# 1 Overview

COLUMBUS and MOLCAS cooperate by exchanging well-defined, transferable quantities, specifically AO integrals, AO density matrices and MO coefficients in the native MOLCAS format by using library functions from the MOLCAS library. Additionally, the RunFile is processed, which is used by MOLCAS to store additional information often required to be passed in between different MOLCAS modules. MOLCAS provides some mechanism to incorporate external programs and making them accessible through the standard MOLCAS input file.

Since it is necessary to link a portion of the MOLCAS library, both COLUMBUS and MOLCAS must be compiled and linked in a compatible way, that is to say (i) the **same** compilers and (ii) **compatible** compiler options. Hence, both codes cooperate on a binary level and do not rely on some file conversion utilities.

Please note, that the MOLCAS library is frequently restructured, so that files produced with a previous version may or may not be compatible with the previous version of MOLCAS (cf. page 48).

# 2 Binary Distributions

Starting from version 7.1 COLUMBUS can be obtained in the form of a special binary distribution suitable for operation in combination with MOLCAS. This contains only a subset of the COLUMBUS binaries along with `columbus.exe`, a wrapper program that is accessed by the MOLCAS driver `molcas.exe`.

The installation procedure is as follows:

1. download the tar ball with the binaries for COLUMBUS 7.1 (`colmoldistribution_<DATE>.tgz`)
2. unpack the tar ball on the target system
   (`tar -xzvf colmoldistribution_<DATE>.tgz`);
3. change to the directory `colmoldistrib`, copy your MOLCAS license file into this directory and execute `./usersetup.sh`; this script adjusts some settings depending on your target system architecture. The license file is necessary, because the MOLCAS driver program `MOLCAS/molcas.exe` runs license checks.
4. execute some test and verification programs
   `./runtests.sh -mpitest.x`
   ensures that the mpi version included in the distribution operates properly on your system
   `./runtests.sh -seriel`
   runs a few test cases with the included COLUMBUS and MOLCAS binaries including various single-point calculations and structure optimizations
   `./runtests.sh -parallel`
   runs a few test cases with the included (seriel) MOLCAS and parallel COLUMBUS version

Executing `./runtests.sh` without option runs all three groups of test cases automatically.

5. To this end `COLUMBUS` and `MOLCAS` binaries are operational on the target system; however, you may want to use your own complete and possibly parallel or free version of `MOLCAS` instead of the stripped subset functionality included in the tar-ball.
   To do so run
   `./usersetup.sh --local-molcas-version=<absolute path to MOLCAS installation>`
   Subsequently rerun the test and verifaction programs (cf. step (4))

**Technical Details**

For the execution with the precompiled `MOLCAS` binaries it is necessary to have `molcas.rte` to contain the following lines - as produced automatically by `usersetup.sh`

```
RUNBINARY= LD_LIBRARY_PATH=/bigscratch/colmoldistrib/syslibs '$program'
RUNBINARYSER= LD_LIBRARY_PATH=/bigscratch/colmoldistrib/syslibs '$program'
RUNSCRIPT='$program $input'
```

In addition `usersetup.sh` copies `syslibs/ld-linux-x86-64.so.` to `/tmp/ld-linux-x86-64.so.2`. This ensures, that libc and the dynamic loader are consistent with older Linux distributions. Otherwise, you would typically see a segmentation violation or a relocation error during execution of the `MOLCAS` binaries:

```
relocation error: ... libc.so.6: symbol _dl_find_dso_for_object, version GLIBC_PRIVATE not defined
in file ld-linux-x86-64.so.2 with link time reference
```

Corresponding settings for the `COLUMBUS` binaries are automatically taken care for by `columbus.exe` - a statically linked binary.

# 3 Execution

Make sure, that both the startup script for molcas (`molcas`) as well as the `COLUMBUS` subdirectory of the colmoldistribution are added to your `$PATH` environment variable.

Prepare some input file (cf. the `TESTS` subdirectory for examples) with the naming convention `<project_name>.input`

`runcolmol <project_name>`

`runcolmol` is a small shell script that may be adjusted to suit your particular needs which calls the `MOLCAS` driver.

