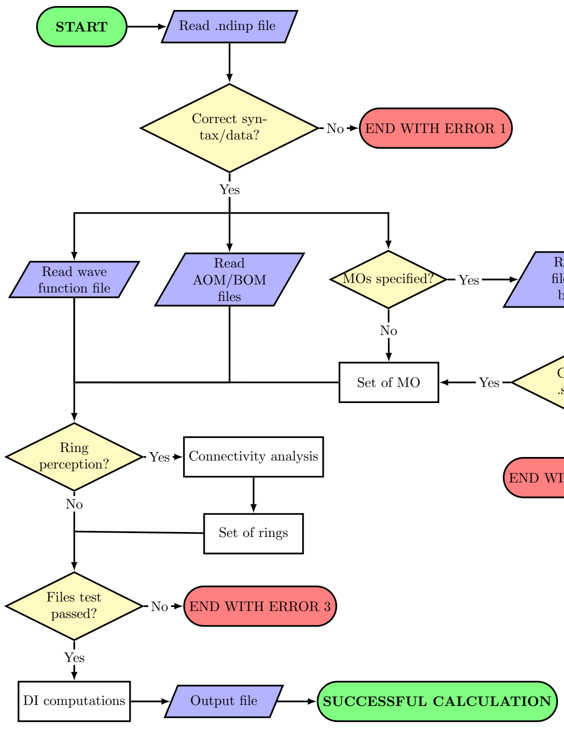


Figure 1.1: Simplified flowchart of the NDELOC 1.2.17 program execution. Terminal blocks are represented by a green stadium shape, while the input/output and decision blocks correspond to blue rhomboid and yellow rhombus shapes, respectively. Last, the rectangle is used to stand for process blocks in the program. The acronyms of delocalization index (DI), atomic overlap matrix (AOM), basin overlap matrix (BOM) and molecular orbital (MO) are also employed in this chart.

1.4 Terms of use, dependencies and license

NDELOC is free software under GPLv3 license. We refer to the GitHub web page for more details about the license:

`https://github.com/nom05/ndeloc`

A small part of the program needs to use a *quicksort* module to efficiently sort distances between pairs of atoms in a parallelized way. For this purpose, we have added the module called *mo_quicksort*, together with its dependencies *mo_utils* and *mo_kinds*, from the “JAMS Fortran library”, distributed under the MIT license by Matthias Cuntz, Juliane Mai and Stephan Thober (`https://github.com/mcuntz/jams_fortran`).

1.5 How to cite NDELOC 1.2.17

- N. Otero and M. Mandado, NDELOC v1.2.17 (2022), Universidade de Vigo.

This program was developed considering the following works:

- Mandado, M. et al. *J. Comput. Chem.* **2007**, *28*, 127–136.
- Mandado, M. et al. *J. Comput. Chem.* **2007**, *28*, 1625–1633.

The first version of the current program was developed for this work:

- Karamanis, P. et al. *J. Am. Chem. Soc.* **2014**, *136*, 7464–7473.

2 Theoretical background and units employed in NDELOC

The electron N-center (or multicenter) delocalization indices, $(\varDelta_N)$, represent a measure of how many electrons can be delocalizated or shared among several atomic regions with respect to a null Fermi hole density, whereas their localization counterparts symbolize how many electrons belong to a single atom. Therefore, the former ones can be also negative in certain conditions. The (de)localization (localization and delocalization) indices implemented in NDELOC follow the QTAIM-based scheme implementation* of prof. Mandado[7,8] via the Generalized Population Analysis (GPA).[10] The Ponec $\varDelta_N$ can be separated into α $(\varDelta_N^\alpha)$ and β $(\varDelta_N^\beta)$ components:

$$\varDelta_N = \varDelta_N^\alpha + \varDelta_N^\beta \tag{2.1}$$

being $\varDelta_N^\alpha$ (and $\varDelta_N^\beta$ by analogy):

$$\varDelta_N^\alpha = N \sum_{j=1}^{(n-1)!} \left[\hat{P}_j \delta_N^\alpha(i_1, i_2, \ldots, i_N)\right] \tag{2.2}$$

where $\hat{P}_j$ is the permutation operator, and $\delta_N^\alpha(i_1, i_2, \ldots, i_N)$ is the "N-center electron delocalization term", defined by Giambiagi et al.[11–13] The latter is assigned to every specific permutation, even though when $n \leq 3$, $\varDelta_N^\alpha$ is equivalent to $\delta_N^\alpha(i_1, i_2, \ldots, i_N)$. In monodeterminantal wave functions, $\delta_N^\alpha(i_1, i_2, \ldots, i_N)$ can be written as summations of occupied spin orbital products (ϕ_{i_n}):

$$\delta_N^\alpha(i_1, i_2, \ldots, i_N) = \sum_{i_1=1}^{n_{\text{occ}}^\alpha} \sum_{i_2=1}^{n_{\text{occ}}^\alpha} \cdots \sum_{i_N=1}^{n_{\text{occ}}^\alpha} \int_{\Omega_1} \phi_{i_1}^\alpha(\vec{r}_1)\phi_{i_2}^\alpha(\vec{r}_1)\,\mathrm{d}\vec{r}_1 \ldots \int_{\Omega_N} \phi_{i_N}^\alpha(\vec{r}_N)\phi_{i_1}^\alpha(\vec{r}_N)\,\mathrm{d}\vec{r}_N \tag{2.3}$$

Note that the summation runs over all α spin orbitals but a selection can be chosen like all π spin orbitals, for instance. Ω_j and $\phi_{i_j}^\alpha$ stand for the real-space atomic partitioning and the different atomic overlap matrices (AOMs) elements. These elements are combined as products, whose values are ranged between 0 and 1 (due to MO normalization). This provokes that the indices decrease in powers of the index order and the values are not comparable with others of different order. With the purpose of mitigating this drawback, several normalizations were proposed taking into account the index order, N:

- Ponec's approach:
$$\varDelta_N^{\text{Ponec norm}} = 2^{N-1} \cdot \varDelta_N$$

*This scheme was implemented in the real space QTAIM framework, but it is general and applicable to other real space partitionings such as ELF and Hirshfeld-based approaches.

