

# Niedoida User's Guide

September 20, 2007



# Contents

<b>I</b>	<b>Getting started</b>	<b>1</b>
1	Introduction	3
2	Installation	5
2.1	Prerequisites	5
2.2	Compilation	5
2.3	Installation	6
3	The Fine Print	7
<b>II</b>	<b>Niedoida</b>	<b>9</b>
3.1	Capabilities	11
4	Invoking niedoida	13
4.1	Command-line interface	13
4.2	Configuration file	13
4.2.1	Configuration settings	13
4.3	Input description	14
4.3.1	Comments	14
4.3.2	Parameter definitions	14
4.4	Parameters	16
4.4.1	Top level	16
4.4.2	atoms	19
4.4.3	scf	19
4.4.4	units	22
4.4.5	limits	22
4.4.6	integration	22
4.4.7	mp2	23
4.4.8	cis	24
4.4.9	properties	25
4.4.10	output	27
4.5	Examples	27
<b>III</b>	<b>Elektrycerz</b>	<b>31</b>
5	Introduction	33

<b>6 Invoking elektrycerz</b>	<b>35</b>
6.1 Command-line interface . . . . .	35
6.2 Input description . . . . .	35
6.2.1 Comments . . . . .	35
6.2.2 Parameter definitions . . . . .	35
6.3 Parameters . . . . .	36
<b>7 Examples</b>	<b>39</b>
<b>A License</b>	<b>41</b>
<b>B HDF5 License</b>	<b>43</b>
<b>C PETSc License</b>	<b>45</b>
<b>D TAO License</b>	<b>47</b>
<b>E Boost License</b>	<b>49</b>
<b>F Pearls of Wisdom</b>	<b>51</b>

# List of Tables



# List of Figures



# List of examples

4.1	Comments . . . . .	14
4.2	Simple parameters . . . . .	15
4.3	A compound parameter . . . . .	16
4.4	A list parameter . . . . .	16
4.5	Molecule specification . . . . .	19
4.6	Water molecule . . . . .	28
4.7	Ammonia molecule . . . . .	29
6.1	Comments . . . . .	36
7.1	Four fullerene molecules, each represented as a single submolecule	40



Part I

Getting started



# Chapter 1

## Introduction

Niedoida[1] is a general-purpose quantum-chemical and microelectrostatic package, at the moment in the pupal state. For up-to-date information about **niedoida** check <http://www.chemia.uj.edu.pl/~niedoida/en/>.

The **niedoida** package consists of two programs, **niedoida** and **elektrycerz**. They implement the quantum-chemical and the microelectrostatic calculations, respectively.



# Chapter 2

## Installation

This chapter covers all the information necessary to compile and install **niedoida**.

### 2.1 Prerequisites

Before you start compiling **niedoida**, you need a few additional software packages necessary to compile it. The full list of dependencies is

- Tools
  - gcc version 3.4.2 (<http://gcc.gnu.org/>)
  - python version 2.4 (<http://www.python.org/>)
  - scons version 0.96 (<http://www.scons.org/>)
  - T<sub>E</sub>X system, eg. teTeX (<http://www.tug.org/teTeX/>)
  - rubber version 0.99.8 (<http://rubber.sourceforge.net/>)
- Libraries
  - boost version 1.32.0 (<http://www.boost.org/>)
  - HDF5 version 1.6.3.patch (<http://hdf.ncsa.uiuc.edu/HDF5/>)
  - LAM/MPI version 7.1.1 (<http://www.lam-mpi.org/>) or MPICH2 version 1.0 (<http://www-unix.mcs.anl.gov/mpi/mpich2/>)
  - TAO (Toolkit for Advanced Optimization)[2] version 1.9
  - PETSc[3, 4, 5] version 2.3.3

### 2.2 Compilation

**niedoida** build system is based on the scons software construction tool. More information about scons can be found at <http://www.scons.org/>.

To compile **niedoida**, run **scons** in the main directory of the program. If needed, **scons** may be parametrized with compilation options.

It is possible to set several compilation options. They may specify compilation mode and list of directories being searched by the compiler to fulfill external

dependencies. Full list of options, and their default values can be obtained by running `scons -h` in the main directory of the program.

Running `scons` generates all libraries and programs comprising **niedoida**, and full documentation in the PDF format.

Basic tests of most of the components can be performed by running `scons check`.

## 2.3 Installation

The recommended way of installing **niedoida** involves building binary packages and then installing them in the target system using standard package management tools.

**niedoida** build system allows for creation of binary RPM and DEB packages. The DEB packages are built by running `scons deb` in the main directory of the program. The resulting files can be found in the `packages/deb` subdirectory. The RPM packages are built by running `scons rpm`. The resulting files can be found in the `packages/rpm` subdirectory.

