Forte Documentation

Release 0.1.1

Evangelista lab

Sep 13, 2018

Contents:

1 Capabilities **3**

2 Dependencies **5**

3 Running Forte **7**
- 3.1 Obtaining Forte . . . 7
- 3.2 Compiling Forte . . . 7
- 3.3 Setting up the PYTHONPATH . . . 8
- 3.4 Frequently asked questions . . . 8
  - 3.4.1 "ImportError: dynamic module does not define init function (initforte)" . . . 8

4 Getting started **9**
- 4.1 Running the test cases . . . 9

5 Specifying a wave function in Forte **11**
- 5.1 Number of electrons and spin . . . 11
- 5.2 Definition of orbital spaces . . . 12
- 5.3 Orbital space specification . . . 12
- 5.4 Partial specification of orbital spaces and space priority . . . 12

6 Two-electron Integral Types **15**
- 6.1 Conventional integrals . . . 15
- 6.2 Density Fitting (DF) and Cholesky Decomposition (CD) . . . 16
- 6.3 Disk-based Density Fitting (DiskDF) . . . 16
- 6.4 Integral Selection Keywords . . . 16

7 Full configuration interaction **19**
- 7.1 Running the test cases . . . 19

8 Driven Similarity Renormalization Group **21**
- 8.1 Basic DSRG . . . 21
  - 8.1.1 DSRG Options . . . 21
- 8.2 MR-LDSRG(2) . . . 21
  - 8.2.1 MR-LDSRG(2) Options . . . 21

9 Programmer's Manual **23**
- 9.1 Psi4 . . . 23
  - 9.1.1 Symmetry and the Dimension class . . . 23

9.1.2 The `Vector` and `Matrix` classes 23

10 List of Forte options **25**

11 Indices and tables **47**

Forte is an open-source suite of quantum chemistry methods for strongly correlated electrons.

Forte is an open-source suite of state-of-the-art quantum chemistry methods applicable to chemical systems with strongly correlated electrons. The code is written as a plugin to Psi4 in C++ with C++11 functionality, and it takes advantage of shared memory parallelism throughout.

CHAPTER 1

Capabilities

In general, Forte is composed of two types of methods:

1. **Active space methods**
    (a) Full/complete active space configuration interaction (FCI)/(CASCI)
    (b) Adaptive configuration interaction (ACI)
    (c) Projector configuration interaction (PCI)
    (d) Complete active space self-consistent field (CASSCF)
    (e) Density Matrix Renormalization Group self-consistent field (DMRG-SCF)
2. **Methods for dynamical correlation**
    (a) **Driven similarity renormalization group (DSRG)**
        i. DSRG-MRPT2
        ii. DSRG-MRPT3
        iii. MR-LDSRG(2)
    (b) Uncontracted multireference configuraiton interaction (MRCI)

Note that the active space methods, notably FCI, ACI, and PCI, can operate within the full orbital basis defined by the user-selected basis set. In this case, these methods also recover dynamical correlation.

CHAPTER 2

Dependencies

In order to run Forte, the following are required:

1. A Recent version of Psi4
2. CMake version 3.0 or higher
3. The tensor library Ambit

CHAPTER 3

Running Forte

3.1 Obtaining Forte

You can download the source code of Forte from GitHub.

To clone the latest version of the repository run:

```
git clone https://github.com/evangelistalab/forte.git forte
```

3.2 Compiling Forte

Once you have the current versions of Psi4, CMake, and Ambit, follow these instructions to install Forte:

1. Run psi4 in the Forte folder:

   ```
   psi4 --plugin-compile
   ```

Psi4 will generate a CMake command for building Forte that looks like:

```
cmake -C /usr/local/psi4/stage/usr/local/psi4/share/cmake/psi4/psi4PluginCache.cmake
    -DCMAKE_PREFIX_PATH=/usr/local/psi4/stage/usr/local/psi4
```

2. Run the cmake command generated in 1. appending the location of Ambit's cmake files (via the `-Dambit_DIR` option):

   ```
   cmake -C /usr/local/psi4/stage/usr/local/psi4/share/cmake/psi4/psi4PluginCache.
   ↪cmake
       -DCMAKE_PREFIX_PATH=/usr/local/psi4/stage/usr/local/psi4 .
       -Dambit_DIR=<ambit-bin-dir>/share/cmake/ambit
   ```

3. Run make:

```
make
```

3.3 Setting up the PYTHONPATH

If Forte in installed in the folder /<path>/forte, then PYTHONPATH should contain /path/. Note that if you include /<path>/forte in PYTHONPATH you will get an error (see Frequently asked questions).