- Cioslowski's approach:

$$\Delta_N^{\text{Ciosl. norm}} = \underbrace{\frac{\Delta_N}{|\Delta_N|}}_{\text{Index sign}} \cdot |\Delta_N|^{\frac{1}{N}}$$

Finally, it is also important to highlight that, for the Kohn-Sham (KS) formalism, the monodeterminental wave function is an idempotent approximation from the real wave function, though reasonably good with respect to the correct density functional theory (DFT) indices.[14]

If not stated otherwise, the NDELOC program outputs its results in atomic units (electrons) and angstroms (Å) for populations and distances, respectively.

3 Computational cost

The computational cost of real space DIs depends on the number of occupied MOs defined by the user and scales via the power of the index order. In addition, the program performs two kinds of delocalization indices, using the definitions of Giambiagi and Ponec. The former does not include the permutation operator and the user needs to know the correct connectivity of the ring to specify the right order of atoms (we will see a workaround to avoid user interaction in Section 6.5 on Page 16).

When the permutation operator is considered, the calculation will be obtained through Ponec's formulation, and $(N-1)!$ permutations will be used for each ring calculation in the molecule.

For example, consider the coronene molecule with 78 occupied MOs. The all-MO Giambiagi 6-DI calculation will perform 78^6 products (per ring), that is, the huge amount of 225 199 600 704 products. What's more, this value is multiplied by $(6-1)! = 120$ with Ponec's approach, *i.e.*, 27 023 952 084 480 products are computed. These astronomical values (imagine larger systems) can be substantially reduced with a truncated set of MOs like in π-aromatic systems.

4 Program compilation

NOTE: The preferred platform or, more specifically, the platform the developers use is GNU/Linux. MS Windows is an untested alternative. Through WSL or WSL2 the compilation would be equivalent to the following instructions. It is beyond the scope of this manual to explain how to activate these options in MS Windows.

The program source code will be available in `https://github.com/nom05/ndeloc`. To install the program, follow the instructions:

1. Clone the repository (if not previously downloaded/included in a physical medium):

   ▷ git clone github.com/nom05/ndeloc

   ```
   Cloning into 'ndeloc'...
   remote: Enumerating objects: 155, done.
   remote: Total 155 (delta 0), reused 0 (delta 0), pack-reused 155
   Receiving objects: 100% (155/155), 63.84 KiB | 1.20 MiB/s, done.
   Resolving deltas: 100% (80/80), done.
   ```

2. Enter in the directory:

   ▷ cd ndeloc

3. Verify your current CMake version[*] is equal or greater than 2.8.12.

   ▷ cmake --version

   ```
   cmake version 3.22.3
   CMake suite maintained and supported by Kitware (kitware.com/cmake).
   ```

4. Create a new directory called "build" and enter inside, for example:

   ▷ mkdir build; cd build

[*]From Wikipedia: In software development, CMake is cross-platform free and open-source software for build automation, testing, packaging and installation of software by using a compiler-independent method. CMake is not a build system but rather it generates another system's build files. It supports directory hierarchies and applications that depend on multiple libraries. It is used in conjunction with native build environments such as Make, Qt Creator, Ninja, Android Studio, Apple's Xcode, and Microsoft Visual Studio. It has minimal dependencies, requiring only a C++ compiler on its own build system.

5. Next, create the compilation environment:

▷ cmake ..

```
CMake Deprecation Warning at CMakeLists.txt:4 (cmake_minimum_required):
Compatibility with CMake < 2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min> value or use a ...<max> suffix to tell
CMake that the project does not need compatibility with older versions.


CMAKE_Fortran_COMPILER is gfortran
CMAKE_GENERATOR_FC is gfortran
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- The Fortran compiler identification is GNU 11.2.0
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Check for working Fortran compiler: /usr/bin/gfortran - skipped
CMAKE_BUILD_TYPEisRelease
-- Found OpenMP_C: -fopenmp (found version "4.5")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP_Fortran: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
OPENMP FOUND
-- Configuring done
-- Generating done
-- Build files have been written to: /tmp/ndeloc/ndeloc/build
```

CMake will detect the compiler searching from several targets. Usually, the first one it finds is GNU Fortran (gfortran). To specify another alternative compiler, use the option "-DCMAKE_Fortran_COMPILER".

▷ cmake -DCMAKE_Fortran_COMPILER=ifort ..

Additionally, in the main file called "CMakeLists.txt" in the parent directory of the source code, you can replace "Release" to "Debug", for debug builds, with your favorite editor (nano, vi, vim, Visual Studio Code, etc.). Simply change the word "Release" by "Debug". Moreover, comment (adding the symbol '#') or uncomment (remove '#') lines with the purpose of disabling or enabling your personal options to personalize your compilation.

WARNING: We recommend to use *gfortran* as default compiler according to our tests. Intel Fortran compiler (*ifort*) does not represent any advantage over the former. The complex nested loops to compute N-(de)localization indices suffer a performance penalty with this compiler.