The **niedoida** packages depend on the following external third-party libraries:

- boost version 1.32.0
- LAM/MPI version 7.1.1
- HDF5 version 1.6.3.patch

## Chapter 3

# The Fine Print

Niedoida and associated documentation is distributed according to the license allowing you to use it only for conducting scientific research, and without any warranty. For details see [Appendix A](#).

For each publication using results obtained by running **niedoida**, we request citation including the name of the program, it's version and all authors. For the current version the required citation is:

```
@Misc{niedoida02,
  author =      {Grzegorz Mazur and Marcin Makowski and Witold
                  Piskorz and {\L}ukasz {\C}wiklik and Mariusz
                  Sterzel and Mariusz Rado{\n} and Barbara
                  Jagoda-{\C}wiklik, Waldemar Kulig and Daniel
                  B{\l}a{\.z}ewicz},
  title =      {Niedoida 0.2},
  year =      2007
}
```

Additionally, binary distribution of **niedoida** is using, among others, HDF5, TAO, PETSc and Boost libraries. Their licenses are reproduced in [Appendices B, C, D and E](#), respectively.



**Part II**  
**Niedoida**



## 3.1 Capabilities

**niedoida** allows for SCF calculations at the Hartree-Fock level of theory. The implemented schemes are

- RHF,
- UHF,
- ROHF (in Longuet-Higgins-Pople approximation[6, 7]).

For all these methods, the following properties are available:

- energy,
- molecular orbitals,
- Mulliken[8, 9, 10, 11] , Löwdin , Hirshfeld , Voronoi and Bader population analyses,
- Mayer[12, 13], Gopinathan-Jug and Nalewajski-Mrozek[14, 15, 16] bond order analyses.

Moreover, **niedoida** implements selected post-HF calculations

- CIS,
- MP2.



# Chapter 4

## Invoking `niedoida`

The chapter covers all the information necessary to prepare input data and run `niedoida`. It is assumed that the program is already installed and configured.

### 4.1 Command-line interface

`niedoida` provides typical command-line interface. It allows for running the program with input being read either from file, or from the standard input.

Running `niedoida` without any arguments causes the program to read input from the standard input, and write results to the standard output. If an argument is provided, it is treated as the input file name. In this case results are written to a file. The results file is created in the same directory and has the same name as the input file, but the extension is changed to `.log`.

### 4.2 Configuration file

Default configuration is stored in `niedoida` during compilation. When running `niedoida`, it reads the global configuration file (if it exists) and then local configuration file (if it exists). Setting a configuration parameter in any of them overrides the previous value.

Configuration files format is free-form. Spaces, tabs and line breaks are uniformly treated as whitespaces, except for literal strings and comments. Configuration file may contain parameter definitions and comments.

Global configuration file is named `niedoida.cfg`, and is stored in directory `<prefix>/share/niedoida`. Default path to the global configuration file is `/usr/local/share/niedoida`.

Local configuration file is named `niedoida.cfg`, and is searched for in the same directory in which the input file is located. If input is read from standard input, local configuration file is searched for in the current working directory.

#### 4.2.1 Configuration settings

**basis\_set** Parameters set defining where and in which format atomic basis sets descriptions are stored. Each entry has the following form

---

```
"<basis_set_name>" "format" "path"
```

---

and the whole set

---

```
basis_set = {
  <entry_1>,
  <entry_2>,
  ...
  <entry_n>
};
```

---

**scratch\_dir** Parameter defining in which directory temporary files are created. Default is /tmp.

### 4.3 Input description

The input format is free-form. Spaces, tabs and line breaks are uniformly treated as whitespaces, except for literal strings and comments. Input may contain parameter definitions and comments.

#### 4.3.1 Comments

Comments may appear anywhere in the input. They either start with the // string, and end at the end of the line, or start with the /\* string and end with the \*/ string. An example comment is shown in Example 6.1.

Comments do not influence the calculations in any way.

---

```
// A single line comment

/* Another single line comment */

/* A multi-line
   comment
*/
```

---

Example 4.1: Comments

#### 4.3.2 Parameter definitions

Parameters control calculations performed by **niedoida**. By setting them to specific values you decide what and how is calculated when the program is run.

Parameters may be of simple, compound or list type. All parameter definitions have the form

```
<parameter_name>=<parameter_value>;
```

The order of parameter definitions is not significant.<sup>1</sup>

### Simple parameters

Simple parameter values are integer or real numbers, identifiers, or literal strings. Examples are shown in Example 4.2.

