

Part I

Introduction

Section 1

Introduction to COLUMBUS

1.1 COLUMBUS, Quantum Chemistry Package

COLUMBUS is a general purpose quantum-chemical program package specialized on generally applicable one and two-component multi-reference methods, in particular MCSCF, MR-SDCI, and MR-AQCC. The availability of general analytical gradients and the corresponding non-adiabatic coupling vectors for one-component MCSCF, SA-MCSCF, MR-CISD and MR-AQCC calculations is an out-standing feature. Note, that the choice of MCSCF and MRCI reference spaces is completely independent from each other and not restricted to a particular form such as CAS or RAS type CSF spaces.

COLUMBUS operates in its entirety within the framework of the GUGA approach and, hence, in a basis of spin-adapted configuration state functions. This leads in particular to certain advantages for two-component MR-CISD calculations within the perturbational approach. Spin-orbit CI calculations may be based on spin-orbit pseudo potentials (e.g. those by M. Dolg et.al.) or on the (abinitio) DKH/AMFI approach.

With the growing number of correlated electrons, the size of the configuration space increases rapidly and quickly reaches $\mathcal{O}(N^8)$ CSFs and more, so that an efficient parallel implementation is essential. COLUMBUS parallelization scheme is utilizing the Global Arrays Toolkit for dynamic load balancing and one-sided communication and is capable of running MRCI calculations with dimensions up to 3 billion CSFs with - for today's standards - modest resource requirements.

The manual partly comprises detailed material in *small print* describing technical aspects which can be skipped on first read.

Part II

Short Guide to COLUMBUS

1.2 Short Guide to COLUMBUS

Part III

User's Guide to COLUMBUS

Section 2

The MOLCAS-COLUMBUS link

2.1 Overview

The interfaces exploit the modular nature of both COLUMBUS and MOLCAS packages and are based upon the exchange of *well-defined quantities* (integrals, derivative integrals, density matrices as well as other simple data such as molecular orbital coefficients, structural data, basis set information, etc.). These data are directly accessed by high-level MOLCAS library routines linked into the COLUMBUS modules. Thus, COLUMBUS inherits the capability to read and write binary MOLCAS data formats in contrast to other concepts, that rely on some converter tool and maintain the same data in different representations. For ASCII files, such as MO coefficient files it is trivial to read/write data in the MOLCAS native format.

However, it is not possible to exchange *ill-defined* data such as n-electron wavefunctions, which cannot necessarily exactly and easily be interconverted between MOLCAS and COLUMBUS representations.

The Cholesky-Decomposition (CD) scheme is currently not supported¹.

The following two subsections serve as a brief overview on the differences concerning execution under control of COLUMBUS and MOLCAS, respectively.

2.1.1 Execution of COLUMBUS under control of MOLCAS

While the input to MOLCAS modules remains unchanged, in addition there appears input for the (external) COLUMBUS modules. The script like features of the MOLCAS input language remain usable.

The COLUMBUS version (7.1) shipped for the operation under control of the MOLCAS driver is a slightly modified version of the stand-alone COLUMBUS package containing a subset of various modules, only. While the stand-alone version has all features of COLUMBUS available, only a subset of the features coming with MOLCAS can be accessed there in an automatic or semi-automatic way under control of the COLUMBUS driver. With the partially stripped version running under control of the MOLCAS driver it is rather the opposite - all features of MOLCAS are necessarily available but not all of the functionality of COLUMBUS can be used in a sensible manner.

The high-level interface is implemented such, that it automatically generates the necessary COLUMBUS input files from the simplified input description in the MOLCAS input file and calls the respective COLUMBUS modules in the appropriate order. In order to have more control over the COLUMBUS modules while

¹The most conventional way to partially use the CD scheme is to rerun SEWARD prior to COLUMBUS with CD disabled.

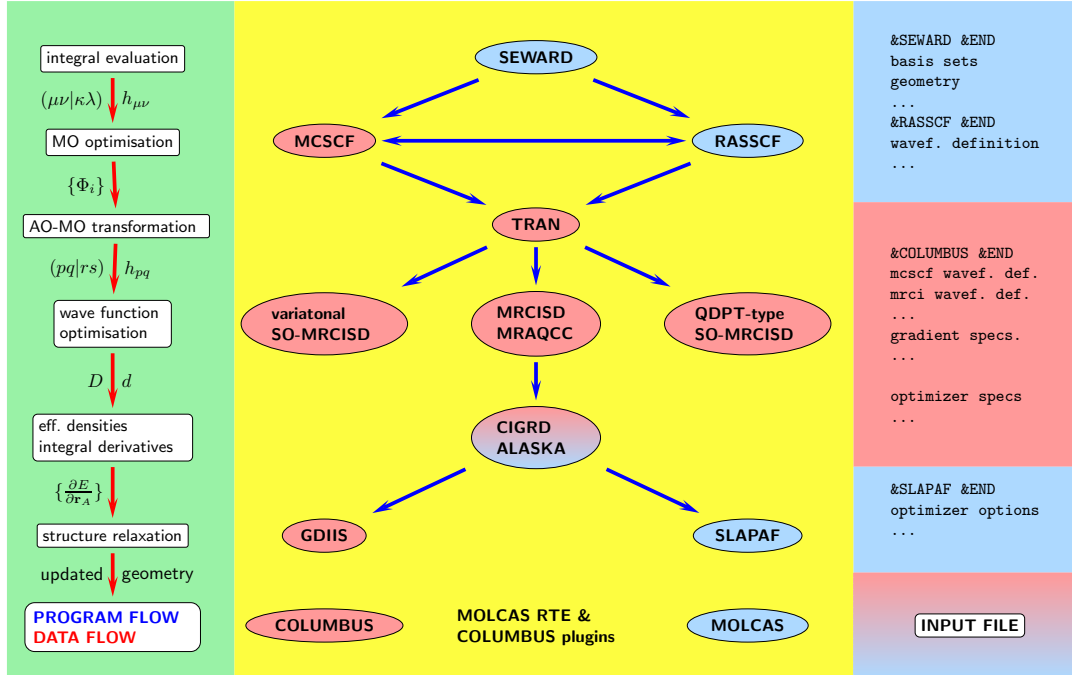


Figure 2.1: Schematically depicted data and program flow for mixed operation of COLUMBUS and MOLCAS under control of the MOLCAS-driver. The depicted work flow refers to single-point energy evaluations and structure optimizations. MOLCAS-input files in principle also allow for much more complex work flows.

avoiding a rather cumbersome MOLCAS input file, it is recommended to follow the subsequent workflow, if MOLCAS style input is not flexible enough:

1. prepare a closely related MOLCAS style input and add the TEST keyword in the general COLUMBUS section²
2. change to the \$project/WORK directory and modify the respective input files in line with the COLUMBUS documentation in the COLUMBUS/docs subdirectory
3. replace the keyword TEST by NOAUTO and re-run the job³

This procedure is primarily useful for specialized options to some COLUMBUS modules and for the construction of more specialized CSF spaces.

2.1.2 Execution of MOLCAS under control of COLUMBUS

From COLUMBUS version 7.0 the COLUMBUS driver facility **runc** supports single-point energy calculations as well as the evaluation of analytical gradients at SA-MCSCF and MR-CISD/MR-AQCC levels of theory. Also support is added to use RASSCF calculations in a variety of ways and to automatically run consistent CASPT2 calculations such that the results can be compared directly. While some (limited) support is added to generate all required COLUMBUS and MOLCAS input files directly with the COLUMBUS input facility **colinp**, the use of special MOLCAS program options requires manual intervention anyway.

²TEST stops execution prior to running the MCSCF or MRCI code after creation of all input files including the DRTs specifying the wavefunction

³NOAUTO assumes that a valid \$project/WORK directory exists containing valid input.

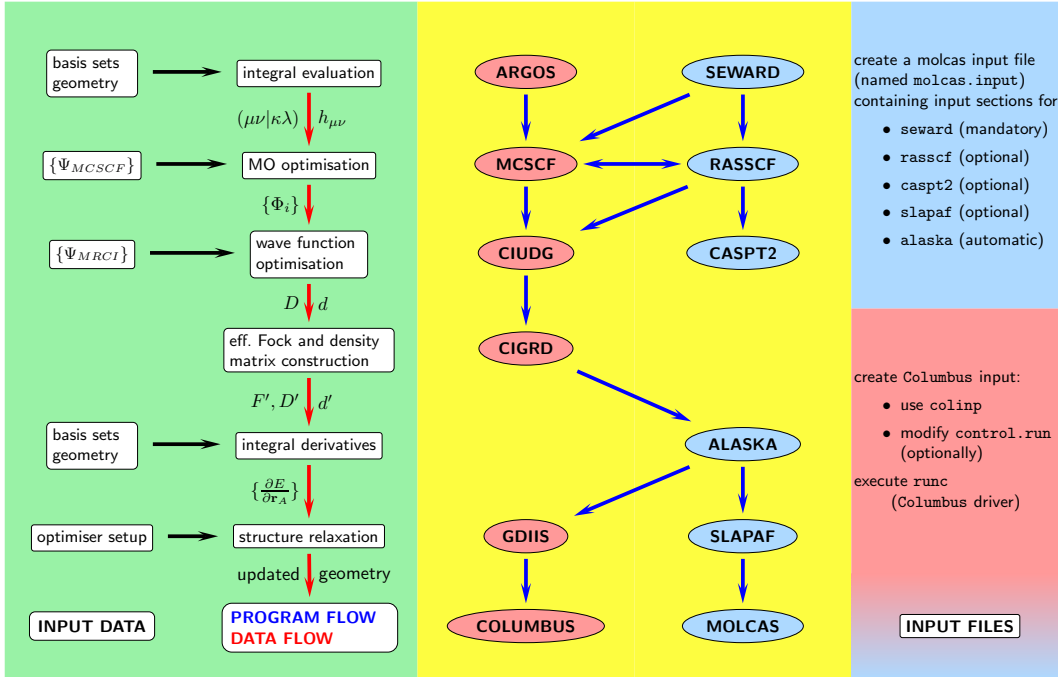


Figure 2.2: Schematically depicted data and program flow for mixed operation of COLUMBUS and MOLCAS under control of the COLUMBUS-driver. The depicted work flow refers to single-point energy evaluations and structure optimizations.