6. And finally compile the code:

▷ **make**

```
Scanning dependencies of target ndeloc.x
[  1%] Building Fortran object CMakeFiles/ndeloc.x.dir/calc/ndelocma.f.o
[  3%] Building Fortran object CMakeFiles/ndeloc.x.dir/calc/scaleovermat.f90.o
[  5%] Building Fortran object CMakeFiles/ndeloc.x.dir/combinatorics/allnr.f.o
/tmp/ndeloc/ndeloc/combinatorics/allnr.f:32:72:

32 |  2     J(L) = J(L - 1) + 1
|                                                                          1
Warning: Fortran 2018 deleted feature: DO termination statement which is not END DO or CONTINUE with
[  7%] Building Fortran object CMakeFiles/ndeloc.x.dir/combinatorics/ft.f.o
[...]
[ 85%] Building Fortran object CMakeFiles/ndeloc.x.dir/read/sudgfchk.f.o
[ 87%] Building Fortran object CMakeFiles/ndeloc.x.dir/read/sudgfdim.f.o
[ 89%] Building Fortran object CMakeFiles/ndeloc.x.dir/read/wfninfo.f.o
[ 91%] Building Fortran object CMakeFiles/ndeloc.x.dir/ring/connect.f.o
[ 92%] Building Fortran object CMakeFiles/ndeloc.x.dir/ring/ringdtct.f.o
[ 94%] Building Fortran object CMakeFiles/ndeloc.x.dir/sort/indexx.f.o
[ 96%] Building Fortran object CMakeFiles/ndeloc.x.dir/goon.f.o
[ 98%] Building Fortran object CMakeFiles/ndeloc.x.dir/ndeloc.f.o
[100%] Linking Fortran executable ndeloc.x
[100%] Built target ndeloc.x
```

The compilation messages on screen between 7 % to 85 % were removed for the sake of simplicity.

TIP: To compile faster use several processor threads (increase the number of threads, #threads) through the option "-j#threads". For example, "make -j3" will set three (3) threads.

NOTE: For very large systems the user could need to edit *ndeloc.param.h* to increase the maximum size of the arrays. The file includes comments to describe the meaning of the parameters.

7. You will find the executable in the current compilation directory:

▷ **ls -tr**

```
CMakeCache.txt  sort  ring  read  output  openmp  loopdir  common  combinatorics
cmake_install.cmake  calc  Makefile  ndeloc.x  CMakeFiles
```

ndeloc.x

5 How to run NDELOC:

NDELOC has a special way to specify the options of the calculation. Without arguments, the executable prints a short help (in this manual, the output is divided into two boxes to improve the page layout space):

▷ ./ndeloc.x (part 1 of 2)

```
                N D E L O C  v.1.2.17
requested file does not exists: .ndinp
Usage: ./ndeloc.x input-file (without extension)
INPUT file HELP/TIPS:
( * = possible options in the corresponding line )
o Line 1 :  WFN/FCHK file with extension+# threads+debug
 *  file   -> File with extension:
  - wfn    reading algorithm.
  - fchk   reading algorithm.
  - molden reading algorithm.
 *  # threads to be used through OpenMP.
 *  debug  -> debug mode is set.
o Line 2 :  OUTPUT          specification:
 *  text   -> file without extension.
 *  xyz    -> print xyz with the N-DELOC indices (if ring is specified) (only for Chemcraft)
o Line 3 :  MO + occ        specification:
 *  all    -> All mol orbitals included.
 *  pi     -> All PI mol orbitals included.
 *  outersh-> Inner s shell not included.
 *  1,3-5  -> Numbering specif. will be considered separated by comma or dash
              (this example: 1,3,4,5).
 *  occ    -> Occupation numbers will be used to scale multidet. overl. matrices.
```

▷ ./ndeloc.x (part 2 of 2)

```
o Line 4 :  N-(DE)LOC index specification and overlap matrix type:
 *  integer-> N-(DE)LOC index order.
 *  deloc  -> Deloc index only (for n>2).
 *  giamb  -> Only Giambiagi index will be considered. ring option in line 5 if available
              is recommended.
 *  aom    -> Atomic overlap matrices (default).
 *  bom    -> Basin  overlap matrices. mwfn is uniquely the compatible format for
              this option.
o Line 5 :  Atom specification:
 *  all    -> All atoms will be included.
 *  heavy  -> All heavy atoms will be included.
 *  1,3-5  -> Numbering specif. will be considered separated by comma or dash
              (this example: 1,3,4,5).
 *  ring   -> Ring detection. It is used together with all prev. options.
o Line 6 :  Atomic Overlap Matrices specification:
 *  read   -> Files will be read below.
 *  fuk    -> fuk format to be used: mol_mol_X01.
 *  int    -> int format to be used: mol_X01.
 *  aimall -> int format to be used: mol_X01. (same file name fmt,diff. data str.
 *  eloc   -> eloc format to be used: mol.
 *  mwfn   -> Multiwfn format to be used. =read is optional and it will read
              the next line to obtain the file name.
 *  Integer-> length for atom label (1=X1,2=X01,3=X001, ...)
o Lines 7-... :  Files (if applicable).
```

As you can observe in the previous execution output without arguments, the program needs an input file containing all the information about the calculation, such as the wave function file, the AOMs and other complementary files depending on the options the user sets. The input file extension has to be ".ndinp" and will be omitted in the argument:

▷ ls *.ndinp

```
coronene.ndinp
```

▷ ./ndeloc.x coronene

In Section 6 the different options of NDELOCwill be described.

6 Input file

As mentioned in Section 5, NDELOC needs an input file with the “.ndinp” extension. The main feature of the input file is that the different options are classified by “thematic” lines and, at least six (6) lines have to be included in any NDELOC input file. In the following subsections, we will explain the available options.