---

```
// a string parameter
title = "an example job";

// an identifier parameter
run_type = single_point;

// an integer parameter
charge = -1;

// a real parameter
energy_threshold = 1e-6;
```

---

Example 4.2: Simple parameters

### Compound parameters

Compound parameters are collections of other parameters. Their definitions have the form

```
<compound_parameter_name> = {
  <parameter_name_1> = <parameter_value>;
  <parameter_name_2> = <parameter_value>;
  // ...
  <parameter_name_n> = <parameter_value>;
};
```

where the embedded parameters may be of any type.

An example is shown in Example 4.3.

### List parameters

List parameters are collections of values. Their definitions have the form

```
<list_parameter_name> = {
  <value_1>,
  <value_2>,
  // ...
  <value_n>
};
```

An example is shown in Example 4.4.

---

<sup>1</sup>Except for the inputs where the same parameter appears more than once. In such cases only the last definition is effective. This (mis)feature should not be relied upon, and it is planned that in the future versions of **niedoida** an attempt to define the same parameter more than once will be reported as error.

---

```
scf = {
  method = rhf;
  energy_threshold = 1e-6;
  density_threshold = 1e-6;
  convergence_accelerator = diis;
};
```

---

Example 4.3: A compound parameter

---

```
atoms = {
  o 0.0000000 0.24618131 0.00000000,
  h 1.4326629 -0.95521837 0.00000000,
  h -1.4326629 -0.95521837 0.00000000
};
```

---

Example 4.4: A list parameter

## 4.4 Parameters

All of the parameters in this section are optional, except for `basis_set` and `atoms`.

Skipping an optional parameter means that the default value is assigned to it. Note: default values are constant, and do not depend on other parameters values. No attempt is made to adjust skipped parameters to those specified in input. This means, for example, that setting multiplicity to 1, and not specifying the SCF method different from the default one (RHF) is reported as input error.

### 4.4.1 Top level

---

**Parameter title**

**Type** string

**Default**

Title of the job.

---

**Parameter run\_type**

**Type** enum (single\_point, geometry\_optimization)

**Default** single\_point

Type of the job.

---

**Parameter** `basis_set`**Type** string**Default**

Name of the atomic basis set. For the mapping between the basis set name and actual definition of the basis set see Section [4.2.1](#).

---

**Parameter** `charge`**Type** integer**Default** 0

Molecular charge.

---

**Parameter** `multiplicity`**Type** positive integer**Default** 1

Multiplicity of electronic state.

---

**Parameter** `atoms`**Type** list**Default**

List of the atoms comprising the system and their coordinates. See Sec. [4.4.2](#)

---

**Parameter** `scf`**Type** compound**Default**

Set of parameters controlling the SCF process. See Sec. [4.4.3](#)

---

**Parameter** `units`**Type** compound**Default**

Specifies the (physical) units system to use for reading input and printing results. See Sec. [4.4.4](#)

**Parameter limits****Type** compound**Default**Set of parameters controlling system usage. ?? See Sec. [4.4.5](#)

---

**Parameter integrals****Type** compound**Default**Integration options. See Sec. [4.4.6](#)

---

**Parameter theory****Type** enum (hf, hfvwn, slater, svwn)**Default** hfBloody theory.

---

**Parameter moller\_plesset****Type** compund**Default**Second order Møller-Plesset options. See Sec. [4.4.7](#)

---

**Parameter cis****Type** compound**Default**Single CI options. See Sec. [4.4.8](#)

---

**Parameter properties****Type** compound**Default**Specifies which properties should be calculated from the SCF wavefunction. See Sec. [4.4.9](#)

---

**Parameter output****Type** compound**Default**

Output options. See Sec. 4.4.10

### 4.4.2 atoms

Parameter `atoms` contains set of atoms from which the molecule is built. It is a list parameter 4.3.2.

Each atom is described in following format:

```
[weight] symbol[_label] x y z
```

where `weight` is a atomic weight of element, `symbol` is a chemical symbol (case insensitive), `label` is arbitrary string and `x y z` are position coefficient of atoms. Parameters in square brackets are optional

---

```
atoms = {  
  16 O 0.0000 0.0000 0.1141,  
  H_alpha 0.0000 0.7803 -0.4563,  
  1 H_beta 0.0000 -0.7803 -0.4563  
}
```

---

Example 4.5: Molecule specification

### 4.4.3 scf

---

**Parameter method****Type** enum (rhf, rohf, uhf)**Default** rhf

Specifies the type of SCF process.

---

**Parameter max\_no\_iterations****Type** positive integer**Default** 30

Maximal number of iterations.

**Parameter** `energy_threshold`

**Type** positive real

**Default**  $10^{-5}$

SCF is not considered converged until the energy difference between two consecutive steps is larger than the threshold.

---

**Parameter** `density_threshold`

**Type** positive real

**Default**  $10^{-5}$

SCF is not considered converged until the density difference between two consecutive steps is larger than the threshold.

