MOLCAS 8: MOLCAS 8: New Capabilities for Multiconfigurational Quantum Chemical Calculations across the Periodic Table.

SUPPORTING INFORMATION

Francesco Aquilante, Jochen Autschbach, Rebecca K. Carlson, Liviu F. Chibotaru, Mickaël G. Delcey, Luca De Vico, Ignacio Fdez. Galván, Nicolas Ferré, Luis Manuel Frutos, Laura Gagliardi, Marco Garavelli, Angelo Giussani, Chad E. Hoyer, Giovanni Li Manni, Hans Lischka, Dongxia Ma, Per-Åke Malmqvist, Thomas Müller, Artur Nenov, Massimo Olivucci, Thomas Bondo Pedersen, Daoling Peng, Felix Plasser, Ben Pritchard, Markus Reiher, Ivan Rivalta, Igor Schapiro, Javier Segarra-Martí, Michael Stenrup, Donald G. Truhlar, Liviu Ungur, Alessio Valentini, Steven Vancoillie, Valera Veryazov, Victor P. Vysotskiy, Oliver Weingart, Felipe Zapata, and Roland Lindh

OUTLINE

S1. COMPUTATIONAL RESOURCES NEEDED BY `rasscf`

The `rasscf` program uses both configuration state functions (CSFs) as well as Slater determinants (SDs), with the latter being used by the `lucia` program to handle the sigma updates ($s = Hc$) and density matrix computation. The `lucia` program is integrated into MOLCAS as `lucia_util` (part of the MOLCAS library).

The memory needed by `lucia_util` is roughly twice the number of determinants, and remains allocated throughout the duration of the `rasscf` run. The memory needed by the `rasscf` program itself amounts to a minimum of 5 determinant vectors. So, in total, the program needs to hold about $7N_{\mathrm{SD}}$ of values in memory at the same time.

The computational cost of the sigma update in terms of operation count is approximately:

$$N_{\mathrm{SD}}(\alpha^2(n_{\mathrm{act}}-\alpha)^2+\beta^2(n_{\mathrm{act}}-\beta)^2+\alpha\beta(n_{\mathrm{act}}-\alpha)(n_{\mathrm{act}}-\beta)$$

For $\alpha = \beta = \frac{n_{\mathrm{act}}}{2}$, this scales as $N_{\mathrm{SD}}n_{\mathrm{act}}^4$. This cost will be multiplied by the total number of CI iterations that are performed during the course of the `rasscf` optimization (i.e. multiple times per RASSCF iteration). The cost for computing the density matrix scales as $N_{\mathrm{SD}}n_{\mathrm{act}}^4$, and is needed only once every RASSCF iteration. Both operations (sigma update and denisty matrix computation) have been parallelized.

S2. COMPUTATIONAL RESOURCES NEEDED BY `caspt2`

The memory needed by the `caspt2` program with respect to active space size depends on two distinct variables: the number of configuration state functions (CSFs, since `caspt2` doesn't use determinants) and the number of active orbitals. We need to store at least 4 times the number of CSFs during the computation of the active density matrices, while the memory needed to store the 3-body active density matrix scales with the sixth power of the number of active orbitals. At a later stage, twice the size of the 3-body density matrix needs to be kept in memory simultaneously. In summary, this leads to a minimum memory requirement of: $\max(4N_{\mathrm{CSF}}, 2n_{\mathrm{act}}^6)$. This means that for large CAS-type wavefunctions, the number of CSFs will be the determining factor for the required memory, while for RAS-type wavefunctions with a large number of active orbitals (i.e. more than 30 for a non-symmetric system) the size of the density matrices will start to play a role in the memory requirement.

A large part of the computational cost of the entire `caspt2` program depends mainly on the computation and diagonalization of the 3-body active density matrix. The computational cost scales as $N_{\text{CSF}} n_{\text{act}}^6$ for the construction of the matrix and n_{act}^9 for the diagonalization. Both of these algorithms are parallelized. With a (future) DMRG interface, construction of the density matrix will be outsourced, and the limit on the number of active orbitals coming from `caspt2` will be determined by the memory requirements for storage of the matrices and the time it takes to perform the diagonalization.