6.1 Line 1: wave function file, number of threads and debug options

Summary of options for Line 1:

```
o Line 1 :  WFN/FCHK file with extension+# threads+debug
 *  file   -> File with extension:
  - wfn    reading algorithm.
  - fchk   reading algorithm.
  - molden reading algorithm.
 *  # threads to be used through OpenMP.
 *  debug  -> debug mode is set.
```

In the first line of the input file, the user specifies the wave function file, the number of processors and, if necessary, the “debug” flag, respecting this command order and each option separated by a blank space with respect to the previous one. PROAIMS, Gaussian formatted checkpoint and molden files are supported and the extension is mandatory for each file. The number of threads must be of integer kind. *debug* will print very detailed information of the steps the program is carrying out.

Example 1: Calculation using a coronene PROAIMS wave function file and 8 threads

coronene.wfn 8

6.2 Line 2: output file name and enabling .xyz output files

Summary of options for Line 2

```
o Line 2 :  OUTPUT          specification:
 *  text   -> file without extension.
 *  xyz    -> print xyz with the N-DELOC indices (if ring is specified)
                    (only for Chemcraft)
```

In the second line of the input file, the user controls the output specification. The main file is where the program prints all the calculation details, the rings detected (if requested) and the results. The extension of the output file is added automatically and the program will never overwrite an existent file, since it will rename the old one (adding an extra extension by means of the command “date +%g%+m%d%H%M%S”) and will stop with a warning message. In Section 8 you will find more information about the structure of this output file.

The .xyz output files are generated whether the “ring” option is set in Line 5 (see Section 6.5 for more details) and the order of the (de)localization indices is greater than 2 in Line 4 (see Section 6.4). The file names will be generated according to the main output file name and adding the “-ndel$N$-*approach*.xyz” suffix, where N is the index order and *approach* only has two possible texts, “gia” and “pon”, for Giambiagi and Ponec approaches, respectively. Therefore, the program will generate two .xyz files by default, but one specifically if Giambiagi’s definition is enabled in Line 4 (see Section 6.4).

The .xyz files contain the element symbols and Cartesian coordinates in Å of each atom, and one line per detected ring. In this (these) line(s), NDELOC includes the symbol of a ghost atom (X), the Cartesian coordinates of the ring geometrical center and an extra column including the corresponding delocalization value. This .xyz file can be opened with Chemcraft (`https://chemcraftprog.com/`) to visualize the delocalization indices. Note that the values are in atomic units and multiplied by 10^{-4} and 10^{-3} for the Giambiagi and Ponec approaches, respectively.

Example 2: Choosing the output file name and activating the generation of xyz output files

coronene-6del xyz

6.3 Line 3: MOs set and post-HF AOMs scaling

Summary of options for Line 3

```
o Line 3 :  MO + occ        specification:
 *  all     -> All mol orbitals included.
 *  pi      -> All PI     mol orbitals included.
 *  sigma   -> All SIGMA mol orbitals included.
 *  outersh -> Inner s shell not included.
 *  1,3-5   -> Numbering specif. will be considered separated by comma or dash
               (this example: 1,3,4,5).
 *  occ     -> Occ. numbers will be used to scale multidet. overl. matrices.
```

In the third line of the input file, the user specifies the set of MOs to be employed in the calculation (this is mandatory) and if the AOMs come from a multideterminental calculation and will be scaled via the occupation numbers (optional). Four kinds of labels can be employed to define the MOs:

all	The full set of MOs will be considered.
pi	A set of MOs is read from an existent .som or .sg file.
outersh	The number of MOs to be considered are decreased removing the 1s MOs of the heavy atoms.
range	The numbering of the MOs set is defined by means of a range spefied by hand. For instance, a range of “1,3–5” corresponds to request the MOs 1, 3, 4 and 5.

With respect to the scaling of multideterminental AOMs, the label *occ* activates the option. However, keep in mind Gaussian format checkpoint files do not include explicitly the occupation numbers[*] and only .wfn and .molden wave function formats can be used.

Example 3: Defining a MOs set and activating the post-HF AOMs scaling

7,10-30 occ

6.4 Line 4: index order, kind of calculation, AOMs-type specification

Summary of options for Line 4

```
o Line 4 :  N-(DE)LOC index specification and overlap matrix type:
 *  integer-> N-(DE)LOC index order.
 *  deloc   -> Deloc index only (for n>2).
 *  giamb   -> Only Giambiagi index will be considered.
               ring option in line 5 if available is recommended.
 *  aom     -> Atomic overlap matrices (default).
 *  bom     -> Basin  overlap matrices. mwfn is uniquely the compatible format
               for this option.
```

In the fourth line of the input file, the user will set the index order (mandatory and expressed as an integer, $2 \leq N \leq 14$ is currently implemented)[†]. The rest of the labels can be set optionally:

deloc It requests a calculation with delocalization indices only (by default the localization indices are also computed), saving time if not needed (it uses the resources equivalent to a Giambiagi's calculation).

giamb By default NDELOC performs Ponec indices that always include the Giambiagi permutation. The addition of this label reduces dramatically the computational resources ($(N-1)!-1$ times less) skipping all the Ponec permutations except one. Keep in mind Giambiagi's approach is not always the best option to represent the molecular aromaticity. In the case of setting this option, consider to use *ring* in Line 5 (Section 6.5) to take into account an automatic ring connectivity.

AOM Rigorously, we cannot think about atomic overlap matrices (AOMs) when the molecular space partition scheme is not atomic-centered defined, as is the case of the ELF approach, based on basins. NDELOC, in addition, checks by default the number of both AOMs and atoms is at most equal, stopping the calculation otherwise. If the user considers to use BOMs the label *bom* will be needed in this line. Else, the label *aom* is automatically and internally used by default. Multiwfn BOM files are only currently supported (see Line 6 in Section 6.6).

[*]The occupation numbers could be calculated from the density matrix and the MO coefficients, but this is beyond the scope of the program. On the other hand, they could be included from a Gaussian calculation in the .fchk file, replacing the MO energies, by means of IOPs options (see Gaussian IOPs reference). Anyway, this last possibility is not implemented in the source code of NDELOC at the present time.