---

**Parameter** `convergence_accelerator`

**Type** enum (none, diis, oda)

**Default** diis

The convergence accelerator to use in SCF process.

---

**Parameter** `shift_1`

**Type** positive real

**Default** 0

Activates level shifting. In case of RHF it denotes the shift of virtual orbital energies with respect to the occupied orbitals. In case of ROHF it gives the energy shift of singly occupied orbitals with respect to the doubly occupied orbitals. Finally in case of UHF it denotes the shift of virtual orbital energies with respect to the occupied orbitals for  $\alpha$  spin. Warning! Using level shifting may lead to unphysical results.

---

**Parameter** `shift_2`

**Type** positive real

**Default** same as `shift_1`

In case of ROHF it gives the energy shift of virtual orbitals with respect to the singly occupied orbitals. In case of UHF it denotes the shift of virtual orbital energies with respect to the occupied orbitals for  $\beta$  spin. Skipping this parameter means that it is equal `shift_1`.

---

**Parameter** `initial_guess`**Type** enum (fragments, core\_hamiltonian, from\_file)**Default** fragments

Type of initial guess.

---

**Parameter** `initial_guess_filename`**Type** enum**Default**Name of the file from which MO coefficients are read when `initial_guess = from_file`.

---

**Parameter** `occupations`**Type** enum (aufbau, fermi)**Default** aufbauDetermines how occupation numbers are assigned to molecular orbitals during the SCF procedure. Allowed values are `aufbau` (occupations based on *Aufbau* principle) and `fermi` (smearing of electrons due to Fermi-Dirac distribution of fixed width).

---

**Parameter** `degeneracy_threshold`**Type** positive real**Default**Applies only if `occupations = aufbau`. Then, if the value was given electrons are uniformly smeared in orbitals which energy difference with HOMO is less than the value. If no value was given occupation numbers are always integer, irrespective of possible HOMO quasi-degeneracy.

---

**Parameter** `smear`**Type** positive real**Default** 0.001Applies only if `occupations = fermi`. If so, electrons are distributed according to Fermi-Dirac formula with  $kT$  equal `smear`. Actual Fermi level value is assumed to be  $(E_{HOMO} + E_{LUMO})/2$ .

#### 4.4.4 units

---

**Parameter** energy

**Type** enum (eV, hartree)

**Default** eV

Energy unit to use.

---

**Parameter** length

**Type** enum (bohr, angstrom)

**Default** bohr

Length unit to use.

---

**Parameter** storage

**Type** enum (byte, kilobyte, megabyte, gigabyte)

**Default** megabyte

Storage unit to use.

#### 4.4.5 limits

---

**Parameter** cpu\_time

**Type** positive real?

**Default** 7200

Maximal time of processor running. ?

#### 4.4.6 integration

---

**Parameter** engine

**Type** enum (naive, os1)

**Default** os1

Type of integration engine.

---

**Parameter threshold****Type** positive real**Default**  $10^{-10}$ ?

Integration miscount. ??

---

**Parameter cache\_size****Type** positive integer**Default** 16

Storage capacity. ???

---

**Parameter storage****Type** enum (none, local, shared, in\_core)**Default** none?

The way of integrals storage. ???

#### 4.4.7 mp2

---

**Parameter order****Type** natural**Default** 0

Order of perturbation theory calculations. ??

---

**Parameter memory\_pool****Type** natural**Default** 64

??

---

**Parameter no\_frozen****Type** natural**Default** 0

Number of frozen molecular orbital.

**Parameter** no\_deleted

**Type** natural

**Default** 0

Number of deleted virtual orbitals.

#### 4.4.8 cis

---

**Parameter** multiplicity

**Type** enum (singlet, triplet, both)

**Default** both

Requested multiplicity of excited states.

---

**Parameter** no\_frozen

**Type** natural

**Default** 0

Number of frozen molecular orbital.

---

**Parameter** no\_deleted

**Type** natural

**Default** 0

Number of deleted virtual orbitals.

---

**Parameter** no\_states

**Type** natural

**Default** 0

Number of states.

### 4.4.9 properties

---

**Parameter** `population_analysis`**Type** compound**Default**

Specifies which population analysis should be performed. This compound argument is a collection of simple parameters having the form: `<population_analysis_name> = <bool_value>`, where `<population_analysis_name>` is a name of population analysis and `<bool_value>` should be non-zero to switch the proper analysis on or 0 to switch it off. See Sec. [4.4.9](#).

---

**Parameter** `bond_order_analysis`**Type** compound**Default**

Specifies which bond order analysis should be performed. This compound argument is a collection of substitution of form: `<bond_order_analysis_name> = <bool_value>`, where `<bond_order_name>` is a name of bond order analysis and `<bool_value>` should be non-zero to switch the proper analysis on or 0 to switch it off. See Sec. [4.4.9](#)

---

**Parameter** `max_multipole_moment_order`**Type** natural**Default** ?

Maximal order of multipole moment.???