The remaining memory needs scale roughly with the number of basis functions as $n_{\text{occupied}}^2 n_{\text{virtual}}^2$. When running in parallel, one can avoid needing this amount of memory by using Cholesky vectors in `seward` in combination with the on-demand algorithm for the computation of the right-hand side in `caspt2` (keyword RHSD). This lowers the memory requirements for each process to the the total size of Cholesky vectors. The same memory requirement of $n_{\text{occupied}}^2 n_{\text{virtual}}^2$ is also needed during the final PCG phase of `caspt`. This can be reduced to a minimum of around $n_{\text{occupied}}^2 n_{\text{virtual}}$ per process when running in parallel.

S3. GASSCF IN ACTION

A simple test case is described here to show how to prepare GASSCF inputs (Listing S1). The GASSCF method is activated by the `GASScf` keyword within the `rasscf` module. The total number of GAS spaces immediately follow. Then, for each GAS space the active orbitals (per Irrep) as well as the cumulative minimum and maximum number of electrons are given. For the example given here a total of six active electrons are distributed in six valence active orbitals, namely the 2p orbitals involved in the bonding. The 1s and 2s orbitals are kept INACTIVE. In this case the GAS spaces are chosen in a Generalized Valence Bond (GVB) fashion with bonding and anti-bonding orbitals coupled in the same GAS space. The minimum and maximum occupation numbers are set in a way that for each space only two electrons are allowed. We define these subspaces as non-connected GAS spaces as no inter-space excitations are allowed. To notice that at no time a GAS space is completely full (doubly occupied orbitals). This is a feature of GAS that cannot be reproduced by the RAS approach.

```
&RASSCF
 LINEAR
 nActEl
  6 0 0
 FROZen
  0 0 0 0 0 0 0 0
 INACTIVE
  2 0 0 0 2 0 0 0
 GASScf
  3                  ! total number of GAS spaces
  1 0 0 0 1 0 0 0   ! bonding and anti-bonding 2pz orbitals in GAS1
  2 2                ! Two electrons in GAS1
  0 1 0 0 0 1 0 0   ! bonding and anti-bonding 2px orbitals in GAS2
  4 4                ! Two electrons in GAS2
  0 0 1 0 0 0 1 0   ! bonding and anti-bonding 2py orbitals in GAS3
  6 6                ! Two electrons in GAS3
 DELEted
  0 0 0 0 0 0 0 0
 Symmetry
  1
 Spin
  1
```

LISTING S1: A possible GASSCF setup for the nitrogen molecule in D_{2h} point group. Three GAS spaces are chosen. In this case inter-space excitations are not allowed.

S4. MC-PDFT IN ACTION

In Listing S2 an input is given showing how to prepare the input for MC-PDFT calculations. In this example an EMIL instruction is given to loop over the three currently available functionals (TPBE, TBLYP, TLSDA). For each functional a CI calculation (`CIONLY` flag) is run starting from the CI vector as stored in the JOBIPH file (see `JOBIPH` and `CIREstart` flags). The MC-PDFT method is activated by the `KSDFT`, `ROKS` keywords followed by the functiol choice ($DFT variable in this case).

```
>>foreach DFT in (TPBE, TBLYP, TLSDA)
>>> COPY $WorkDir/test.JobIph JOBOLD
 &RASSCF &END
  LINEAR
  CIONLY
  JOBIPH
  CIREstart
  nActEl
   6 0 0
  FROZen
   0 0 0 0 0 0 0 0
  INACTIVE
   2 0 0 0 2 0 0 0
  RAS2
   1 1 1 0 1 1 1 0
  DELEted
   0 0 0 0 0 0 0 0
  KSDFT
  ROKS
  $DFT
  Symmetry
   1
  Spin
   1
  >>enddo
```

LISTING S2: A possible MC-PDFT input for the nitrogen molecule. This input will loop over the three fnctionals currently implemented in Molcas (TPBE, TBLYP, TLSDA).

Root 1	Root 2	Root 3

`&CASPT2`
`MULTISTATE = 3 1 2 3`
`ONLY root = 1`
	`&CASPT2`
	`MULTISTATE = 3 1 2 3`
	`ONLY root = 2`
		`&CASPT2`
		`MULTISTATE = 3 1 2 3`
		`ONLY root = 3`

TABLE S1: The `ONLY root` keyword is used to specify which root to compute, while still obtaining all coupling terms with roots specified by the `MULTISTATE` keyword.

```
Hamiltonian Effective Couplings
------------------------------
                |    1 >
<    1 |   -1183.91762829
<    2 |      -0.00023372
<    3 |      -0.00009067
```

LISTING S3: The output couplings obtained using the `ONLY root 1` keyword.