[†]Consider always the computational requirements to calculate indices with an order greater than 6 ($N > 6$), as is explained in Section 3.

Example 4: Requesting a Ponec's 6-order calculation only calculating delocalization indices with AOMs

```
6 deloc
```

6.5 Line 5: atom specification

Summary of options for Line 5

```
o Line 5 :  Atom specification:
 *  all    -> All atoms will be included.
 *  heavy  -> All heavy atoms will be included.
 *  1,3-5  -> Numbering specif. will be considered separated by comma or
              dash (this example: 1,3,4,5).
 *  ring   -> Ring detection. It is used together with all prev. options.
 *  value  -> Maximum distance between two atoms (optional)
```

In the fifth line of the input file, the atom/basin specification is set. By default, if the ring perception is not set the number of combinations (c) comes from the expression:

$$c = \binom{n}{N} = \frac{n!}{N!(n-N)!} \tag{6.1}$$

where N and n represent the index order and the selected number of atoms, respectively. We reemphasize the importance of considering beforehand the computational resources of the calculations. The number of products will be multiplied by c (see Section 3 to estimate the number of products).

One of the following settings must be specified:

all All combinations will be considered according to the index order specified in Line 4 (see Section 6.4).

heavy Only heavy atoms will be considered.

range The user defines a specific set of atoms. For instance, a range of "1,3-5" corresponds to use the atoms 1, 3, 4 and 5.

Together with one of the aforementioned labels, the NDELOC program has implemented a basic ring perception algorithm, activated by the label *ring* with index orders greater than two (2). It is based on the idea of creating a connectivity matrix with the set of the requested atoms and the program iterates all the combinations to find closed cycles (rings). Do not use it with huge sets because the demands of computational resources scales very fast. By default, the program uses a maximum bond length cut-off considered for C−C and B−N bonds (1.6 Å). To increase this distance, a real value can be read after the atom specification and the *ring* label. If *xyz* label is set in Line 2 (see Section 6.2 for more information) the program creates .xyz files including the DIs at the geometrical center of the rings.

Example 5: Considering all atoms for the ring perception algorithm with a maximum bond length of 1.8 Å

all ring 1.8

The previous example could be valid for P−C bonds, for instance.

6.6 Line 6 and so on: AOMs/BOMs specification

Summary of options for Line 6

```
o Line 6       : Atomic Overlap Matrices specification:
 *  read   -> Files will be read below.
 *  fuk    -> fuk format to be used: mol_mol_X01.
 *  int    -> int format to be used: mol_X01.
 *  aimall -> int format to be used: mol_X01.
              (same file name fmt,diff. data str.)
 *  eloc   -> eloc format to be used: mol.
 *  mwfn   -> Multiwfn format to be used. '=read' is optional and it will read
              the next line to obtain the file name.
 *  Integer-> length for atom label (1=X1,2=X01,3=X001, ...)
o Lines 7-... : Files (if applicable).
```

In the sixth and the subsequent lines, the user will specify the AOM or basin overlap matrix (BOM) format to be read by NDELOC. Remember the latter is only compatible with the output from Multiwfn, as explained in Section 6.4. The program will generate the overlap matrices file names using different schemes depending on the kind of file specified. Let the reference wave function file name, the atom symbol and number be "mol", "X" and 1, respectively. One of the following settings must be specified and the program will search for different file names:

read The program will use the following lines of the NDELOC input file to read the AOM file names in AIMPAC/PROAIMS format (see below). The file names can be arbitrary and must include the extension.

int The program will use Bader's AIMPAC/PROAIMS format to read the files. The file names are generated and will use the following scheme: "mol_X01.int".

fuk The FUKUI output files are compatible with the AIMPAC/PROAIMS format but the file names are generated as: "mol_mol_X01.fuk" names.

aimall The AOMs will be obtained from the AIMAll .int files, generated as "x1.int".

eloc STOCK format will be employed. The file name will be generated as "mol.eloc".

mwfn Multiwfn format will be used (`http://sobereva.com/multiwfn/`). By default, the program will consider either aom.txt or bom.txt depending on the specification in Line 4 (see Section 6.4 for more details). When the label is *mwfn=read*, the file name will be read in the next line (useful if the user has renamed the "aom.txt" or "bom.txt" file).

There are situations in which the total number of atoms is greater than 99 and, consequently, the atomic labels will not correspond to "X01" but "X001". Optionally, the user can control the number of digits including an integer (3 by default, "X001") to generate the file names correctly.

Example 6: Defining a calculation with FUKUI AOMs using "X001" as atomic symbol and numbering scheme

```
fuk 3
```

7 Output on screen

In this section we will discuss about the NDELOC output on screen. A typical calculation, coronene[*], will be used as reference:

▷ **cat coronene-6del.ndinp**

```
coronene.fchk 4
coronene-6delpi xyz
pi
6 deloc
heavy ring
fuk
```

In this example a Gaussian formatted checkpoint file is used as reference wave function. The calculation uses four (4) threads, the output file name is "coronene-6delpi.ndout", and the program will generate special Chemcraft xyz files as requested. The π-symmetric MOs set is specified, using an external file called "coronene.som" or "coronene.sg". We consider the computation of 6-DI uniquely (localization values are not included, therefore), and the set of atoms only includes the heavy atoms, *i.e.*, the carbon atoms, but the program will particularly employ the combinations forming rings.