---

**population\_analyses**

---

**Parameter** `mulliken`**Type** boolean**Default** 1

Controls whether Mulliken population analysis is performed.

---

**Parameter** `lowdin`**Type** boolean**Default** 1

Controls whether Löwdin population analysis is performed.

**Parameter hirshfeld****Type** boolean**Default** 0Controls whether Hirshfeld population analysis is performed.

---

**Parameter voronoi****Type** boolean**Default** 0Controls whether Voronoi population analysis is performed.

---

**Parameter bader****Type** boolean**Default** 0Controls whether Bader population analysis is performed.

---

**bond\_order**

---

**Parameter mayer****Type** boolean**Default** 1Enable Mayer bond order analysis.

---

**Parameter gopinathan\_jug****Type** boolean**Default** 1Enable Gopinathan-Jug bond order analysis.

---

**Parameter nalewajski****Type** boolean**Default** 1

Enable Nalewajski bond order analysis.

#### 4.4.10 output

---

**Parameter** binary

**Type** boolean

**Default** 0

Binary form of output.

---

**Parameter** molden

**Type** boolean

**Default** 0

Generate the MOLDEN output. For more information about MOLDEN see <http://www.cmbi.kun.nl/molden/molden.html>.

## 4.5 Examples

Example 4.6 shows an input for energy calculations of water molecule. Let's analyze it closer.

---

```
title = "water molecule";
```

---

Sets the job title. It does not influence the calculations, but is reproduced in the output. Therefore, it can be used to convey information which makes it easier to identify or analyze output later.

---

```
run_type = single_point;
```

---

Causes the calculations to be performed only for geometry given in input. It is the default, so skipping the line would not change the calculations.

---

```
basis_set = "sto-3g";
```

---

Sets the atomic orbitals basis to be used in calculations.

---

```
units = {  
    length = bohr;  
    energy = hartree;  
};
```

---

Determines units used to interpret input, and units which will be used in output. In this case all lengths and energies are in atomic units.

---

```
atoms = {  
  o  0.0000000  0.24618131  0.00000000,  
  h  1.4326629 -0.95521837  0.00000000,  
  h -1.4326629 -0.95521837  0.00000000  
};
```

---

Determines the stoichiometry and geometry of the molecule for which calculations should be performed. In this case it is a water molecule, and geometry is close to optimal.

---

```
title = "water molecule";  
  
run_type = single_point;  
basis_set = "sto-3g";  
  
units = {  
  length = bohr;  
  energy = hartree;  
};  
  
atoms = {  
  o  0.0000000  0.24618131  0.00000000,  
  h  1.4326629 -0.95521837  0.00000000,  
  h -1.4326629 -0.95521837  0.00000000  
};
```

---

Example 4.6: Water molecule

---

```
title = "ammonia molecule";

run_type = single_point;
basis_set = "sto-3g";

units = {
    length = angstrom;
    energy = hartree;
};

atoms = {
    n    0.0000000000  0.0000000000  0.5841387237,
    h   -0.4702866428  0.8145603594  0.1580719249,
    h   -0.4702866428 -0.8145603594  0.1580719249,
    h    0.9405732855  0.0000000000  0.1580719249
};
```

---

Example 4.7: Ammonia molecule



**Part III**

**Elektrycerz**



## Chapter 5

# Introduction

`elektrycerz` implements the SCPF<sup>[17]</sup> method of the polarization energy calculations.



# Chapter 6

## Invoking `elektrycerz`

The chapter covers all the information necessary to prepare input data and run `elektrycerz`. It is assumed that the program is already installed and configured.

### 6.1 Command-line interface

`elektrycerz` provides typical command-line interface. To run `elektrycerz` an argument should be provided. The argument is treated as the input-file name. Results of calculations are written to the output file. The output file is created in the same directory and has the same name as the input file, but the extension is changed to `.log`.

### 6.2 Input description

The input format is free-form. Spaces, tabs and line breaks are uniformly treated as whitespaces, except for literal strings and comments. Input may contain parameter definitions and comments.

#### 6.2.1 Comments

Comments may appear anywhere in the input. They either start with the `//` string, and end at the end of the line, or start with the `/*` string and end with the `*/` string. An example comment is shown in Example 6.1.

Comments do not influence the calculations in any way.

#### 6.2.2 Parameter definitions

Parameters control calculations performed by `elektrycerz`. By setting them to specific values you decide what and how is calculated when the program is run.