# 4 Compatibility with COLUMBUS 7.0

It is necessary to set the environment variable `COLUMBUS_VERSION=FORCE_7.0` in order to notify `columbus.exe` to assume 7.0 binaries since 7.1 operates in a slightly different manner.

There is a set of 7.0 binaries in the `COLUMBUS.7.0` subdirectory, however the link `COLUMBUS` per default points to the `COLUMBUS.7.1` subdirectory. Reset the link to the 7.0 binaries (`rm`

`COLUMBUS; ln -s COLUMBUS.7.0 COLUMBUS`). At the time of writing the verification procedure may incorrectly detect failure.

Using self-compiled versions of 7.0 is equally possible by copying the respective binaries to the `COLUMBUS.7.0` subdirectory. For the parallel version, modify the MPI variable settings in `COLUMBUS.7.0/runcolmol` to reflect the path to the MPI installation. Using your own version also set the environment variable `COLUMBUS_RELOCAL=NONE`.

# 5 Requirements/Incompatabilities

Operational combinations of hardware/software stack for a given tar-ball. Asterix marks the combination used for creation. MOLCAS and COLUMBUS executables compiled with the same compiler (ifort/icc) version.

| OS | kernel | glibc | perl | bash | arch | Columbus | Molcas | status |
|---|---|---|---|---|---|---|---|---|
| SUSE 13.1 | 3.11.6-4 | 2.18 | 5.18.1 | 4.2 | x86_64 | 7.1.r1.0.0* | 8.1.14-06-26 | ok |
| SUSE 13.2 | 3.16.6 | 2.19 | 5.20.1 | 4.2 | x86_64 | 7.1.r1.0.0 | 8.1.14-06-26 | ok |
| CentOS 5.11 | 2.6.18 | 2.5 | 5.8.8 | 4.1.2 | x86_64 | 7.1.r1.0.0 | 8.1.14-06-26 | 1) |
| CentOS 6.7 | 2.6.32 | 2.12 | 5.10.1 | 4.1.2 | x86_64 | 7.1.r1.0.0 | 8.1.14-06-26 | ok |
| CentOS 7.0 | 3.10.0 | 2.17 | 5.16.3 | 4.2.46 | x86_64 | 7.1.r1.0.0 | 8.0.15-11-11 | ok |
| SUSE 13.1 | 3.11.6-4 | 2.18 | 5.18.1 | 4.2 | x86_64 | 7.1.r1.0.0* | 8.0.15-11-11 | ok |
| SUSE 13.1 | 3.11.6-4 | 2.18 | 5.18.1 | 4.2 | x86_64 | 7.1.r1.0.1* | 8.0.15-11-11 | ok |

[1] kernel too old.

Compatibility of pre-compiled Columbus binaries with a particular Molcas version while using a different Molcas version at run time.

| (in Col) | Molcas binaries | | | |
|---|---|---|---|---|
| | 7.9_111 | 8.0.15-11-01 | 8.1.14-06-26 | 8.1.15-11-30 |
| 7.9_111 | ok | | | |
| 8.0.15-11-01 | partial[1] | ok | ok | partial[1] |
| 8.1.14-06-26 | partial[1] | ok | ok | partial[1] |

[1] alaska fails to read the effective density and fock matrices since binary format has changed.

# 6 Verification procedure

The `TESTS` subdirectory contains for each test case the input file (`*.input`), a reference output file (`*.output`) , an input file for the verification script (`*.config`) which contains the pair of file names to be tested for consistency along with a variety of tests in terms of a regex expressions. The output of each verification run can be found in `*.verifyls`.

## 6.1 Semantics of the configuration file

The configuration file is parsed line by line. Comment lines start with `#`, lines specifying the pair of files to be compared start by `%` and the pair of files including relative or absolute paths

are separated by `:`. The following lines are considered as one test specification per line refering to the last specified pair of files. Hence, tests following another specification line for the pair of files refer to the latter.

Each test specification consists of 3 or 5 columns separated by `!`.

The first column specifies the regex pattern of the line(s) to be compared.

The second column specifies the column of the data in these lines to be tested (assuming separation of columns by white space).

The third column indicates the comparison threshold.

The pattern of the fourth and fifth column constrain the search for the pattern of column 1 to a range of lines starting with the first appearance of the pattern in column four and ending with the first subsequent appearance of the pattern in column five.