Firstly, the program will print the computational details, defined in the input file, as well as some values extracted from the .fchk, .som (or .sg) and .fuk files. As mentioned above, four threads were specified:

Parallelization details

```
N D E L O C  v.1.2.17

>> Parallelization enabled <<

**************** THREAD INFORMATION ******************
>> Job running using OpenMP.
>> The number of processors is .....   4
>> The number of threads is ........   4
Hello from process       0
Hello from process       3
Hello from process       2
Hello from process       1
>> Elapsed wall clock time ........          0

******************************************************

>> # threads ............ 4
```

The wave function and output files are the next information the program prints:

Wave function and output files information

```
>> FCHK File ............ coronene.fchk
>> OUTPUT File .......... coronene-6delpi.ndout
>> XYZ file option detected <<
```

[*]Coronene is a polycyclic aromatic hydrocarbon (PAH) formed by a central benzene ring fused to other six benzene rings. Its chemical formula is $C_{24}H_{12}$.

Now it is the turn of the MOs. The program has read the .som or .sg file and shows how many orbitals will be used in this calculation:

Number of π MOs

```
>> Specified orbitals ... PI (12 MOs)
```

The next step is to print the index order and more information obtained from the wave function file (.fchk file in this case):

Index order and info from wave function file

```
>> 6-(DE)LOC will be calculated <<
>> NMOs,NBFunc,NAtoms ... 78, 420, 36
>> NHeavy Atoms ......... 24
```

The following details correspond to the number of atoms considered in the calculation and whether the ring detection is enabled:

Number of considered atoms and activation of ring detection

```
>> # Specified atoms .... 24
>> Ring detection will be used <<
>> # (de)loc indices will be calculated with ring detection <<
```

The atomic overlap matrices (AOMs) will be read using the Fukui format. The program will check the number of digits in the atomic label (X001, by default) and will calculate the populations obtained through the MOs set, which we have previously specified.

Number of considered atoms and activation of ring detection

```
>> AOM File type ........ fuk
** Checking number of zeros in the file names
>> POPULATIONS .......... 0.96 0.96 1.00 1.00 0.96 0.96 1.00 1.00 1.00 1.00 0.96
                          0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96
                          0.96 0.96
```

As can be observed in the previous box, all values are in agreement with the typical π atomic populations of carbon atoms.

According to the computational details, NDELOC chooses a parallelization strategy and prints, in addition, information about how the ring perception algorithm is working:

Parallelization strategy and ring perception algorithm details

```
>> Parallelization strategy ... 2 2
>> qcksort spent ........ 0.00 s
>> Bond dist cut-off .... 1.600 A
>> # bonds .............. 30
>> Frontier distances ... d(8,9)=1.349 d(4,8)=1.349 d(7,10)=1.349
                          d(1,5)=2.302 d(23,25)=2.302
                          (3 last atom pairs used in ring detection
                           process together with 2 first unused ones)
>> Connec part spent .... 0.01 s
>> # Detected rings ..... 7
```

After obtaining a distance matrix, the program employs a sorting algorithm (*quick-sort*) to separate bond lengths from the rest of values. The maximum value considering an atom is bonded to another one is 1.600 Å by default, but this can be modified in Line 5 of the input file (Section 6.5). The frontier distances are crucial to know whether the program is detecting the bonded atoms adequately. The program will print the three last pairs of bonded atoms and the next distances the program is discarding. In case of error, the user would observe this list (3 bonded + 2 nonbonded ones) is not accomplished. Finally, a message with the elapsed time of the ring perception algorithm and the number of rings detected by NDELOC is printed. As can be expected, the program detects the seven (7) rings of coronene.

The program prints the number of permutations needed, in case the Ponec approach is used, together with a progression of the calculation. When it is finished, the program writes the output (NDELOC does not create the output file until the end) and the total elapsed time.

Number of Ponec's permutations, progress, message while writing the output file and total elapsed time

```
>> # Permutations ....... 120
** Percent complete for delocalization indices ...100.00%

>> Writing OUTPUT ....... Done
>> Elapsed time ......... 3.04 seconds
```

8 Output files

In this section we will describe the output file with the results of coronene. While the output on screen is equivalent in any index order, two formats of output files are printed depending on the order of the (de)localization indices. Thereby, we distinguish between the output file with $N = 2$ and $N > 2$. Firstly, we will discuss the general case, $N > 2$, in Section 8.1, and the differences when $N = 2$ (Section 8.2).

8.1 Output file when the index order corresponds to $N > 2$

In this case, we use the following input file as reference, whose atomic numbering is shown in Figure 8.1:

▷ **cat coronene-6del.ndinp**

```
coronene.fchk 4
coronene-6delpi xyz
pi
6 deloc
1-10 ring
fuk
```

By default, the unique file generated by NDELOC uses the ".ndout" extension and the name specified in Line 2, as is explained in Section 6.2. However, since the label *xyz*