Parameters may be of simple, compound or list type. All parameter definitions have the form

```
<parameter_name>=<parameter_value>;
```

The order of parameter definitions is not significant.<sup>1</sup>

---

<sup>1</sup>Except for the inputs where the same parameter appears more than once. In such cases

```
// A single line comment  
  
/* Another single line comment */  
  
/* A multi-line  
   comment  
*/
```

---

Example 6.1: Comments

## 6.3 Parameters

All parameters are optional, except for `molecules`. Skipping an optional parameter means that the default value is assigned to it. Note: default values are constant, and do not depend on other parameters values. No attempt is made to adjust skipped parameters to those specified in input.

---

### Parameter title

**Type** string

**Default**

Title of the job.

---

### Parameter threshold

**Type** positive real

**Default**  $10^{-5}$

Required accuracy.

---

### Parameter solver

**Type** enum (cg, jacobi, minres)

**Default** cg

Algorithm to be used to solve the SCPF equations.

---

only the last definition is effective. This (mis)feature should not be relied upon, and it is planned that in the future versions of `elektricerz` an attempt to define the same parameter more than once will be reported as error.

---

**Parameter** `damping_factor`

**Type** real in the range (0; 1)

**Default** 0.5

Damping factor. Used only for `jacobi` solver.

---

**Parameter** `model`

**Type** enum (intermolecular, intramolecular)

**Default** intermolecular

Defines the model of microelectrostatic calculations. In the `intermolecular` there are no interactions between submolecules belonging to the same molecule. In the `intramolecular` model, submolecule may interact with every other submolecule, even with those belonging to the same molecule. For this model an explicit list of non-interacting submolecules may be specified.

---

**Parameter** `molecules`

**Type** list

**Default**

Defines the system for which calculations will be performed. The list consists of entries describing molecules. Every molecule is defined as a list of submolecules. A submolecule is defined by its position, polarizability, and, optionally, the charge. For the `intramolecular` model a list of noninteracting submolecules may be specified.



## Chapter 7

# Examples

---

```
title = "tiny fullerene";

max_no_iterations = 30;
threshold = 1e-6;
solver = jacobi;

molecules = {
  {
    {
      // position
      0.0 0.0 0.0
      // polarizability
      606.675271419 0 606.675271419 0 0 606.675271419
      // charge
      -1
    }
  },
  {
    {
      // position
      13.3717020605 13.3717020605 0.0
      // polarizability
      606.675271419 0 606.675271419 0 0 606.675271419
      // no charge
    }
  },
  {
    {
      0.0 13.3717020605 13.3717020605
      606.675271419 0 606.675271419 0 0 606.675271419
    }
  },
  {
    {
      13.3717020605 0.0 13.3717020605
      606.675271419 0 606.675271419 0 0 606.675271419
    }
  }
};
```

---

Example 7.1: Four fullerene molecules, each represented as a single submolecule

# Appendix A

## License

Copyright (c) 2004, 2005, 2006, 2007 Grzegorz Mazur, Marcin Makowski, Witold Piskorz, Lukasz Cwiklik, Mariusz Sterzel, Mariusz Radon, Waldemar Kulig, Daniel Blazewicz

### LICENSE

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to use it for conducting scientific research.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.



# Appendix B

## HDF5 License

Copyright Notice and Statement for NCSA Hierarchical Data Format (HDF)  
Software Library and Utilities

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities  
Copyright 1998, 1999, 2000, 2001, 2002, 2003, 2004 by the Board of Trustees  
of the University of Illinois. All rights reserved.

Contributors: National Center for Supercomputing Applications (NCSA) at the  
University of Illinois at Urbana-Champaign (UIUC), Lawrence Livermore  
National Laboratory (LLNL), Sandia National Laboratories (SNL), Los Alamos  
National Laboratory (LANL), Jean-loup Gailly and Mark Adler (gzip library).

Redistribution and use in source and binary forms, with or without  
modification, are permitted for any purpose (including commercial purposes)  
provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,  
this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,  
this list of conditions, and the following disclaimer in the documentation  
and/or materials provided with the distribution.
3. In addition, redistributions of modified forms of the source or binary  
code must carry prominent notices stating that the original code was  
changed and the date of the change.
4. All publications or advertising materials mentioning features or use of  
this software are asked, but not required, to acknowledge that it was  
developed by the National Center for Supercomputing Applications at the  
University of Illinois at Urbana-Champaign and to credit the contributors.
5. Neither the name of the University nor the names of the Contributors may  
be used to endorse or promote products derived from this software without  
specific prior written permission from the University or the Contributors,

as appropriate for the name(s) to be used.

6. THIS SOFTWARE IS PROVIDED BY THE UNIVERSITY AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no event shall the University or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

---

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.