Apart from using the integrals and derivatives of the integrals w.r.t. geometric displacements (i.e. the MOLCAS modules **SEWARD** and **ALASKA**), it is also convenient to use the **RASSCF** module for orbital optimization or the **SLAPAF** module for a wide variety of structure optimization schemes. Note, that for analytical gradients, it is **not** possible to directly start from MOs generated with the **RASSCF** module, instead it must be followed by a single **MCSCF** (macro) iteration and the same CSF space with the COLUMBUS **MCSCF** code in order to properly evaluate the analytic gradients.

Hence, the following examples illustrate the possibilities:

- Run a single-point MR-CISD calculation using the DKH Hamiltonian to describe scalar-relativistic effects.
- Run a single-point, variational SO-MR-CISD calculation using the DKH Hamiltonian and atomic mean field integrals to describe spin-orbit coupling.
- Run a single-point, QDPT-type SO-MR-CISD calculation using the DKH Hamiltonian and atomic mean field integrals to describe spin-orbit coupling.
- Run a large scale **RASSCF** calculation followed by MR-CISD/MR-AQCC (with an appropriately reduced reference space).
- Use **RASSCF** to pre-optimize MOs for a subsequent COLUMBUS **MCSCF** calculation
- Use **RASSCF** followed by **CASPT2** and MR-CISD/MR-AQCC to compute energies at different levels of theory, that can be directly compared.
- Use the COLUMBUS analytic gradient and nonadiabatic coupling feature, to optimize structures of minima on the conical intersection seam

2.2 Configuration space definition

COLUMBUS is operating in a basis of spin-adapted configuration state functions (CSFs) which are encoded in terms of a distinct row table (DRT).

In order to construct a CSF space constrained to a given number of electrons, spin multiplicity and spatial symmetry, both MOLCAS and COLUMBUS rely on the concept of active spaces: orbital subsets are defined along with occupation number restrictions. The CSF space is defined by all possible CSFs meeting the occupation number restrictions. This is the familiar CAS or RAS1/RAS2/RAS3 or more general GAS. The flexibility of configuration space definition goes far beyond that, as even individual CSF (in MCSCF) or internal CSF (in MRCI) can be selected or excluded from the CSF space.

2.2.1 MCSCF configuration spaces

The MCSCF CSF space is characterized by the number of electrons, spatial symmetry, spin multiplicity and the definition of frozen core (FC), doubly occupied (DOCC), active and virtual orbitals (VO). Frozen core orbitals are unaffected by the MCSCF orbital optimization and are always doubly occupied. In contrast the DOCC or inactive orbital space is optimized. The virtual orbitals are unoccupied throughout and the space spanned by the VOs changes during optimization. The remaining active orbitals are further decomposed into restricted active space (RAS), complete active space (CAS) and auxiliary space (AUX). In MOLCAS notation this is just RAS1, RAS2 and RAS3. Other specialized CSF spaces such as perfect pairing or generalized valence bond type could also be defined. However, they are not supported in the simplified input scheme.

A note of caution is in place, however: the more a CSF space deviates from a CAS, the more implicit assumptions are made with respect to consistent MOs and consistent MO ordering. If these assumptions are inconsistent or impossible to satisfy (e.g. wrong size and partitioning into RAS1, RAS2 and RAS3 subspaces), orbital optimization will fail or the final wavefunction is not the expected one.

- **FC:** Frozen core orbitals are always doubly occupied and, hence, electrons are not correlated.

The remaining orbitals are divided into

- **DOCC:** These (inactive) orbitals are always doubly occupied in all CSFs.
- **RAS:** The restricted active space is fully occupied with a maximum number of holes in any CSF.
- **CAS:** The complete active space may contain any number of electrons in any CSF.
- **AUX:** The auxiliary orbital space contains a maximum number of electrons in any of the reference configurations.
- **VO:** The virtual orbital space is empty throughout the CSF space.

The ordering is bottom-up within each irreducible representation.

We illustrate the configuration space definition and its precise meaning for furan to compute the 1A_1 ground and the 1^3B_1 excited state including resulting in 28 and 24 CSFs, respectively. (cc-pVDZ)

	a_1	b_1	b_2	a_2	comment
FC	0	0	0	0	no SCF orbitals retained
DOCC	9	6	0	0	σ space
RAS	0	0	0	0	
CAS	0	0	3	2	π -CAS
AUX	0	0	0	0	
VO	26	25	10	9	
total	35	31	13	11	

2.2.2 MRCI configuration spaces

The total CSF space of an MR-CISD calculation is obtained by all single and double excitations from a given set of reference CSFs constrained by the requested symmetry. The **COLUMBUS MRCI** program allows for a large degree of flexibility in defining reference configuration spaces. The orbital space is divided into internal and external orbitals. The internal orbitals are normally occupied in at least one reference configurations whereas the external orbitals contain at most two electrons in any configuration and do not contribute to any reference CSF. As for efficiency reasons only the internal orbital space is explicitly encoded there is only very limited control over the external orbital space.

Certain orbitals may be eliminated completely from the subsequent calculation: The contribution of the electrons in this orbital subspace are transformed into an effective fock matrix and repulsion term and thus lead to a considerable reduction of both the CSF space dimension and the number of electron repulsion integrals. Note of caution: freezing virtual orbitals requires a suitable resolution to avoid significant errors. The QMC resolved virtual orbitals of an MCSCF calculation are not necessarily a good starting point, better choices are e.g. natural orbitals of some previous more restricted MRCI calculation may be more advisable.

- **FC:** Frozen core orbitals are always doubly occupied and, hence, electrons are not correlated.
- **FV:** Frozen virtual orbitals are always unoccupied and, hence, do not contribute to electron correlation.

For reference configuration space definition the remaining orbitals are - in analogy to the MCSCF configuration space definition - further subdivided into

- **REFDOCC:** These (inactive) orbitals are always doubly occupied in all reference configurations.
- **REFRAS:** The restricted active space is fully occupied with a maximum number of holes in any of the reference configurations.
- **REFCAS:** The complete active space may contain any number of electrons for the different reference configurations.
- **REFAUX:** The auxiliary orbital space contains a maximum number of electrons in any of the reference configurations.

The ordering is bottom-up within each irreducible representation.

We illustrate the configuration space definition and its precise meaning for ethylen to compute the ground and the first excited state including a small amount of σ polarisation.

	A_{1g}	B_{1g}	B_{2g}	B_{3g}	A_u	B_{1u}	B_{2u}	B_{3u}	comment
FC									
DOCC									
REFRAS									
REFCAS									
REFAUX									
virtual									
FV									
total									

There is additional flexibility to define reference and final configuration spaces adding various other occupation number and cumulative spin coupling constraints resulting e.g. in GVB, PP and other direct product type spaces. However, these are not supported in the simplified input section.

2.2.3 SO-MRCI configuration spaces

The SO-MRCI method is built as an extension of the non-relativistic mechanism based on CSFs. [Absatz aus dem PERTCI paper plus some info from Yabushita](#)

2.2.4 Approximately Size-Extensive MR-techniques

The COLUMBUS-MRCI code also includes several closely related methods that are approximately size-extensive. The methods can be characterized by the availability of total energies, analytic gradients and transition densities. Some methods are by construction variational or perturbational and further distinguished by being state-specific (separate optimization for each state) or state-universal tied to one particular reference state. While MR-AQCC, LRT-MR-AQCC and MR-ACPF crucially depend on the quality of the reference *wave function*, TE-MR-AQCC depends only on the choice of the reference *space*. LRT-MR-AQCC is a state-universal variant to the state-specific MR-AQCC providing also transition densities.

method	constraints	energy	gradients	transition densities
MR-AQCC	variational, state-specific	yes	yes	no
MR-ACPF	variational, state-specific	yes	yes	no
LRT-MR-AQCC	variational, reference state	yes	yes	yes
TE-MR-AQCC	variational, state-specific	yes	no	no
MRCI+Q	perturbational, state-specific	yes	no	no

2.3 Analytical Gradients

Analytical gradients of the total energy with respect to geometrical distortions are available at the SS-MCSCF, SA-MCSCF, MR-CISD and MR-AQCC level of theory. Although the MCSCF configuration space may be chosen identical to the corresponding definition in the RASSCF module, it is not possible to run analytical gradients (neither for MCSCF nor MR-CISD/MR-AQCC) starting from the output of the RASSCF module. Although the converged RASSCF molecular orbitals may be used as a starting point, it is always necessary to run at least single COLUMBUS MCSCF calculation in order to generate the properly resolved orbitals and to compute the orbital hessian along with additional internal data needed by the effective Fock and density matrix construction step. If the definition of RASSCF and MCSCF configuration spaces matches, this amounts to just an additional single MCSCF calculation, otherwise

the RASSCF orbitals are just the starting guess for the actual MCSCF calculation on which the gradient is ultimately based. Frequently, it is a sensible and efficient strategy to pre-optimize with RASSCF, followed by a MCSCF calculation.