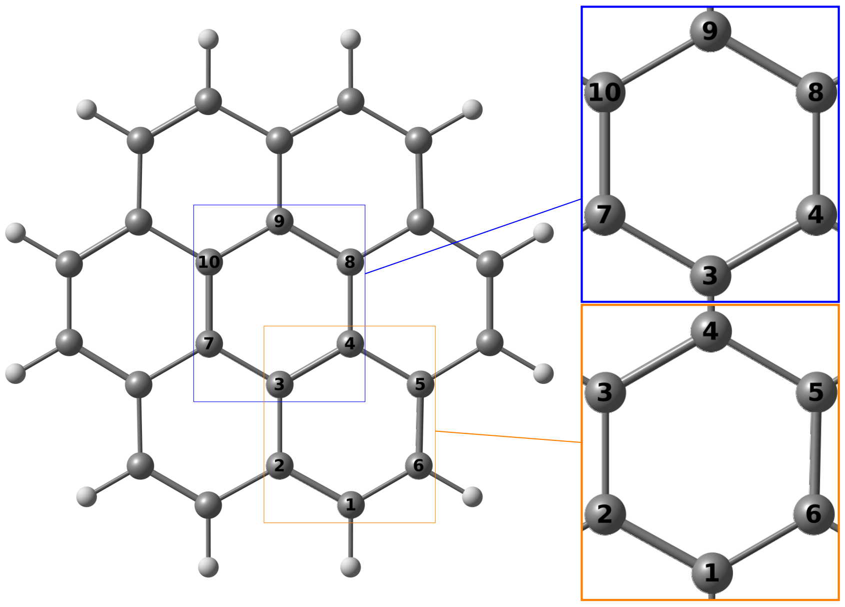
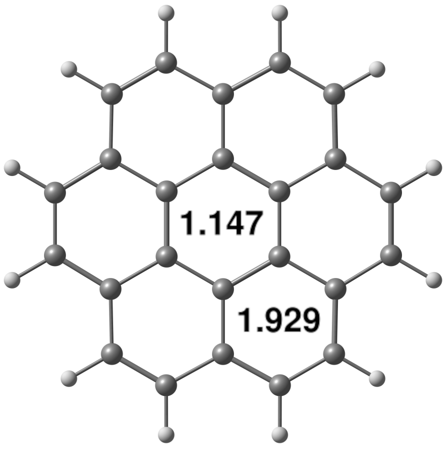
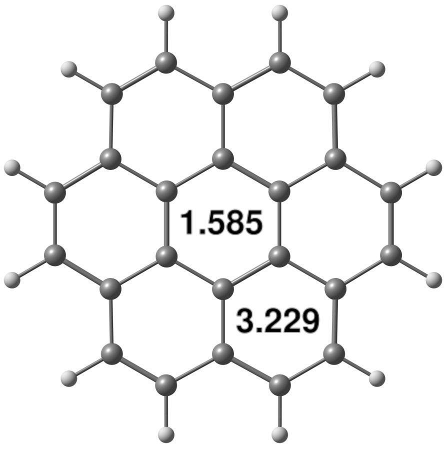


Figure 8.1: Numbering of the symmetrically-independent coronene rings



(a) Giambiagi 6-delocalization index results in au and multiplied by 10^{-4}.



(b) Ponec 6-delocalization index results in au and multiplied by 10^{-3}.

Figure 8.2: fractional occupation Hirsfeld-I (FOHI) Giambiagi (left) and Ponec (right) 6-delocalization indices for the symmetrically-independent rings of the coronene molecule at B3LYP/6-31G(d,p).

is added after the output file name, the program creates two .xyz special files compatible with Chemcraft (`https://chemcraftprog.com/`). We have included an example of a plot generated by the aforementioned program in Figure 8.2.

The first part of the output file consists of a summary of the computational details. Concretely, the input file name the calculation comes from, the wave function file and some molecular parameters such as the wave function type (restricted or unrestricted), the number of MOs, the number of basis functions and the total number of (heavy) atoms are included here:

Header, input and wave functions files together with some molecular parameters

```
+++++++++++++++++++++++++++++++++++++
        N D E L O C  v.1.2.17
+++++++++++++++++++++++++++++++++++++
+ 6-(DE)LOC calculation :  coronene +
+++++++++++++++++++++++++++++++++++++
>> SUMMARY :
* INPUT file ........... coronene-6delpi.ndinp
* FCHK   file ........... coronene.fchk
* NMO,NBFUNC,NAtoms ..... 78, 420, 36 (Restricted)
* # Heavy atoms ......... 24
```

In the case of PROAIMS wave function (.wfn) and MOLDEN (.molden) files, NDELOC will print, in addition, the number of primitive functions and the type of the wave function (HF/KS or post-HF/2hybrid KS).

Next, the MOs selected by the user in a compact way (see Section 6.3 for more details), the index order of the calculation and whether the localization indices were computed (option available if the index order is greater than 2) (see Section 6.4 for more details):

Set of MOs

```
* Specified orbitals .... 55,61-62,68-69,72-78 PI(12)
* (DE)LOC index order ... 6
* LOC indices included .. NO (Available if N>2)
* # (DE)LOC indices ..... 2 (LOC+DELOC)
```

Moreover, the program prints the number of calculations computed.

In the case of post-HF wave functions, and if there are occupation numbers tending to be negligible, the program will truncate the set of MOs, printing a message in the output file. If the scaling of post-HF AOMs or BOMs is activated, a message will be included as well.

The type of overlap matrices is the next information printed in the output. If the calculation employed atomic overlap matrices (AOMs), then the corresponding notification is added in the output together with atoms considered in the calculation in a compact way. Otherwise, a message about the use of basin overlap matrices (BOMs) is included.

Type of overlap matrix and the participating atoms/basins

```
* Atomic overlap matrices (AOM) employed *
* Considered atoms ...... 1-10 (10)
```

When the ring perception is set (see Section 6.5 for more details) NDELOC includes the number of rings detected by the algorithm and the bond length cut-off, *i.e.*, the limit to distinguish between a pair of bonded atoms or another type of distance. In the case of including the *xyz* option in Line 2 (see Section 6.2 for more details) and the index order is greater than 2, the program creates one (if *giamb* label is added in Line 4, see Section 6.4) or two .xyz files and prints a notification in the main output file.

Ring perception and xyz files

```
* # detected rings ...... 2
* Bond dist cut-off ..... 1.600
* XYZ Files with DELOC .. YES
```

The next step of the output file will show the AOM or BOM files used by the program:

AOM or BOM files

```
>> FILES   :
* AOM File   1 .......... coronene_coronene_C001.fuk
* AOM File   2 .......... coronene_coronene_C002.fuk
[...]
* AOM File  10 .......... coronene_coronene_C010.fuk
```

If the *xyz* option is correctly set, the program will print the names of the .xyz files and the multiplication factor of the indices. Keep in mind that if the calculation only includes the Giambiagi approach, only the Giambiagi approach file is generated.