DISCLAIMER:

This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

---

# Appendix C

## PETSc License

### COPYRIGHT NOTIFICATION

(C) COPYRIGHT 1995-2004 UNIVERSITY OF CHICAGO

This program discloses material protectable under copyright laws of the United States. Permission to copy and modify this software and its documentation is hereby granted, provided that this notice is retained thereon and on all copies or modifications. The University of Chicago makes no representations as to the suitability and operability of this software for any purpose. It is provided "as is" without express or implied warranty. Permission is hereby granted to use, reproduce, prepare derivative works, and to redistribute to others, so long as this original copyright notice is retained.

### Software authors

Mathematics and Computer Science Division  
Argonne National Laboratory,  
Argonne IL 60439 FAX: (630) 252-5986  
Any questions or comments on the software may be directed  
to [petsc-maint@mcs.anl.gov](mailto:petsc-maint@mcs.anl.gov).

Argonne National Laboratory with facilities in the state of Illinois, is owned by The United States Government, and operated by the University of Chicago under provision of a contract with the Department of Energy.

### DISCLAIMER

THIS PROGRAM WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED BY AN AGENCY OF THE UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES GOVERNMENT NOR ANY AGENCY THEREOF, NOR THE UNIVERSITY OF CHICAGO, NOR ANY OF THEIR EMPLOYEES OR OFFICERS, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS. REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCT, PROCESS, OR SERVICE BY TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE, DOES NOT NECESSARILY CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF. THE VIEW AND OPINIONS OF AUTHORS EXPRESSED HEREIN DO NOT NECESSARILY STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF.

# Appendix D

## TAO License

### COPYRIGHT NOTIFICATION

(C) COPYRIGHT 1998-2005 UNIVERSITY OF CHICAGO

This program discloses material protectable under copyright laws of the United States. Permission to copy and modify this software and its documentation is hereby granted, provided that this notice is retained thereon and on all copies or modifications. The University of Chicago makes no representations as to the suitability and operability of this software for any purpose. It is provided "as is" without express or implied warranty. Permission is hereby granted to use, reproduce, prepare derivative works, and to redistribute to others, so long as this original copyright notice is retained.

Software authors

Citation: Steve Benson, Lois Curfman McInnes, Jorge More, Todd Munson and Jason Sarich,

TAO User Manual, Preprint ANL/MCS-TM-242 Revision 1.9,  
Mathematics and Computer Science Division,  
Argonne National Laboratory, 2007.

Argonne IL 60439 FAX: (630) 252-5986

Any questions or comments on the software may be directed to  
tao-comments@mcs.anl.gov.

Argonne National Laboratory with facilities in the state of Illinois, is owned by The United States Government, and operated by the University of Chicago under provision of a contract with the Department of Energy.

## DISCLAIMER

THIS PROGRAM WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED BY AN AGENCY OF THE UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES GOVERNMENT NOR ANY AGENCY THEREOF, NOR THE UNIVERSITY OF CHICAGO, NOR ANY OF THEIR EMPLOYEES OR OFFICERS, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS. REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCT, PROCESS, OR SERVICE BY TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE, DOES NOT NECESSARILY CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF. THE VIEW AND OPINIONS OF AUTHORS EXPRESSED HEREIN DO NOT NECESSARILY STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF.

## Appendix E

# Boost License

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



## Appendix F

# Pearls of Wisdom

C++: an octopus made by nailing extra legs onto a dog.  
-- Anonymous

FORTRAN was the language of choice for the same reason that  
three-legged races are popular.  
-- Ken Thompson

There's no Moore law for the productivity of programmers.  
-- Robert Dewar

\* Me: My brain is burnt out after trying to follow what the iterators  
\* have to do.  
\* Joerg: That's Dr. Stepanov's gift for us ;-)  
-- Boost developer

DFT is assuming a local external potential ....  
-- Bernd Schimmelpfennig

Failure is not an option. It comes bundled with your Microsoft product.  
-- Derek Harkness

A bug in the code is worth two in the documentation.  
-- Anonymous

If the only thing you know is a hammer, everything looks like a nail.  
-- Anonymous

We always know what we're doing -- we're superheroes, we're C++ programmers.  
-- Aaron W. LaFramboise

The use of COBOL cripples the mind; its teaching should, therefore, be  
regarded as a criminal offence.  
-- Edsger W. Dijkstra, SIGPLAN Notices, Volume 1

Inefficient abstractions are a dime a dozen.