S5. MS-RASPT2 JOB-FARM

As an example, consider the first three singlet roots of a molecule are to be computed at the MS-RASPT2 level of theory. As described in previous sections, the first step is to compute a three roots, state average RASSCF reference wave function. The so obtained JobIph file contains the multiconfigurational reference wave function, to be passed to the multireference perturbation theory treatment.

The input commands necessary for the MS-RASPT2 calculations are given in Table S1. In this example, each of the single root calculations took 9 hours (compared to 22 hours for a single job with all three roots). In the single root 1 output, the Hamiltonian effective couplings are then printed as reported in Listing S3. Similar outputs are obtained from roots 2 and 3. These coupling terms are needed to prepare the input for the final MS-RASPT2 calculation, using the `EFFEctive Hamiltonian Couplings` keyword, as shown in Listing S4. The final calculation is very fast, as it only involves a multistate diagonalization.

S6. ORBITALS FROM SO-RASSI CALCULATIONS

The orbitals discussed in Section III B can be generated with the keywords `SONORB` and `SODIAG` in `rassi`. Both keywords are followed by the number of electronic states to consider, and a list of state numbers. For example,

```
 &CASPT2
MULTISTATE
3 1 2 3
EFFEctive Hamiltonian Couplings
-1183.91762829       0.00003950     -0.00003218
   -0.00023372  -1183.88855319      0.00098690
   -0.00009067     -0.00882893  -1183.87399725
```

LISTING S4: The `EFFEctive Hamiltonian Couplings` keyword is followed by the coupling terms as printed for the various roots. Each column is the output from one single root calculation, as in Listing S3.

```
SONORB
2
1
2
```

selects the first two SO-RASSI states and generates `SONATTDENS.`n`.`n files specifying the NOs for state n. For the $5f^1$ systems of Fig. 6 this corresponds to the ground state doublet. The corresponding information about the NSOs for the spin magnetization m^u with $u = X, Y, Z$ are stored in files `SONATSDENS.`n`.`n`.`u. NSOs for linear combinations of the state components diagonalizing the Zeeman Hamiltonian for a field along the magnetic axes $v = X_m, Y_m, Z_m$ of Section III C can be generated with

```
SODIAG
2
1
2
```

The NSO data are stored in files `SODI`v`SDENS.`n`.`n`.`u. The magnetic axes are specified in file `SODIAG.MAXES`. The orbital data files have a similar structure as the usual `RasOrb` and `SpdOrb` files generated by the `rasscf` module. The `grid_it` tool can be used to generate volume data to visualize the orbitals and spin magnetizations. Other formats were not tested. The NO occupations and NSO spin populations are listed under `OCCUPATION NUMBERS` near the end of each file.

S7. OPTIMIZATION WITH COMPOSITE GRADIENTS

The following is an example input for a geometry optimization using the “composite gradients” described in Section V B. It is an optimization of the ground state of the molecule shown in Fig. 23:

```
&GATEWAY
```

```
  Coord = 3Me-acrolein.xyz
  Basis = ANO-RCC-VDZP
  Group = NoSymm
  RICD
  Constraints
    Me1 = Fragment C1 H8  H9  H10
    Me2 = Fragment C3 H11 H12 H13
    Me3 = Fragment C5 H14 H15 H16
  Values
    Me1 = Fix phantom
    Me2 = Fix phantom
    Me3 = Fix phantom
  End of constraints

>>> DoWhile

&SEWARD

&SCF
&MBPT2
&ALASKA

&RASSCF
  FileOrb = $CurrDir/init.RasOrb
  Charge  = 0
  Ras2    = 5
  NActEl  = 6 0 0
  CIRoot  = 4 4 1
  RlxRoot = 1
&CASPT2
  Multistate = 1 1
  NoMultiState
  Imaginary shift = 0.1
&ALASKA
  KeepOldGradient

&SLAPAF
  Cartesian
  Thrsholds = 5.0D-6 3.0D-4

>>> EndDo
```

The initial geometry is provided in the file `3Me-acrolein.xyz`, and the initial orbitals in `init.RasOrb`, both in the input directory.

The methyl groups are defined with the `Fragment` constraint type. With the `phantom` qualifier, the constraints are only used for numerical differentiation, and the optimization is performed without actual constraints.

To compute the composite gradient, first an MP2 calculation is done (`mbpt2` module), and its analytical gradient is computed. Then comes a CASPT2 calculation and its gradient.

Since Molcas does not have an analytical implementation for CASPT2 gradients, numerical differentiation will be performed, and the “phantom” constraints are applied, only the non-methyl degrees of freedom are differentiated. The `KeepOldGradient` keyword keeps the previously computed MP2 gradient for the methyl degrees of freedom. Finally `slapaf` computes a new geometry using the composite gradient.