[References to the gradients and papers](#)

2.4 Non-adiabatic coupling vectors

2.5 Structure Optimization

Structure optimization decomposes into three aspects:

- the definition of suitable coordinates for optimization including optimization constraints such as freezing coordinates
- type of characteristic point on the PES to be searched for
- the update scheme to generate an improved geometry for the next step in the optimization cycle

Table Structure optimizations

The recommended default scheme is to use SLAPAF saddle point and minima searches along with the available features for coordinate definition and update scheme. A couple of input examples are given below. This procedure is applicable to situations, where SLAPAF processes the gradient of a single electronic gradient only. In case of searches for the minimum on an intersystem crossing (i.e. absence of non-adiabatic coupling terms for the pair of states), the input consists of two gradients (one per electronic state) plus the energies. As of the time of writing SLAPAF is not able to recognize the corresponding COLUMBUS information. The same applies to search for the minimum of a crossing seam (i.e. including non-adiabatic coupling terms for the pair of states).

In these situations one has to resort to the internal coordinate definition and structure optimization modules coming with COLUMBUS.

COLUMBUS internally relies on a Newton-Raphson procedure starting with diagonal Hessian where the diagonal elements are set according to some heuristic rules. The Hessian is updated from the analytical gradients in the subsequent optimization steps.

2.6 The COLUMBUS MCSCF program

While the MCSCF code is of completely general nature in the sense, that an arbitrary configuration space may be selected, this leads to much fewer possibilities to exploit efficient matrix linear algebra, while the more limited configuration spaces in RASSCF boil down to endless matrix operations. The differences are particularly apparent for large CSF spaces. On the other hand the RASSCF code optimizes the wavefunction by the super CI method[1], effectively evaluating solely gradient terms with respect to the wavefunction parameters (CI and MO coefficients). While this allows for efficient implementation and modest resource requirements, the convergence rate is frequently disappointingly slow requiring hundreds of macro iterations. The COLUMBUS MCSCF code explicitly constructs the second order derivatives with respect to CI and MO coefficients. While this improves convergence, it drastically increases computational

cost, disk space and memory usage. Also, the mcsf code is currently limited to at most 1 million CSFs but it is certainly not advisable to combine large CSF spaces with large basis sets.

The MCSCF codes offers state-averaged MCSCF calculations where the constituting states are not constrained to the same symmetry or spin multiplicity. This allows to run SA-MCSCF calculations for high symmetry systems within the highest possible Abelian pointgroup without breaking the proper symmetry of the molecular orbitals and correspondingly of the electronic states.

2.7 The COLUMBUS MRCISD program

The COLUMBUS MRCI program may be executed after molecular orbital optimization by means of the SCF, RASSCF or MCSCF programs. In fact the energy can be evaluated based on any input orbitals with a matching CSF space definition. This includes both standard non-relativistic or scalar relativistic MRCI calculations as well as two-component spin-orbit CI calculations[2].

The COLUMBUS MRCI program generates Multi Reference SDCI[2], ACPF[3], AQCC[4] and AQCC-v[5] wavefunctions. AQCC and ACPF belong to a class of approximately size-extensive functionals applicable to the multi reference case. The program is based on the Direct CI method[6]. Coupling coefficients are generated on the fly with the Graphical Unitary Group Approach[7].

The COLUMBUS MRCI program also generates state-specific natural orbitals that can be fed into the property program to evaluate certain one electron properties. The natural orbitals are also useful for Iterated Natural Orbital (INO) calculations.

The program is particularly well-suited for large configuration spaces. Several eigenvectors can be computed simultaneously for MR-SDCI calculations while approximate size-extensive functionals are state-selective.

Functionals

The many-electron wavefunction is expanded in terms of spin-adapted configuration state functions (CSFs) Φ_i . The full CI space is partitioned into four disjoint subspaces P , Q' , Q and R . P denotes the reference configuration space, spanned by a subset of all configurations that can be obtained by distributing the electrons over the internal (active+inactive) orbitals, only. Q' denotes all single and double excitations out of $\Phi_i \in P$ not contained in P with all external orbitals unoccupied (also termed internal configurations) while Q contains those single and double excitations not contained in either P nor Q' . R finally denotes all CSFs not contained in either P , Q and Q' .

The full CI wavefunction reads

$$\Psi^{FCI} = \Psi_P + \Psi_{Q'} + \Psi_Q + \Psi_R = \sum_{i \in P} c_i^P \Phi_i^P + \sum_{i \in Q'} c_i^{Q'} \Phi_i^{Q'} + \sum_{i \in Q} c_i^Q \Phi_i^Q + \sum_{i \in R} c_i^R \Phi_i^R \quad (2.1)$$

The uncontracted MR-CISD expansion is truncated to $\Psi_P + \Psi_{Q'} + \Psi_Q$ with all $c_i^P, c_i^{Q'}, c_i^Q$ optimized independently. As the truncated MR-CISD expansion is not size-extensive, a rapidly increasing differential electron correlation error is introduced beyond about 10 correlated electrons. Approximately size-extensivity corrected methods restore size-extensivity by incorporating the effect of higher excitations from the R space in an approximate way. MR-AQCC includes the effect of disconnected quadruple excitations and may be considered as an approximation to MR-CCSD. As a variational method with respect to the wavefunction expansion coefficients analytical gradients can be implemented efficiently exploiting the Hellmann-Feynman theorem[?]. Very accurate data for excitation energies, equilibrium bond lengths

and harmonic frequencies have been obtained at the CBS limit in a benchmark study[8, 9]. Thus, it is the method of choice for many-electron targets.

Here, we focus on a family of functionals of the state-specific correlation energy ΔE_α defined as the difference between the energy of the reference wavefunction E_0^α and the total Energy E .

$$F_\alpha(\mathbf{c}^\alpha) = \frac{(\sum_i c_i^\alpha \Phi_i | \hat{H} - E_0^\alpha | \sum_i c_i^\alpha \Phi_i)}{\sum_{i \in P, Q'} (c_i^\alpha)^2 + G \sum_{i \in Q} (c_i^\alpha)^2} \quad (2.2)$$

The available energy functionals differ in the normalization of the denominator and the associated G -values are given as

method	g
MR-CISD	1
MR-ACPF	$\frac{2}{N}$
MR-AQCC	$1 - \frac{(N-3)(N-2)}{N(N-1)}$
MR-AQCC-v	$1 - \frac{(N-3)(N-2)(V-3)(V-2)}{N(N-1)V(V-1)}$

N denotes the number of correlated electrons N and V the number of virtual (unoccupied) MOs.

Note, that wavefunctions for ground and excited states of the same symmetry based on the size-extensivity corrected functionals are not orthogonal since the reference energy defines different Hamiltonians.

Diagonal Shift Technique

The above energy functionals can be cast into an MR-CISD eigenvalue problem with diagonal CI matrix element shifted by Δ_0 [5],

$$\langle \Phi_i | (\mathcal{H} - E_0^\alpha + \underbrace{\sum_{k \notin \text{int}} (1 - G) \Delta E_0 | \Phi_k \rangle \langle \Phi_k |}_{\Delta_0} \sum_j c_j^\alpha \Phi_j) = \Delta E^\alpha c_i^\alpha \quad (2.3)$$

The projection operator $\sum_{k \notin \text{int}} | \Phi_k \rangle \langle \Phi_k |$ ensures that solely matrix elements of non-internal CSFs (CSFs with electrons in external orbitals or with excitations out of the internal orbitals doubly occupied in *all* reference CSFs) are modified. The method-specific constant G is given in the table above with N being the number of *correlated* electrons. In case of MRAQCC, $\Delta E_0 = \Delta E_0^\alpha$ is the correlation energy computed with the MRAQCC functional w.r.t. the energy of the reference wavefunction. Since ΔE_0^α occurs on both sides of the equation, it is computed iteratively by reinserting the current estimate until convergence. Owing to the similarity of the MR-CISD and their approximately size-extensivity corrected counterparts and the diagonal shift technique, the wavefunction optimization can be carried out by the standard Davidson subspace scheme using modified subspace representations of the CI and Overlap matrices[5]. Hence, the overhead compared to MR-CISD is negligible.

LRT-AQCC

LRT-MRAQCC[10], a perturbative extension to the state-selective MRAQCC method[4], offers a natural way to consistently derive approximately size-extensive ground and excited state energies as well as

transition densities. Hence, diagonal and off-diagonal matrix elements of the model space Hamiltonian can be described consistently.

In case of LRT-MRAQCC, ΔE_0 is the MRAQCC correlation energy of the reference state, i.e., LRT-MRAQCC is no longer a state-specific functional and the electronic states are mutually orthogonal as they share the same Hamiltonian.