-- Andrei Alexandrescu

# Bibliography

- [1] Grzegorz Mazur, Marcin Makowski, Witold Piskorz, Łukasz Ćwiklik, Mariusz Sterzel, Mariusz Radoń, Waldemar Kulig Barbara Jagoda-Ćwiklik, and Daniel Błażewicz. Niedoida 0.2, 2007. 3
- [2] Steve Benson, Lois Curfman McInnes, Jorge Moré, Todd Munson, and Jason Sarich. TAO user manual (revision 1.9). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2007. <http://www.mcs.anl.gov/tao>. 5
- [3] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. <http://www.mcs.anl.gov/petsc>. 5
- [4] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004. 5
- [5] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997. 5
- [6] H. C. Longuet-Higgins and J. A. Pople. The electronic spectra of aromatic molecules iv: Excited states of odd alternant hydrocarbon radicals and ions. *Proc Phys Soc*, A68:591, 1955. 11
- [7] Alojzy Gołębiewski. *Chemia Kwantowa Związków Organicznych*. PWN, 1973. 11
- [8] R.S. Mulliken. Electronic population analysis on LCAO-MO molecular wave functions. i. *J Chem Phys*, 23:1833–1840, 1955. 11
- [9] R.S. Mulliken. Electronic population analysis on LCAO-MO molecular wave functions. ii. overlap populations, bond orders, and covalent bond energies. *J Chem Phys*, 23:1841–1846, 1955. 11
- [10] R.S. Mulliken. Electronic population analysis on LCAO-MO molecular wave functions. iii. effects of hybridization on overlap and gross ao populations. *J Chem Phys*, 23:2338–2342, 1955. 11

- [11] R.S. Mulliken. Electronic population analysis on LCAO-MO molecular wave functions. iv. bonding and antibonding in lcao and valence-bond theories. *J Chem Phys*, 23:2343–2346, 1955. 11
- [12] I. Mayer. On bond orders and valences in the ab initio quantum chemical theory. *Int J Quantum Chem*, 29:73–84, 1986. 11
- [13] I. Mayer. Charge, bond order and valence in the ab initio scf theory. *Chem Phys Lett*, 97:270–274, 1983. 11
- [14] Roman F. Nalewajski and Janusz Mrozek. Modified valence indices from the two-particle density matrix. *Int J Quantum Chem*, 51:187–200, 1994. 11
- [15] R. F. Nalewajski, J. Mrozek, and G. Mazur. Quantum chemical valence indices from the one-determinantal difference approach. *Can J Chem*, 74(6):1121–1130, 1996. 11
- [16] Janusz Mrozek, Roman F. Nalewajski, and Artur Michalak. Exploring bonding patterns of molecular systems using quantum mechanical bond multiplicities. *Pol J Chem*, 72:1779–1791, 1998. 11
- [17] D. B. Knowles and R. W. Munn. Polarization energy calculations in molecular crystals. *J Mat Sci*, 5:89–93, 1994. 33

# Index

- boost, 5, 6
- CIS, 11
- gcc, 5
- HDF5, 5, 6
- MOLDEN, 27
- MP2, 11
- MPI
  - LAM/MPI, 5, 6
  - MPICH2, 5
- parameter
  - atoms, 17
  - bader, 26
  - basis\_set, 17
  - binary, 27
  - bond\_order\_analysis, 25
  - cache\_size, 23
  - charge, 17
  - cis, 18
  - convergence\_accelerator, 20
  - cpu\_time, 22
  - damping\_factor, 37
  - degeneracy\_threshold, 21
  - density\_threshold, 20
  - energy, 22
  - energy\_threshold, 20
  - engine, 22
  - gopinathan\_jug, 26
  - hirshfeld, 26
  - initial\_guess, 21
  - initial\_guess\_filename, 21
  - integrals, 18
  - length, 22
  - limits, 18
  - lowdin, 25
  - max\_multipole\_moment\_order, 25
  - max\_no\_iterations, 19
  - mayer, 26
  - memory\_pool, 23
  - method, 19
  - model, 37
  - molden, 27
  - molecules, 37
  - moller\_plesset, 18
  - mulliken, 25
  - multiplicity, 17, 24
  - nalewajski, 26
  - no\_deleted, 24
  - no\_frozen, 23, 24
  - no\_states, 24
  - occupations, 21
  - order, 23
  - output, 19
  - population\_analysis, 25
  - properties, 18
  - run\_type, 16
  - scf, 17
  - shift\_1, 20
  - shift\_2, 20
  - smear, 21
  - solver, 36
  - storage, 22, 23
  - theory, 18
  - threshold, 23, 36
  - title, 16, 36
  - units, 17
  - voronoi, 26
- PETSc, 5
- population analysis, 11
  - Bader, 11
  - Hirshfeld, 11
  - Löwdin, 11
  - Mulliken, 11
  - Voronoi, 11
- python, 5
- RHF, 11
- ROHF, 11
- rubber, 5

scons, 5

TAO, 5

TeX, 5

UHF, 11

valence analysis, 11

    Gopinathan-Jug, 11

    Mayer, 11

    Nalewajski-Mrozek, 11