S8. THE MOLCAS/COLUMBUS LINK

S8.1. Computational Details

The high-symmetry structure of C_{6v} symmetry was chosen for the adsorption complex where the Cu atom is moved toward the center of the benzene ring. Since only Abelian groups can be treated in Columbus, the operational symmetry is reduced to C_{2v} and all symmetry notations refer to it. The benzene molecule is located in the xy plane with the center of gravity in the origin and the x-axis being perpendicular to one CC bond. The y axis passes through two opposite carbon atoms. The Cu atom is located along the z-axis.

The SA-MCSCF calculations preceding the MR-CISD computation were based on a direct product of two complete active spaces(CAS): CAS(11e,6o)$\otimes$CAS(6e,6o). The first space containing 11 electrons and 6 orbitals refers to the Cu atom and includes the 3d and 4s subshells. The second space relates to the benzene molecule (Bz). The state averaging was performed over the states Cu(^2S)$\otimes$Bz($^1A_1$), Cu(^2D)$\otimes$Bz($^1A_1$), Cu(^2D)$\otimes$Bz($^3B_2$). The Cu(^2S)$\otimes$Bz($^3B_2$) states were excluded from state-averaging since the Cu(^2S) is located too high in energy at the MCSCF level. To improve the description of the Cu(^2S) in this state averaging it was given a weight of five whereas for the other states a weight of one was used. Based on the molecular orbitals (MOs) obtained from these SA-MCSCF calculations a reference space of Cu(CAS(11e,6o)$\otimes$CAS(4e,4o) was used at MR-CISD level moving the lowest and highest π orbital of benzene from the active to the doubly occupied space and virtual space, respectively.

The basis set consisted of a cc-pVDZ basis[1] for C and H and of the small core relativistic effective core potential (RECP) and the corresponding basis set (8s7p6d1f) contracted to [4s4p3d1f] as developed by Peterson.[2]

S8.2. Mixed operation of Molcas and Columbus

The interfaces exploit the modular nature of both Columbus and Molcas packages and are based upon the exchange of *well-defined quantities* (integral, 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 wave functions, which cannot necessarily exactly and easily be interconverted between Molcas and Columbus representations.

The Cholesky decomposition scheme is currently not supported when using the Molcas-Columbus interface.

S8.3. 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 (see Fig. S1) 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