# 7 Trouble Shooting

`COLUMBUS` 7.1 differs considerable from its predecessor both in terms of technical implementation as well as in terms of algorithmical details. Hence, although all test cases pass the test criteria, only a subset of the possible usage combinations are covered.

There are two types of technical failures:

1. operatings system specific issues
   Both `COLUMBUS` and `MOLCAS` binaries are not fully statically linked but still require the `glibc` libraries (the basic system libraries of a Linux system). The `syslibs` subdirectory contains the proper versions of all necessary shared libraries and the default setup is to use these shared libraries whenever a `COLUMBUS` or `MOLCAS` binary is executed.
   Note, that a few years ago, there was a major change in the linux kernel moving from version 2.6 to version 3.x. Current Linux distributions such as SUSE.13.x and CentOS 7.x are based on the 3.x kernel, while older releases are based on 2.6 kernel. Running binaries compiled under the 3.x kernel may or may not run on 2.6 kernel based systems. With exception of `molcas.exe`, which is supplied solely as a binary by the `MOLCAS` developers, all other binaries can be provided as fully statically linked binaries (at the expense of increased size).
   Parallel `COLUMBUS` codes are linked statically to avoid additional dependencies on the MPI runtime scripts. However, even so, a consistent libc version of `gethostbyname` must exist on the target system.
2. interoperability issues
   They may arise if the `COLUMBUS` version is combined with your own `MOLCAS` version. Incompatibility arises here from (i) different (incompatible) file structure of your own `MOLCAS` version typically giving rise to message complaining about non-existing or unreadable files or (ii) possible inconsistencies when mixing binaries produced by different compilers.

# 8 Revision Log

r1.0.0 initial version open for testing

r1.0.1
- corrected -mpitest.x option
- corrected call of parallel Columbus codes in columbus.exe
- added support to read LUMORB format 2.0

r1.0.2
- added support for changes in the initialization of integral routines for 8.1 newer than March 2015
- added support for generalized slapaf bookkeeping handling single state optimizations; corresponding minimalistic changes to 8.1 sources commited on 15/12/08.
- depending on the molcas version string the codes switches between the usage of `tran.x` and `tran.x_81`.

r1.0.4
- Several bugfixes and optimizations in the Columbus part.
- Switched from mpich1.2.7 to mpich2-1.4.1.
  Note, that mpich1.2.7 is script based while mpich2-1.4.1 is partially based on binaries loading shared libraries. This might produce failures with older operating system distributions.
- The fix from feb 9,2016 prevents a failure due to the incorporated BLAS libraries on a system with avx2 registers and has missing links added to the mpich2/bin subdirectory.

r1.0.5
- Several bugfixes and optimizations in the Columbus part, especially fixing the `daio: maxrec exceeded`, due to an incorrect file size estimate (thanks to N. Bogdanov, to point out the problem and provision of a test case).
- This version changes for some additional files to compression, thereby reducing the disk and memory space requirements. Per default the parallel CI code keeps everything in memory while the seriel CI code uses disk space, instead. Although, it is not yet moved in to the interface, there is the possibility to drastically reduce the memory consumption for the subspace vectors forcing them to disk (ciudgin: fileloc=1,0,0 to be added manually, currently) - for the parallel code, consider using a SSD then. Vice versa, the seriel code can be forced to run fully in memory (ciudgin: fileloc=1,1,1 to be added manually).
- Early termination of the CI code due to incorrect evaluation of integral distribution size fixed.

r1.1.0 Major changes in the low-level operation in order to facilitate multi- and many-core operation:

- Global Array Toolkit completely replaced by a small library directly utilizing shared memory functionality

– global shared counter replaced by a faster scheme based on semaphores which does not suffer from saturation effects
– new integral storage format for more efficient storage of integrals, densities and ci vectors reducing disk usage for extended systems by up to 90%
– new parallelization strategy that largely decouples load balancing from the associated communication volume leading to a reduction of the network load by more than one order of magnitude

Thomas Müller August 23, 2016

Jülich Supercomputing Centre
Institute of Advanced Simulation
Forschungszentrum Jülich
D-52425 Jülich.