TE-AQCC

While the above functionals depend explicitly through E_0^α on the reference wavefunction, total energy AQCC replaces the functional of the correlation energy by a functional of the total energy and E_0^α is substituted by the relaxed reference energy \tilde{E}_0^α defined as

$$\tilde{E}_0^\alpha = \frac{\langle \sum_{i \in P} c_i^\alpha \Phi_i^\alpha | \hat{H} | \sum_{i \in P} c_i^\alpha \Phi_i^\alpha \rangle}{\sum_{i \in P} (c_i^\alpha)^2} \geq E_0^\alpha \quad (2.4)$$

It follows that $\hat{\Delta}$ is amended by an off-diagonal term $\hat{\tilde{\Delta}}$ over the reference space, only.

$$\hat{\tilde{\Delta}} = \sum_{l \in P} \sum_{k \in P} \left(|\Phi_l\rangle \langle \Phi_l| \frac{(G-1) \sum_{i \in Q} (c_i^\alpha)^2}{\sum_{i \in P} (c_i^\alpha)^2} (\hat{H} - \tilde{E}_0^\alpha) |\Phi_k\rangle \langle \Phi_k| \right) \quad (2.5)$$

TE-AQCC is sensitive to the choice of the *reference space*, only, in spirit similar to the multi-reference analogs of the SDCI type Davidson correction[?],

$$E^{CI+Q1} = E_{CI}^\alpha + (E_{CI}^\alpha - \tilde{E}_0^\alpha)(1 - c_0^2) \quad (2.6)$$

$$E^{CI+Q2} = E_{CI}^\alpha + (E_{CI}^\alpha - \tilde{E}_0^\alpha) \frac{1 - c_0^2}{c_0^2} \quad (2.7)$$

$$E^{CI+Q3} = E_{CI}^\alpha + (E_{CI}^\alpha - \tilde{E}_0^\alpha) \frac{1 - c_0^2}{2c_0^2 - 1} \quad (2.8)$$

where $c_0^2 = \sum_{i \in P} c_i^\alpha \Phi_i^\alpha$ denotes the relaxed weight of the reference space in the final normalized MR-CISD wavefunction.

2.7.1 Performance Issues

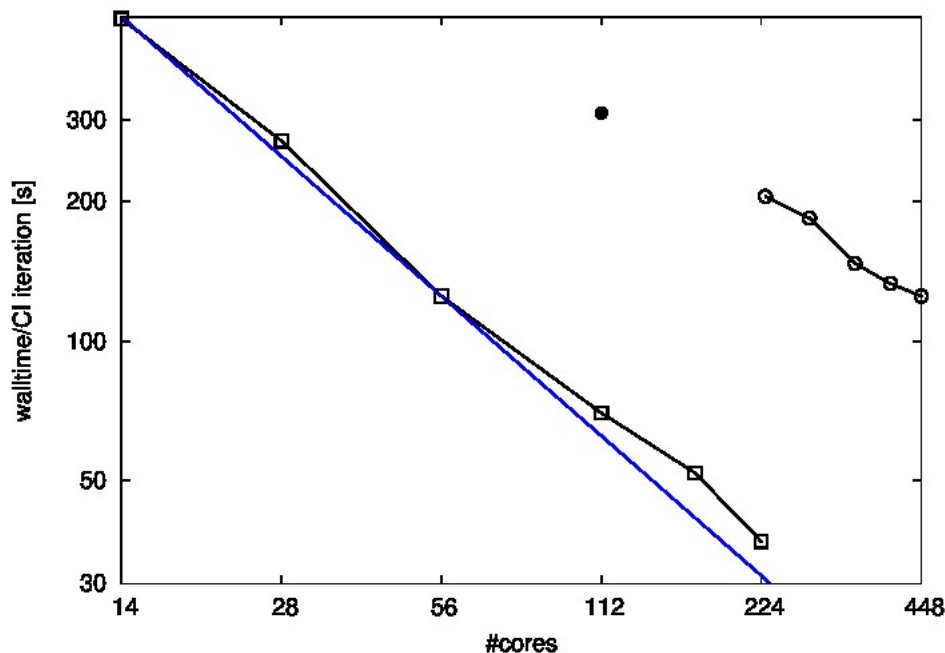


Fig. 2: Figure 2: Parallel performance of the COLUMBUS MR-CISD code (JUROPATEST, each node is supplied with 2 Intel Xeon CPU E5-2695 v3 CPUs with 14-cores per CPU, 2.3 GHz, 128 GB DDR4 memory, Infiniband FDR HCA network). Wall clock time (seconds) per Davidson iteration for (i) Ozone ground state, MR-CISD,cc-pV5Z, ref. CAS(12e,9o), C2v, 4.11 108 CSFs (boxes); (ii) Ozone ground state, MR-CISD,cc-pV5Z, ref. CAS(18e,12o), C2v, 15108 CSFs (circles); (iii) Cu- Benzene, variational SO-MRCISD, cc-pVDZ, ref CAS(11e,6o)CAS(4e,4o), C2v, 1 109 CSFs (bullets). Ideal scaling (blue line).

The strong scaling performance plot in the above figure demonstrates that almost ideal scaling can be achieved. Since the performance data have been carried out on a test system for the upcoming successor of the existing general purpose system JUROPA, certain machine characteristics affecting the overall performance (eg. memory and process pinning) are not perfectly adjusted giving rise to slight irregularities in the performance data. On the existing general purpose system JUROPA, the scaling is almost ideal up to approximately 1000 cores. Please note, that up to version 7.0 the performance characteristics for a large number of cores is mostly determined by the ability of the network to cope with random point-to-point traffic. This dependency is reduced at version 7.1.

Performance tuning used to be a tricky procedure and has now become somewhat automated. The key questions answered in the following paragraphs are

- why is performance tuning strongly case and machine dependent
- what is the underlying performance model
- how are case and machine dependencies handled

Parallel direct CI: segmented matrix-vector products

From Revision 1.1.0 onwards the parallelization strategy has fundamentally changed and the subsequent discussion and description does not yet reflect these changes.

The direct CI approach[?] using the Davidson diagonalization [?, ?] replaces the explicit construction of the CI matrix \mathbf{H} by an iteratively improved subspace representation of the problem. It relies on an efficient formation of matrix vector products $\sigma = \mathbf{H}\mathbf{v}$ which uses up about 99% of the computer time. The GUGA scheme naturally divides the configuration space into Z, Y, X and W type CSFs corresponding to CSFs with zero, one, two triplet and singlet coupled electrons in the external orbital space, respectively. In second quantized form, the Hamiltonian can be written in terms of one and two electron MO integrals and the generators of the Unitary Group

$$\hat{H} = \sum_{ij} h_{ij} \hat{E}_{ij} + \frac{1}{2} \sum_{ijkl} (ij|kl) (\hat{E}_{ij} \hat{E}_{kl} - \delta_{jk} \hat{E}_{il}) \quad (2.9)$$

Evaluating the matrix element $\langle \Phi_m | \hat{H} | \Phi_n \rangle$ of the CI matrix \mathbf{H} thus reduces to the computation of the respective coupling coefficient $\langle \Phi_m | \hat{E}_{ij} | \Phi_n \rangle$, $\langle \Phi_m | \hat{E}_{ij} \hat{E}_{kl} - \delta_{jk} \hat{E}_{il} | \Phi_n \rangle$ and their contraction with the MO integrals. Hence, the σ vector formation can be decomposed into non-vanishing contributions. Each contribution can be classified by a task type $T_{s,s'}^I$ characterized by a class of integrals I (classified according to the number of external orbital indices (0,1,2,3,4)) and a pair of segment types s, s' ($s, s' \in Z, Y, X, W$) as indicated in Figure 3. The individual task $T_{\tilde{s}, \tilde{s}'}^I$ carries the specific segment indices \tilde{s}, \tilde{s}' . Four-external integrals have non-vanishing contributions to CSFs sharing the same internal path and, hence, can occur only within a single segment $\tilde{s} = \tilde{s}'$. With multiple segments per segment type an almost arbitrary number of independent tasks can be generated and evaluated in any order. Each task requires random access to the segments $\sigma_{\tilde{s}}$ and $v_{\tilde{s}'}$ along with a single pass through all integrals of class I . At the same time the *local* memory demands for each task are reduced to the size of the segment pair (σ_i, v_i) plus some buffer space for integrals. Up to version 7.0 3- and 4-external integrals are stored in distributed memory while 0-,1-,2-external integrals are available as local copy in the memory of each process. From version 7.1 all integrals are stored in shared memory as one copy per node, so that there is no integral-related internode data communication. In addition, the entire subspace expansion with a preset maximum dimension n_{vmax} v_i and σ_i vectors is stored in distributed memory, giving rise to $2n_{vmax}n_{CSF}/n_{core}$ double precision words storage requirements per core (or per process).