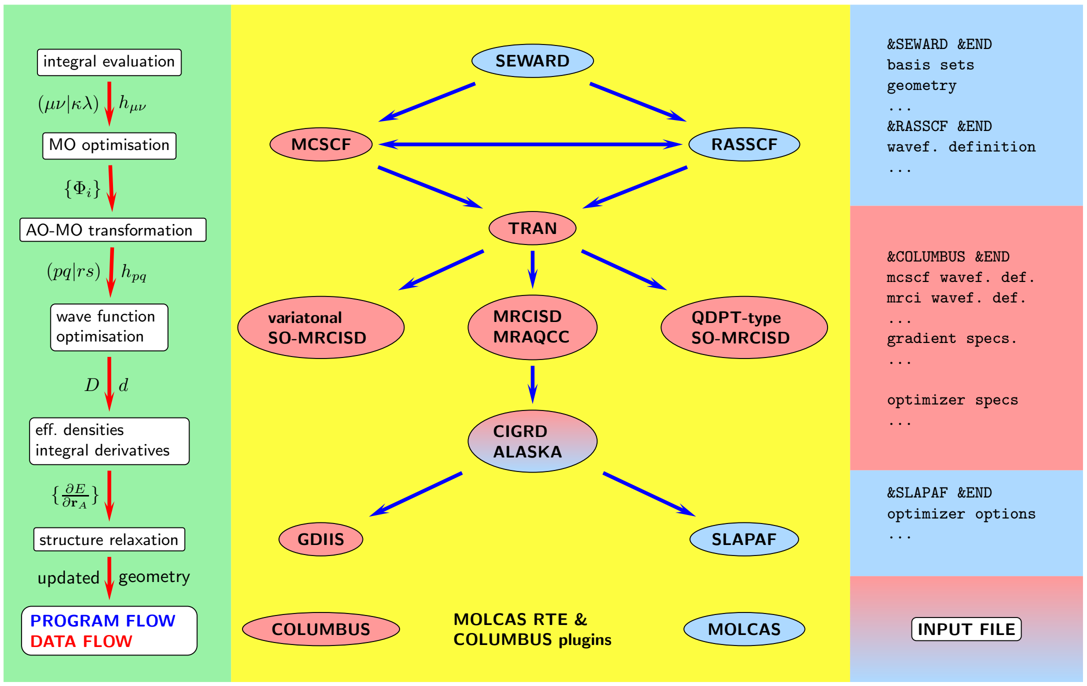


FIG. S1: Schematically depicted data and program flow for mixed operation of COLUMBUS and MOLCAS under control of the MOLCAS-driver. The depicted work flow refers to single-point energy evaluations and structure optimizations. MOLCAS-input files in principle also allow for much more complex work flows.

control over the COLUMBUS modules while avoiding a rather cumbersome MOLCAS input file, it is recommended to first prepare a closely related MOLCAS style input file, execute it with the INPUTONLY option, modify these input files, replace the line INPUTONLY by NOAUTO which suppresses the input file generation all together and rerun the calculation. It is primarily useful for specialized options to some COLUMBUS modules and for the construction of more specialized CSF spaces.

Due to various technical details, it is not always straightforward to compile MOLCAS and COLUMBUS in a consistent way — especially for satisfactory parallel operation on HPC systems, while ordinary PC clusters are frequently inadequate for parallel operation because of insufficient network bandwidth. Hence, this particular COLUMBUS version is distributed as a set of binaries, supporting serial and SMP-level parallelism out of the box. After downloading and unpacking the binaries, all that is necessary is to add the file `columbus.prgm` to the `MOLCAS/data` directory and to define the environment variable `COLUMBUS` pointing to the COLUMBUS installation directory. After running the test suite, it is ensured that your installed MOLCAS version is binary compatible (for details refer to COLUMBUS manual).

The following two examples show the implementation and use of the COLUMBUS/MOLCAS link under the control of the Molcas driver.

Example 1:

Methylene singlet, RASSCF followed by parallel MR-CISD calculation on 2 Cores

```
*-------------------------------------------------------------------------------
* CH2 / cc-pvtz / C2v
*-------------------------------------------------------------------------------
 &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
end of input

 &COLUMBUS &END
BEGIN_SECTION_GENERAL
TITLE
 METHYLEN TEST CASE
NCPU=2
END_SECTION_GENERAL
BEGIN_SECTION_MRCI
DRT
MULTIPLICITY
 1
SYMMETRY
 1
ELECTRONS
 8
NROOT
 1
END_DRT
```

```
REFDOCC
 1 0 0 0
REFRAS
 0 0 0 0
REFCAS
 3 1 2
SUBSPACEDIM
 5
ITERATIONS
 40
ACCURACY
 0.001
REFSPACE
 FULL
MRCISD
GENSPACE
EXLVL
 2
PARALLEL
END_SECTION_MRCI
end of input
```

Example 2

Excited triplet state structure optimization at parallel MRCISD level of theory, SA-MCSCF over $^1A_1$ and $^3B_1$.

```
>>> Do while <<<
 &SEWARD   &END
symmetry
x y
basis set
C.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
C1 2.07569713438866      0.00000000000000     -1.09096195658541
C2 1.36025367031838      0.00000000000000      1.40829741933368
end of basis
basis set
O.cc-pVDZ.Dunning.9s4p1d.3s2p1d.
O1 0.00000000000000      0.00000000000000     -2.62222340632217
end of basis
basis set
H.cc-pVDZ.Dunning.4s1p.2s1p.
H1  2.61859940891225      0.00000000000000      3.05678038225054
H2  3.90513434048509      0.00000000000000     -2.06300002903017
end of basis
end of input
*-------------------------------------------------------------------------------
 &SCF  &END
occupied
9 6 2 1
end of input
*-------------------------------------------------------------------------------
 &RASSCF  &END
```