.xyz output file names

```
Multiplication factors (gia,pon): 10000,1000
* XYZ File ............. coronene-6delpi-ndel6-gia.xyz
* XYZ File ............. coronene-6delpi-ndel6-pon.xyz
```

The last section of the output file will evidently include the results together with the numbering of the atoms/basins considered. Two kinds of normalizations are also printed to make comparable the results with other values from different index order.

Results

```
>> RESULTS :
coronene 6-DELOC :                                        Giambiagi      Ponec
* Index    1 ............. C001 C002 C003 C004 C005 C006  1.929E-04  3.229E-03
-         Ponec's norm.                                   6.174E-03  1.033E-01
- Cioslowski's norm.                                      2.404E-01  3.844E-01
* Index    2 ............. C003 C004 C008 C009 C010 C007  1.147E-04  1.585E-03
-       Ponec's norm.                                     3.672E-03  5.072E-02
- Cioslowski's norm.                                      2.204E-01  3.415E-01
```

Finally, the overall elapsed and the most computationally demanding partial times are included:

Elapsed time

```
>> qcksort spent .......... 0.00 seconds
>> Connect analysis spent .. 0.00 seconds
>> Elapsed time ........... 0.25 seconds
```

8.2 Output file when the index order corresponds to $N = 2$

In this section we will discuss the differences between the general output file, when the index order is greater than two (Section 8.1), and the 2-order calculation. Consider the following input file:

▷ cat coronene-2de-loc.ndinp

```
coronene.fchk 4
coronene-2de-loc
pi
2
all
fuk
```

The main changed part corresponds to the `RESULTS` (the `SUMMARY` structure of the computational details is kept unaltered). There, NDELOC prints both localization and delocalization indices as diagonal and off-diagonal elements of a triangular matrix, respectively (useful to be opened with a spreadsheet software, such as M$ Excel or Libreoffice Calc).

Fragment of the triangular matrix of 2-(de)localization indices for coronene

```
>> RESULTS :
                      C001        C002        C003        [...]
coronene  C001      0.37077
coronene  C002      0.28676     0.32624
coronene  C003      0.05361     0.29211     0.34398
[...]
```

The localization indices correspond to the electron population not shared by the atoms while the 2-delocalization values represent the quantity of electrons partaken by two atoms. Normally, if the atoms are directly neighbors, the index is interpreted as the *bond order*. Otherwise, they can be used to analyze mesomeric/resonance effects.

Inasmuch as the triangular matrix can be "uncomfortable" to find a specific value, NDELOC prints the localization indices for each atom the user has requested in a separated block:

Block of 2-localization indices

```
LOC :
C001     0.37077
C002     0.32624
C003     0.34398
C004     0.34400
C005     0.32622
[...]
```

The next block is evidently formed by all the 2-delocalization indices*:

Block of 2-delocalization indices

```
DELOC :
1      2     0.28676     << BOND ORDER
1      3     0.05361
1      4     0.04550
1      5     0.06649
1      6     0.49491     << BOND ORDER
1      7     0.00485
1      8     0.00635
```

Finally, the elapsed time is also included:

Elapsed time

```
>> Elapsed time ........... 0.07 seconds
```

*We have added the text "BOND ORDER" to remark these values on purpose but NDELOC does not have this feature currently.

Bibliography

(1) Bader, R. F. W., *Atoms in Molecules: A Quantum Theory*; International Ser. of Monogr. on Chem; Clarendon Press: 1994.

(2) Becke, A. D.; Edgecombe, K. E. *J. Chem. Phys.* **1990**, *92*, 5397–5403.

(3) Savin, A.; Jepsen, O.; Flad, J.; Andersen, O. K.; Preuss, H.; von Schnering, H. G. *Angew. Chem. Int. Ed.* **1992**, *31*, 187–188.

(4) Hirshfeld, F. L. *Theor. Chim. Acta* **1977**, *44*, 129–138.

(5) Bultinck, P.; Van Alsenoy, C.; Ayers, P. W.; Carbó-Dorca, R. *J. Chem. Phys.* **2007**, *126*, 144111.

(6) Geldof, D.; Krishtal, A.; Blockhuys, F.; Van Alsenoy, C. *J. Chem. Theory Comput.* **2011**, *7*, 1328–1335.

(7) Mandado, M.; González-Moa, M. J.; Mosquera, R. A. *J. Comput. Chem.* **2007**, *28*, 127–136.

(8) Mandado, M.; González-Moa, M. J.; Mosquera, R. A. *J. Comput. Chem.* **2007**, *28*, 1625–1633.

(9) Karamanis, P.; Otero, N.; Pouchan, C. *J. Am. Chem. Soc.* **2014**, *136*, 7464–7473.

(10) Bultinck, P.; Ponec, R.; Damme, S. V. *J. Phys. Org. Chem.* **2005**, *18*, 706–718.

(11) Giambiagi, M.; de Giambiagi, M. S.; Mundim, K. C. *Struct. Chem.* **1990**, *1*, 423–427.

(12) Bollini, C. G.; Giambiagi, M.; de Giambiagi, M. S.; Paiva de Figueiredo, A. *J. Math. Chem.* **2000**, *28*, 71–81.

(13) Bollini, C. G.; Giambiagi, M.; de Giambiagi, M. S.; de Figueiredo, A. P. *Struct. Chem.* **2001**, *12*, 113–120.

(14) Poater, J.; Solà, M.; Duran, M.; Fradera, X. *Theor. Chem. Acc.* **2002**, *107*, 362–371.

Author's signature:

Nicolás Otero Martínez Marcos Mandado Alonso