Partitioning the N_s CSFs of type s into n_s segments divides the total work of $T_{ss'}^I$ into $n_s n_{s'}$ tasks each reading and writing one segment as well as running a single pass through all integrals of class I . The total inter-node data transfer volume $V_{ss'}^I$ (in Bytes) depends at most quadratically on the number of segments for the integral contribution and linearly for the σ and trial vector component taking into account that 0-,1- and 2-external integrals are kept as local copies (version ≤ 7.0):

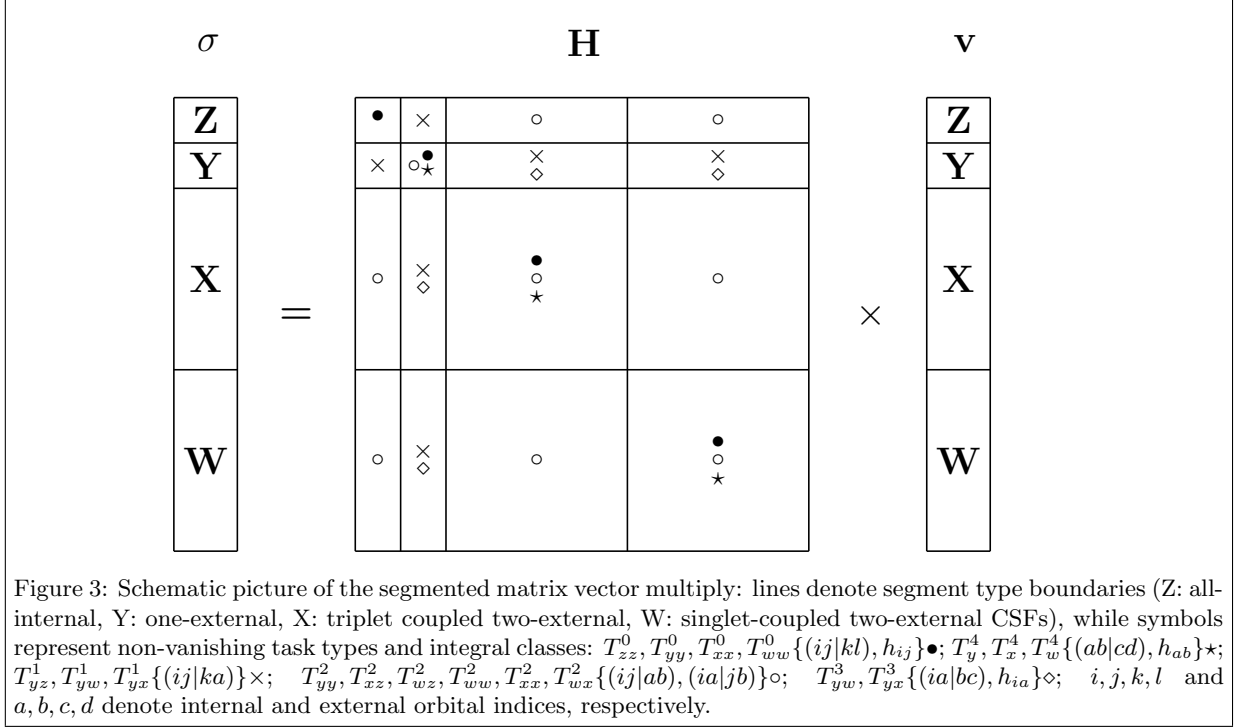
$$V_{ss'}^I = 8(n'_s N_s + n_s N'_s) I \leq 2 \quad (2.10)$$

$$V_{ss'}^I = 8(n'_s N_s + n_s N'_s) + n_s n'_s (n_{int})(n_{ext})^3 \quad I = 3 \quad (2.11)$$

$$V_s^4 = 16N_s + n_s (n_{ext})^4 \quad I = 4 \quad (2.12)$$

In case of version 7.1 the second term in each line does not contribute to the internode data transfer, since one copy per node is stored.

To feed n_{core} cores, ideally it is sufficient to generate $\mathcal{O}(n_{core})$ tasks (i.e. $n_s = n'_s = \mathcal{O}(\sqrt{n_{cores}})$) so that the total communication volume scales $\mathcal{O}(n_{cores}^\alpha)$ with $\frac{1}{2} \leq \alpha \leq 1$ and the *average* communication volume per CPU *drops* with increasing processor usage. To be more precise, while up to version ≤ 7.0 $V_{ss'}$ depends quadratically (through the 3-external integrals) on the number of segments n_s , version 7.1 depends at most linearly on n_s .



So far, we have anticipated a rather uniform distribution of the computational cost per CI matrix element $\mathbf{H}_{i,j}$. Owing to the structure and sparsity of the CI matrix this is not the case. The total execution time $t_{\bar{s}\bar{s}'}^I$ for a task $T_{\bar{s}\bar{s}'}^I$ decomposes into

$$t_{\bar{s}\bar{s}'}^I = t_c + t_\sigma + t_d \approx \alpha n_c + \beta n_u + V/\gamma \quad (\text{I}=0,1,2,3) \quad (2.13)$$

$$t_s^4 = t_\sigma + t_d \approx \beta' n_{CSF} + V/\gamma \quad (\text{I}=4) \quad (2.14)$$

(i) the data transfer time t_d is given by $V_{\bar{s}\bar{s}'}^I$ divided by some effective network bandwidth γ , (ii) the evaluation time t_d of the internal contribution to the coupling coefficient is on average proportional to their number of non-vanishing number of GUGA loops n_c and (iii) the formation the matrix vector product t_σ . The formation of the σ vector makes extensive use of linear algebra and t_σ is on average proportional to the number of valid upper walks n_u while the prefactor β depends on task type, point group symmetry, n_{ext} and processor characteristics (e.g. cache, clock frequency). t_σ can vary by many orders of magnitude even among tasks of the same type! Both n_u and n_c depend solely on reference space definition, the number and ordering of internal orbitals and the number of correlated electrons. Both quantities are entirely independent of the basis set size. The T_s^4 tasks are special insofar the computational cost is uniformly spread among all tasks and proportional to the length of the CI segment.

Load balancing

Figure 4 (referred to as cost matrix) displays n_u for selected task types at a resolution of 150 x 150 segments. Since the cost matrix is independent of the basis set size, each segment is characterized by the range of internal walks included. The number of encoded CSFs for each segment is approximately given by scaling the number of internal walks by a factor of 1 (Z), n_{ext}/n_{irrep} (Y) and n_{ext}^2/n_{irrep} (X,W). Figure 4 shows that the cost matrix differs by up to 6 orders of magnitude per segment block in a non-continuous pattern and the percentage of non-vanishing blocks varies from 65% to 7% for the different task types. It is quite obvious, that the nonuniform distribution of load generally prevents load-balancing based on some fixed number of *equal-sized* segments and we need - especially for large core numbers - to

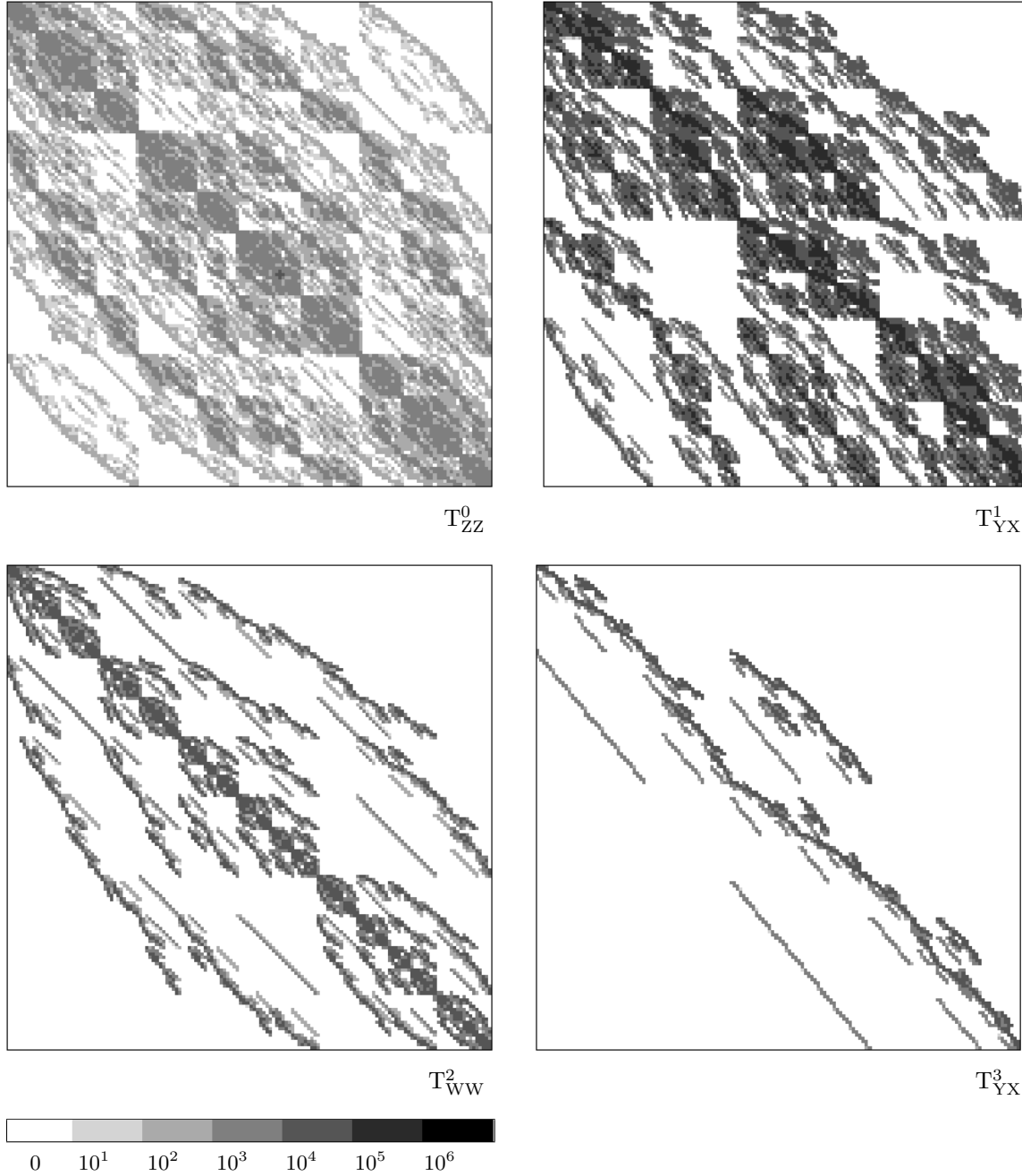


Figure 2.3: Cost matrix for a CAS(12,12) with all inactive electrons frozen (D_{2h}). The logarithmic gray scale of the patches encodes the relative work load required to compute the respective subblock of the CI matrix. The resolution is 150×150 blocks per task type T_{ss}^I , corresponding to 192 Z, 2266 Y 1883 X and 1132 W type encoded internal walks per block. For the EXT D basis with ≈ 35 orbitals per irrep, each block encodes 192 Z, 79×10^3 Y, 288×10^3 X or 173×10^3 W configurations.

specify in addition to the number of segments also the segment boundaries. For a small number of cores (say 32-64)- especially on a single SMP node-, however, it is fortunately sufficient to use a straightaway reasonable initial guess for the necessary number of equal-sized segments which works in combination with dynamic load balancing. Specifying both number of segments as well as boundaries for calculation employing a larger number of cores is usually untractable to do manually.