3.4 Frequently asked questions

3.4.1 "ImportError: dynamic module does not define init function (initforte)"

Make sure that your PYTHONPATH does not include the Forte directory. That is, if Forte is in /<path>/forte then PYTHONPATH should contain /path/ and not /<path>/forte.

CHAPTER 4

Getting started

4.1 Running the test cases

Forte provides test cases for most of all methods implemented. This is a good place to start if you are new to Forte. After compiling and setting up `PYTHONPATH`, you can run the test cases:

```
cd tests/methods
python run_forte_tests_travis.py
```

CHAPTER 5

Specifying a wave function in Forte

5.1 Number of electrons and spin

By default, Forte will determine the number of electrons from the atomic charges and total molecular charge. The total molecular charge is assumed to be 0 unless specified by the user in the input file:

```
molecule {
  0 1 # <-- charge and multiplicity (see below)
  ...
}
```

For certain computations, Forte allows the user to compute a solution with a well defined value of total spin ($\hat{S}^2$) and spin projection onto the z axis ($\hat{S}_z$).

The total spin is controlled by the option `MULTIPLICITY`. This quantity is related to the total spin quantum number `S` by the condition `MULTIPLICITY = 2S + 1`. If the input file does not specify the option `MULTIPLICITY`, Forte will read the multiplicity from the `Wavefunction` object passed by Psi4.

The projection of spin onto the z axis is controlled by the option `MS`. This option is of type `double`, so it should be specified as `0.0`, `-1.5`, etc. If the user does not specify the option `MS`, Forte deduces a value consistent with the option `MULTIPLICITY`. Modules will select either the lowest or highest value of `MS` compatible with `MULTIPLICITY`, depending on internal details of the implementation. For example, if `MULTIPLICITY = 3` and `MS` is not specified, the FCI code in Forte will assume that the user is interested in the solution with $M_S = 0$.

For example, the following input requests the $M_S = 0$ component of a triplet state:

```
# triplet, m_s = 0
set forte{
    multiplicity = 3
    ms = 0.0
}
```

while the following gives the $M_S = -1$ component:

```
# triplet, m_s = 1
set forte{
    multiplicity = 3
    ms = -1.0
}
```

5.2 Definition of orbital spaces

Running a Forte computation requires specifying a partitioning of the molecular orbitals. Forte defines five orbital spaces:

1. Frozen doubly occupied orbitals (`FROZEN_DOCC`). These orbitals are always doubly occupied.

2. Restricted doubly occupied orbitals (`RESTRICTED_DOCC`). Orbitals that are treated as doubly occupied by method for static correlation. Restricted doubly occupied orbitals are allowed to be excited in in methods that add dynamic electron correlation.

3. Active orbitals (`ACTIVE`). Used to define active spaces for static correlation methods. These orbitals are partially occupied.

4. Restricted unoccupied orbitals (`RESTRICTED_UOCC`). Also called virtuals, these orbitals are ignored by methods for static correlation but considered by dynamic correlation approaches.

5. Frozen unoccupied orbitals (`FROZEN_UOCC`). These orbitals are always unoccupied.

5.3 Orbital space specification

Selecting the correct set of orbitals for a multireference computation is perhaps one of the most important step in setting up an input file.

Forte takes advantage of symmetry, so for each orbital space the user must provide the number of orbitals in each irrep. Forte cna handle only Abelian groups, so each orbital space is a vector of integers with at most eight entries. Irreps are arranged according to Cotton's book (*Chemical Applications of Group Theory*).