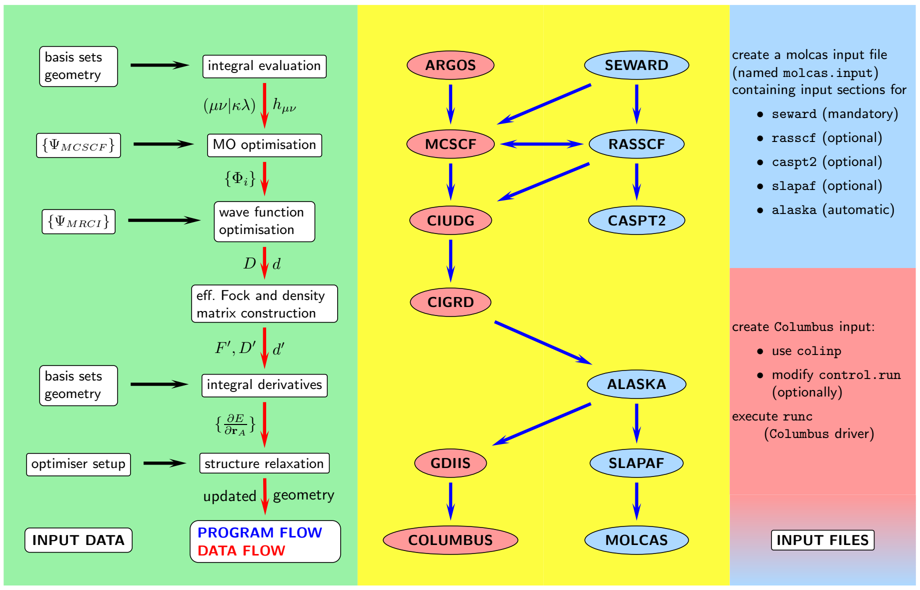
```
inactive
9 6 0 0
ras2
0 0 3 2
nactel
6 0 0
lumorb
Thrs
1.0E-9 1.0E-6 1.0E-6
Iter
70,25
OUTORBITALS
 CANONICAL
end of input

 &COLUMBUS &END
BEGIN_SECTION_GENERAL
TITLE
 FURAN pi space CAS, optimize 3B1
NCPU
 8
MEMORY
 1000 MB
PRINT
 1
END_SECTION_GENERAL
BEGIN_SECTION_MRCI
DRT
MULTIPLICITY
 3
SYMMETRY
 2
NROOT
 1
ELECTRONS
 36
END_DRT
SUBSPACEDIM
 5
ITERATIONS
 40  1
ACCURACY
 0.001
REFSPACE
 FULL
MRCISD
GENSPACE
EXLVL
 2
FROZENCORE
 3 2 0 0
REFDOCC
 6 4 0 0
```

```
REFRAS
 0 0 0 0
REFCAS
 0 0 3 2
PARALLEL
END_SECTION_MRCI
BEGIN_SECTION_MCSCF
TITLE
 SA-MCSCF 1A1+3B1
DRT
MULTIPLICITY
 1
SYMMETRY
 1
NROOT
 1
ELECTRONS
 36 0 0
END_DRT
DRT
MULTIPLICITY
 3
SYMMETRY
 2
NROOT
 1
ELECTRONS
 36 0 0
END_DRT
DOCC
 9 6 0 0
CAS
 0 0 3 2
ITERATIONS
 30 100 100
END_SECTION_MCSCF

BEGIN_SECTION_GRAD
ROOT
 1
END_SECTION_GRAD
end of input

 &SLAPAF &END
 ITERATIONS
  7
 End of Input
>>> EndDo <<<
```



create a molcas input file (named `molcas.input`) containing input sections for

- `seward` (mandatory)
- `rasscf` (optional)
- `caspt2` (optional)
- `slapaf` (optional)
- `alaska` (automatic)

create `Columbus` input:

- use `colinp`
- modify `control.run` (optionally)

execute `runc` (Columbus driver)

FIG. S2: 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.

S8.4. 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 (see Fig. S2). 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.

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

For details, refer to the COLUMBUS manual and COLUMBUS tutorial, respectively.

Representative example input and results files for different example cases can be found in the `$COLUMBUS/EXAMPLES/MOLCAS` directory of the 7.0 version of COLUMBUS. These include

- HF-MCSCF-CI
  a simple COLUMBUS MR-CI job with MOLCAS integrals
- BENZENE_gradient
  Computation of MR-CI gradients with MOLCAS integrals
- HF-RASSCF-CI_AQCC_ACPF
  combination of the MOLCAS `rasscf` program with the COLUMBUS CI program
- Acetylene-RASSCF-AQCC-CASPT2
  orbital generation with the MOLCAS `rasscf` module followed by MOLCAS `caspt2` and COLUMBUS MR-AQCC
- p-benzyne-geomopt
  structure optimization with MOLCAS integrals.

REFERENCES

[1]T. H. Dunning, Jr., *J. Chem. Phys.* **1989**, *90*, 1007.

[2]K. A. Peterson, C. Puzzarini, *Theor. Chem. Acc.* **2005**, *114*, 283–296.