Thus, an iterative scheme has been devised to generate a list of non-vanishing tasks from the cost matrix and scaling factors constrained to minimum inter node data transfer volume, optimum load balancing, maximum memory usage and number of cores. It is essentially an optimization problem relying on a heuristic procedure to avoid an exponential scaling. Such a scheme relies on reasonably reproducible task timings, an assumption that does not hold in the case of network saturation or the inability of the network to cope with a large amount of random data traffic inevitably arising with dynamic load balancing. Also competition of multiple data-intense processes on a single node for memory and cache can produce undesirable fluctuations of the individual task timings. While the basis set independent cost matrix can be directly computed from the DRT the scaling factors α, β are both basis set *and* machine dependent. Hence, it is necessary to carry out a boot-strapping technique: first set-up the calculation for a small number of cores, run a few CI-iterations, derive the scaling factors from the performance data, redo the setup for a range of cores and pick the most appropriate setup for the given situation.

2.8 Input Description and Examples

2.8.1 Dependencies

The program needs the full set of one- and two-electron integrals generated by the program **SEWARD**. Currently, Cholesky decomposition schemes *are not* supported. It is usually recommended to generate initial orbitals by RHF (**SCF**) in closed shell cases or by MCSCF calculations (**RASSCF**).

2.8.2 Files

Input files

<i>File</i>	<i>Contents</i>
ORDINT	Two-electron integrals from SEWARD .
ONEINT	One-electron property integrals from SEWARD .
ONEREL	Relativistic one-electron integrals from SEWARD .
AMFI	Relativistic one-electron atomic mean field integrals from SEWARD .
LUMORB	Molecular orbital coefficients in ASCII format as generated e.g. by SCF or RASSCF .

Output files

<i>File</i>	<i>Contents</i>
<i>nocoef_ci.drtn.m</i>	The natural orbitals for CI root m of DRT n where n and m are integers.
<i>civout.drtn</i>	The CI vector information file for the calculation of the n th DRT.
<i>civfl.drtn</i>	The CI vector file containing all roots for the calculation of the n th DRT.
<i>ciudgls.drtn</i>	The output of the CI calculation for DRT n.
<i>ciudgsm.drtn</i>	The short summary of the results for DRT n.
<i>molcasinfo</i>	Symmetry and basis set information extracted from MOLCAS files.

Note that these file names are the FORTRAN file names used by the program, so they have to be mapped to the actual file names. This is usually done automatically in the MOLCAS system. This does not apply to the output files.

Local files

<i>File</i>	<i>Contents</i>
<i>cidrtin,cidrtmsin</i>	These automatically generated input files contain the configuration space definition for the CIDRT.X and CIDRTMS.X which write the DRT files <i>cidrtfl.drtn</i> .
<i>cisrtin</i>	The automatically generated input file for the integral sorting step.
<i>ciudgin</i>	The automatically generated input file for the COLUMBUS MRCI program.

These input files can be edited manually for special options (refer to the original **COLUMBUS** documentation). To avoid overwriting the modified input files use the **NOAUTO** key word.

2.8.3 Input

This section describes the input to the **COLUMBUS MRCI** interface in the MOLCAS program system, with the namelist input:

```
&COLUMBUS &END
```

Keywords

Keywords must be provided with their full name.

The following is a list of compulsory keywords:

<i>Keyword</i>	<i>Meaning</i>
END OF INPUT	This marks the end of the input data.

There are three different sections in the COLUMBUS input, which are bracketed by the section headers(**BS_***) and trailers (**ES_***), respectively.

<i>Keyword</i>	<i>Meaning</i>
BS_GEN	This marks the begin of the keywords for the general section, such as memory consumption, number of CPUs etc.
ES_GEN	This marks the end of the keywords for the general section.
BS_MRCI	This marks the begin of the keywords for the CI wavefunction specification and options to the MRCI code.
ES_MRCI	This marks the end of the keywords for the MRCI part
BS_MCSCF	This marks the begin of the keywords for the MCSCF wavefunction specification and options to the MCSCF code.
ES_MCSCF	This marks the end of the keywords for the MCSCF part
BS_GRAD	This marks the begin of the keywords for the analytical MRCI or MCSCF gradient.
ES_GRAD	This marks the end of the keywords for gradient specifications

The wavefunction specifications for both MCSCF and MRCI are internally implemented in terms of Distinct Row Tables (DRTs).

In case of state-averaged MCSCF calculations, there may be multiple DRTs differing in spatial symmetry and spin multiplicity and - in principle- even different CSF spaces. In this interface implementation, multiple DRTs are constrained to the same orbital subspace definitions and occupation number restrictions.

In case of MRCI it is frequently of interest to evaluate multiple MRCI wavefunctions differing in terms of spin multiplicity or spatial symmetry, in order to compute a reasonably complete overview over excited states or to find a point of intersystem crossing. In this interface implementation, multiple DRTs are - just as for MCSCF - constrained to the same orbital subspace definitions and occupation number restrictions.

Hence, within the sections marked by **BS_MCSCF**, **ES_MCSCF**, **BS_MRCI**, **ES_MRCI** there is at least a single DRT specification marked by **DRT** and **END_DRT**. DRT specifications may carry a DRT specific label for further references.

The allowed DRT specification keywords are

<i>Keyword</i>	<i>Meaning</i>
DRT	marks the begin of a DRT specification
SPIN	This indicates the spin multiplicity of the CI or MCSCF wavefunction. In case of SO-MRCI wavefunctions, this value indicates the maximum spin multiplicity to be included (cf. to Annex).
SYMM	This indicates the spatial symmetry of the non-spin-orbit CI or MCSCF wavefunction; in case of spin-orbit CI wavefunctions, this includes the spin symmetry. Please refer for the specific details to the section on spin-orbit CI calculations.
REFSYMM	The following line lists all allowed reference wavefunction irreps. Note, that the final configuration space is built from all EXLVL excitations of correct symmetry from all reference wavefunctions, even though the latter are not necessarily of correct symmetry. MRCI section, only.
NROOT	For MCSCF and MRCISD calculations the following line specifies the number of lowest electronic states of this spin multiplicity and symmetry to be computed. For state-specific MRAQCC, MRACPF and MRAQCC-V calculations the next line is interpreted as the electronic state to be optimized, where the maximum overlap of the wavefunction with the reference function #NROOT is the selection criterion. For several states with MRAQCC, MRACPF and MRAQCC-V just specify multiple DRT sections with the respective NROOT entries of interest (cf. test33p.input)
ELECTRONS	There are three numbers to be read in. The first one specifies the total number of electrons <i>including</i> doubly occupied orbitals (MCSCF) and frozen core orbitals (MRCI). The next two numbers refer to occupation restrictions of orbital subspaces used for the definition of the MCSCF space (CAS, RAS, AUX) or equivalently for the MRCI reference space (REFCAS, REFRAS, REFAUX). Hence, the second one the number of electrons in the complete active subspace (CAS, REFCAS, in MOLCAS notation RAS2) and the third one the maximum number of holes in the restricted active subspace (RAS, REFRAS, in MOLCAS notation RAS1) and the maximum number of electrons in the auxiliary active subspace (AUX, REFAUX, in MOLCAS notation RAS3).
LABEL	Specifies an arbitrary, though unique ASCII label for the current DRT specifications.
END_DRT	marks the end of a DRT specification

The following is a list of keywords for the general section.

<i>Keyword</i>	<i>Meaning</i>
BS_GEN	marks the begin of the general section applying to all COLUMBUS modules.
TITLE	The following line is treated as title line.
NCPU	indicates the number of available CPUs (or equivalently cores)
MEMORY	available physical memory per core in MB; recall that the operating system also fancies some memory.

PRINT	indicates the print level
TEST	Test the input. The input is processed, the DRT(s) and input files are generated and some tests are carried out with no further action.
NOAUTO	Take over the input files within the existing \$project/WORK input files and start the calculation. This adjust and manipulate the COLUMBUS input files, such as additional input options, permutation of MOs etc.
ES_GEN	marks the end of the general section.

The following is a list of keywords for the MCSCF section.

<i>Keyword</i>	<i>Meaning</i>
BS_MCSCF	marks the begin of the MCSCF section.
TITLE	The following line is treated as a title for the MCSCF section.
DOCC	The line following this keyword specifies the number of doubly occupied orbitals per irreducible representation. Default is empty DOCC.
RAS	The line following this keyword specifies the number of restricted active orbitals per irreducible representation. Default is empty RAS.
CAS	The line following this keyword specifies the number of complete active orbitals per irreducible representation. Default is empty CAS.
AUX	The line following this keyword specifies the number of auxiliary orbitals per irreducible representation. Default is empty AUX.
NITER	The line following this keyword specifies three numbers indicating the maximum number of MCSCF (macro) iterations, the maximum number of CI iterations and the maximum number of PSCI iterations.
CONV	The line following this keyword specifies the convergence criteria in terms of estimated energy error, orbital rotation norm and CSF rotation norm.
DIAG	The following line specifies the algorithm to solve the CI eigenvalue problem. hmat_full constructs the full CI Hamiltonian and uses a standard eigensolver to solve for the lowest roots. hmat_iter constructs the full CI Hamiltonian and solves iteratively for the lowest eigenvectors. nohmat_iter uses an iterative, direct CI method to find the lowest eigenvectors.
ES_MCSCF	marks the end of the MCSCF section.

The following is a list of keywords for the MRCI section.

<i>Keyword</i>	<i>Meaning</i>
BS_MRCI	marks the begin of the MRCI section
MRCISD	This keyword is used to perform an ordinary Multi-Reference Singles and Doubles CI, MR-SDCI, calculation.

MRACPF	This keyword tells the program to use the Average Coupled Pair Functional, MRACPF.
MRAQCC	This keyword tells the program to use the Average Quadratic Coupled Cluster Functional, MRAQCC.
MRAQCC-V	This keyword tells the program to use the Average Quadratic Coupled Cluster Functional, MRAQCC-v. Note, that the keywords MRCISD, MRACPF, MRAQCC, MRAQCC-V are mutually exclusive. Default is MRCISD.
LRT-MRAQCC	This keyword selects Linear-Response-Theory based MRAQCC. It is assumed that the first DRT entry specifies the reference state (usually the ground state) and subsequent DRTs specify the excited states to be computed.
EXLVL	This keyword defines the maximum excitation level used to create the final configuration space from the reference configuration space. The allowed values 0,1 and 2 are entered on the following line. Default is 2 (single and double excitations).
SSDIM	The two numbers following this line define the maximum and minimum subspace dimension, respectively. The default for the minimum value is the number of roots, the default for the maximum value is the number of roots plus 4. Note, that for serial execution this keyword determines the amount of disk I/O, whereas for parallel execution it determines the memory consumption.
CONV	This keyword defines the requested accuracy in terms of the maximum norm of the residuum vector, i.e. $ \mathbf{r} = \mathbf{H}\mathbf{C} - \mathbf{E}\mathbf{C} $. This corresponds approximately to a convergence to $\Delta E \approx \mathbf{r} ^2$. Default is 0.0001.
NITER	This keyword defines the maximum number of Davidson subspace iterations to be carried out. Default is 10.
REFSPD	This keyword indicates the kind of reference space diagonalization to be used. Valid values are NONE , FULL and ITER . NONE indicates that no reference space diagonalization is carried out. Start vectors are the NROOT unit vectors with the lowest diagonal CI matrix elements. This option is available with MRCISD, only. FULL indicates full reference space diagonalization by a standard eigensolver. Start vectors are the NROOT eigenvectors with the lowest energy. ITER indicates that iterative reference space diagonalization is carried out using the Davidson subspace method for the lowest NROOT roots, which form the start vector set. Default is FULL . Usually reference space diagonalization is beneficial for convergence and pre-requisite for any non-MRCISD calculation. For large reference spaces ($N \geq 500$) the cubic scaling of the (serial) eigensolver may result in prohibitively long startup times. Here the (parallel) iterative scheme is much faster and thus to be preferred. However, the iterative diagonalization may not necessarily provide all low-energy roots for calculations on molecules using a point group of lower order than they actually belong to (e.g. diatomics treated in an abelian point group).
FROZEN	The first line following this keyword specifies the number of frozen core orbitals per irreducible representation in addition to those frozen at the MCSCF level. The second line following this keyword specifies the number frozen virtual orbitals per irreducible representations counting from the highest unoccupied MO downwards.

REFDOCC	The line following this keyword specifies the number of reference doubly occupied orbitals per irreducible representation. Default is empty REFDOCC.
REFRAS	The line following this keyword specifies the number of reference restricted active orbitals per irreducible representation. Default is empty REFRAS.
REFCAS	The line following this keyword specifies the number of reference complete active orbitals per irreducible representation. Default is empty REFCAS.
REFAUX	The line following this keyword specifies the number of reference auxiliary orbitals per irreducible representation. Default is empty REFAUX.
GENSPACE	This keywords indicates to apply generalized interacting space restrictions. Only those configurations are included into the final configuration space which have a non-vanishing matrix element with at least one of the reference configurations. Default is no generalized interacting space restrictions.
FINALW	If this keyword appears the final W vector as also written to disk. This essentially saves one Davidson iteration on restart.
PARALLEL	Execute MRCI code in parallel with NCPU processes (overrides any MOLCAS settings)
TITLE	Title for the MRCI step
ES_MRCI	marks the end of the MRCI section

The following is a list of keywords for the GRAD section.

<i>Keyword</i>	<i>Meaning</i>
BS_GRAD	marks the begin of the gradient section
ROOT	The next line indicates the root to enter the structure optimization.
TITLE	Title of the Gradient part
ES_GRAD	marks the end of the gradient section

2.8.4 Commented Input Examples

Example 1 (serial)

```

*$Revision: 1.2 $
*-----
* Molecule: CH2
* Basis: cc-pvtz
* Symmetry: C2v
* Program Flow: Seward-Scf-Rasscf-columbus(mrci)  Single Point
*-----
  &SEWARD  &END
symmetry
x y
basis set
C.cc-pVTZ.Dunning.10s5p2d1f.4s3p2d1f.
C 0.000000 0.000000 -0.190085345
end of basis
basis set
H.cc-pVTZ.Dunning.5s2p1d.3s2p1d.
H 0.00000000 1.645045225 1.132564974
end of basis
end of input
*-----
  &SCF  &END
occupied
3 0 1 0
end of input
*-----
  &RASSCF  &END
inactive
1 0 0 0
ras2
3 1 2 0
nactel
6 0 0
lumorb
Thrs
1.0E-9 1.0E-6 1.0E-6
Iter
70,25
OUTORBITALS
CANONICAL
end of input
  &COLUMBUS  &END
BS_GEN
TITLE
METHYLEN TEST CASE
ES_GEN
BS_MRCI
DRT
SPIN
1
SYMM
1
NROOT
1
ELECTRONS
8
END_DRT
SSDIM

```



```
5
NITER
 40 1
CONV
 0.001
REFSPD
FULL
MRCISD
GENSPACE
EXLVL
 2
REFDOCC
 1 0 0 0
REFRAS
 0 0 0 0
REFCAS
 3 1 2
ES_MRCI
end of input
```

Example 2 (serial)

```

*$Revision: 1.2 $
*-----
* Molecule: CH2
* Basis: cc-pvtz
* Symmetry: C2v
* Program Flow: Seward-Scf-Rasscf-columbus(mraqcc)  Single Point
*-----
  &SEWARD  &END
symmetry
x y
basis set
C.cc-pVTZ.Dunning.10s5p2d1f.4s3p2d1f.
C 0.000000 0.000000 -0.190085345
end of basis
basis set
H.cc-pVTZ.Dunning.5s2p1d.3s2p1d.
H 0.00000000 1.645045225 1.132564974
end of basis
end of input
*-----
  &SCF  &END
occupied
3 0 1 0
end of input
*-----
  &RASSCF  &END
inactive
1 0 0 0
ras2
3 1 2 0
nactel
6 0 0
lumorb
Thrs
1.0E-9 1.0E-6 1.0E-6
Iter
70,25
OUTORBITALS
CANONICAL
end of input
  &COLUMBUS &END
BS_GEN
TITLE
METHYLEN TEST CASE
PRINT
1
ES_GEN
BS_MRCI
DRT
SPIN
1
SYMM
1
NROOT
1
ELECTRONS
8
END_DRT
SSDIM

```

```
5
NITER
 40 1
CONV
 0.001
REFSPD
FULL
MRAQCC
GENSPACE
EXLVL
 2
REFDOCC
 1 0 0 0
REFRAS
 0 0 0 0
REFCAS
 3 1 2
ES_MRCI
end of input
```

Example 3 (serial)

```

*-----
* Molecule: Ethylen
* Basis: cc-pvdz
* Symmetry: D2h
* Program Flow: Seward-Scf-columbus(SA-MCSCF - MRCISD)  Single Point
* CAS(4e,6o), state-averaging over 1 $\hat{B}_{1u}(\pi-\pi^*)$  + 2 $\hat{B}_{1u}(\pi-\pi^*)$ , 1  $\hat{A}_g(\pi^2)$ 
* CI frozen 1s, REFCAS(4e,6o), 1-2  $\hat{B}_{1u}(\pi-\pi^*)$  + 1  $\hat{A}_g(\pi^2)$ 
* no genspace
*-----
&SEWARD  &END
symmetry
x y z
basis set
C.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
C1  0.00000000  0.00000000  1.26396910
end of basis
basis set
H.cc-pVDZ.Dunning.4s1p.2s1p.
H1  0.00000000  1.75374650  2.32607580
end of basis
end of input
*-----
&SCF  &END
end of input
*-----
* Note, we start the Columbus MCSCF again from
* the SCF orbitals (just for comparison)
* Note, subsequent CI calculation depends on the
* MO resolution, which is not identical in Molcas and Columbus
* (cf. test 11)
>>>COPY $Project.ScfOrb mocoef_mc.lumorb
&COLUMBUS &END
BS_GEN
TITLE
Ethylen
ES_GEN
BS_MCSCF
DRT
SPIN
1
SYMM
5
NROOT
2
ELECTRONS
16 0 0
END_DRT
DRT
SPIN
1
SYMM
1
NROOT
1
ELECTRONS
16 0 0
END_DRT
DOCC
3 0 1 0 2 0 0 0

```

```
CAS
  0 2 1 0 0 2 1 0
NITER
  30 100 100
ES_MCSCF
BS_MRCI
DRT
SPIN
  1
SYMM
  1
NROOT
  1
ELECTRONS
  16
END_DRT
DRT
SPIN
  1
SYMM
  5
NROOT
  2
ELECTRONS
  16
END_DRT
SSDIM
  5
NITER
  40 1
CONV
  0.001
REFSPD
  FULL
MRCISD
GENSPACE
EXLVL
  2
FROZEN
  1 0 0 0 1 0 0 0
  0 0 0 0 0 0 0 0
REFDOCC
  2 0 1 0 1 0 0 0
REFCAS
  0 2 1 0 0 2 1 0
ES_MRCI
end of input
```

Example 4 (parallel)

```

*-----
* Molecule: FURAN
* Basis: cc-pvdz
* Symmetry: C2v
* Program Flow: Seward-Scf-Rasscf-columbus(mcscf-mrci-cigrd-alaska)-slapaf
*               structure optimization (ground state)
*-----
>>> Do while <<<<
  &SEWARD  &END
symmetry
x y
basis set
C.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
C1 2.07569713438866      0.000000000000000      -1.09096195658541
C2 1.36025367031838      0.000000000000000      1.40829741933368
end of basis
basis set
O.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
O1 0.000000000000000      0.000000000000000      -2.62222340632217
end of basis
basis set
H.cc-pVDZ.Dunning.4s1p.2s1p.
H1 2.61859940891225      0.000000000000000      3.05678038225054
H2 3.90513434048509      0.000000000000000      -2.06300002903017
end of basis
end of input
*-----
>>> IF ( ITER = 1 ) THEN <<<
  &SCF  &END
occupied
9 6 2 1
end of input
*-----
  &RASSCF  &END
inactive
9 6 0 0
ras2
0 0 3 2
nactel
6 0 0
lumorb
Thrs
1.0E-9 1.0E-6 1.0E-6
Iter
70,25
OUTORBITALS
CANONICAL
end of input
>>> ENDIF <<<
  &COLUMBUS  &END
BS_GEN
TITLE
  FURAN pi space CAS
NCPU
4
MEMORY
800 MB
PRINT
1

```

```

ES_GEN
BS_MRCI
DRT
SPIN
1
SYMM
1
NROOT
1
ELECTRONS
36
END_DRT
SSDIM
5
NITER
40 1
CONV
0.001
REFSPD
FULL
MRCISD
GENSPACE
EXLVL
2
FROZEN
3 2 0 0
0 0 0 0
REFDOCC
6 4 0 0
REFRAS
0 0 0 0
REFCAS
0 0 3 2
PARALLEL
ES_MRCI
BS_MCSCF
DRT
SPIN
1
SYMM
1
NROOT
1
ELECTRONS
36 0 0
END_DRT
DOCC
9 6 0 0
CAS
0 0 3 2
NITER
30 100 100
ES_MCSCF
BS_GRAD
ROOT
1
PAR_CIGRD
ES_GRAD
end of input
&SLAPAF &END
ITERATIONS
7

```

```
End of Input  
>>> EndDo <<<
```


Example 5 (parallel)

```

*-----
* Molecule: FURAN
* Basis: cc-pvdz
* Symmetry: C2v
* Program Flow: Seward-Scf-Rasscf-columbus(sa-mcscf-mrci-cigrd-alaska)-slapaf
*               state-averaged MCSCF (1  $\hat{A}_1$  + 1  $\hat{3}B_1$ )
*               excited state structure optimization for 1  $\hat{3}B_1$ 
*-----
>>> Do while <<<<
&SEWARD &END
symmetry
x y
basis set
C.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
C1 2.07569713438866      0.000000000000000      -1.09096195658541
C2 1.36025367031838      0.000000000000000      1.40829741933368
end of basis
basis set
O.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
O1 0.000000000000000      0.000000000000000      -2.62222340632217
end of basis
basis set
H.cc-pVDZ.Dunning.4s1p.2s1p.
H1 2.61859940891225      0.000000000000000      3.05678038225054
H2 3.90513434048509      0.000000000000000      -2.06300002903017
end of basis
end of input
*-----
&SCF &END
occupied
9 6 2 1
end of input
*-----
&RASSCF &END
inactive
9 6 0 0
ras2
0 0 3 2
nactel
6 0 0
lumorb
Thrs
1.0E-9 1.0E-6 1.0E-6
Iter
70,25
OUTORBITALS
CANONICAL
end of input
&COLUMBUS &END
BS_GEN
TITLE
FURAN pi space CAS, optimize 3B1
NCPU
4
MEMORY
1000 MB
PRINT
1
ES_GEN

```

```
BS_MRCI
DRT
SPIN
  3
SYMM
  2
NROOT
  1
ELECTRONS
  36
END_DRT
SSDIM
  5
NITER
  40  1
CONV
  0.001
REFSPD
  FULL
MRCISD
GENSPACE
EXLVL
  2
FROZEN
  3 2 0 0
  0 0 0 0
REFDOCC
  6 4 0 0
REFRAS
  0 0 0 0
REFCAS
  0 0 3 2
PARALLEL
ES_MRCI
BS_MCSCF
TITLE
  SA-MCSCF 1A1+3B1
DRT
SPIN
  1
SYMM
  1
NROOT
  1
ELECTRONS
  36 0 0
END_DRT
DRT
SPIN
  3
SYMM
  2
NROOT
  1
ELECTRONS
  36 0 0
END_DRT
DOCC
  9 6 0 0
CAS
  0 0 3 2
NITER
```

```
    30 100 100
ES_MCSCF
BS_GRAD
ROOT
  1
ES_GRAD
end of input
&SLAPAF &END
ITERATIONS
  7
End of Input
>>> EndDo <<<
```


Section 3

Installation Guide

Part IV

Installation Guide

3.1 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 52).

3.2 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 -xvzf 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 -serial`
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 (serial) 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 verification 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.

3.3 Execution

Make sure, that both the startup script for molcas (`molcas`) as well as the COLUMBUS subdirectory of the colmold-isttribution 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.

3.4 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`.

3.5 Requirements/Incompatibilities

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.

3.6 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**.

3.6.1 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 referring 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.

3.7 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.

3.8 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 committed 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 serial 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 serial 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

January 5, 2017

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

Part V

Advanced Examples and Annexes

3.9 Introduction to DRTs in COLUMBUS

add non-relativistic DRT add SO-DRT, even and odd electron case

Bibliography

- [1] B. O. Roos. The complete active space scf method in a fock-matrix-based super-ci formulation. *Int. J. Quant. Chem.*, S14:175, 1980.
- [2] H. Lischka, R. Shepard, R. M. Pitzer, I. Shavitt, M. Dallos, T. Müller, P. G. Szalay, M. Seth, G. S. Kedziora, S. Yabushita, and Z. Zhang. *Phys. Chem. Chem. Phys.*, 3:664, 2001.
- [3] R. J. Gdanitz and R. Ahlrichs. The averaged coupled-pair functional (acpf): a size-extensive modification of mr ci(sd). *Chem. Phys. Letters*, 143:413, 1988.
- [4] P. G. Szalay and R. J. Bartlett. *Chem. Phys. Letters*, 214:481, 1993.
- [5] P. G. Szalay. In R. J. Bartlett, editor, *Modern Ideas in Coupled Cluster Methods*. World Scientific, Singapore, 1997.
- [6] B. O. Roos. A new method for large-scale CI calculations. *Chem. Phys. Letters*, 15:153, 1972.
- [7] I. Shavitt. *Int. J. Quantum Chem.*, S11:131, 1977.
- [8] R. Shepard, G. S. Kedziora, H. Lischka, I. Shavitt, Th. Müller, P. G. Szalay, M. Kallay, and M. Seth. The accuracy of molecular bond lengths computed by multireference electronic structure methods. *Chem. Phys.*, 349:37, 2008.
- [9] Th. Müller, M. Dallos, H. Lischka, Z. Dubrovay, and P. G. Szalay. *Theor. Chem. Acc.*, 105:227, 2001.
- [10] P. G. Szalay, Th. Müller, and H. Lischka. Excitation energies and transition moments by the multireference averaged quadratic coupled cluster (mr-aqcc) method. *Phys. Chem. Chem. Phys.*, 2:2067, 2000.