The following is an example of a computation on BeH_2. This system has 6 electrons. We freeze the Be 1s-like orbital, which has A_1 symmetry. The $2a_1$ orbital is restricted doubly occupied and the $3a_1/1b_2$ orbitals belong to the active space. The remaining orbitals belong to the `RESTRICTED_UOCC` set and no virtual orbitals are frozen:

```
set forte{
    #                 A1 A2 B1 B2
    frozen_docc      [1 ,0 ,0 ,0]
    restricted_docc  [2 ,0 ,0 ,0]
    active           [1 ,0 ,0 ,1]
    restricted_uocc  [4 ,0 ,2 ,3]
    frozen_uocc      [0 ,0 ,0 ,0]
}
```

5.4 Partial specification of orbital spaces and space priority

Specifying all five orbital spaces for each computation is tedious and error prone. Forte can help reduced the number of orbital spaces that the user needs to specify by making certain assumptions. The class that controls orbital spaces (`MOSpaceInfo`) assumes that orbital spaces have the following priority:

```
ACTIVE > RESTRICTED_UOCC > RESTRICTED_DOCC > FROZEN_DOCC > FROZEN_UOCC
```

When the input does not contain all five orbital spaces, Forte will infer the size of other orbital spaces. It first sums up all the orbitals specified by the user, and then assigns any remaining orbital to the space not specified in the input that has the highest priority.

In the case of the BeH_2 example, it is necessary to specify only the `FROZEN_DOCC`, `RESTRICTED_DOCC`, and `ACTIVE` orbital spaces:

```
set forte{
    frozen_docc      [1 ,0 ,0 ,0]
    restricted_docc  [2 ,0 ,0 ,0]
    active           [1 ,0 ,0 ,1]

    # Forte will automatically assign the following:
    # restricted_uocc  [4 ,0 ,2 ,3]
    # frozen_uocc      [0 ,0 ,0 ,0]
}
```

the remaining 9 orbitals are automatically assigned to the `RESTRICTED_UOCC` space. This space, together with `FROZEN_UOCC`, was not specified in the input. However, `RESTRICTED_UOCC` has higher priority than the `FROZEN_UOCC` space, so Forte will assign all the remaining orbitals to the `RESTRICTED_UOCC` set.

A Forte input with no orbital space specified will assign all orbitals to the active space:

```
set forte{
    # Forte will automatically assign the following:
    # frozen_docc      [0 ,0 ,0 ,0]
    # restricted_docc  [0 ,0 ,0 ,0]
    # active           [7 ,0 ,2 ,4]
    # restricted_uocc  [0 ,0 ,0 ,0]
    # frozen_uocc      [0 ,0 ,0 ,0]
}
```

Note, that except for full CI computations with small basis sets, in all other cases this computation might be unfeasible.

As a general rule, it is recommended that user run a SCF computations and inspect the orbitals prior to selecting an active space.

CHAPTER 6

Two-electron Integral Types

Forte can handle different types of exact and approximate two-electron integrals. This section describes the various options available and their properties/limitations. The selection of different integral types is controlled by the option `INT_TYPE`

6.1 Conventional integrals

Conventional integrals are the default choice for Forte. When this option is selected, Forte will compute and store the two-electron integrals in the molecular orbital (MO) basis ϕ_p.

$$\langle pq|rs\rangle = \int dx_1 dx_2 \phi_p(x_1)^* \phi_q(x_2)^* r_{12}^{-1} \phi_r(x_1) \phi_s(x_2)$$

These integrals are computed with Psi4's `IntegralTrasform` class and written to disk. Forte will store three copies of these integrals, the antisymmetrized alpha-alpha and beta-beta integrals

$$\langle p_\alpha q_\alpha \| r_\alpha s_\alpha \rangle, \langle p_\beta q_\beta \| r_\beta s_\beta \rangle,$$

and the alpha-beta integrals (not antisymmetrized)

$$\langle p_\alpha q_\beta | r_\alpha s_\beta \rangle,$$

for all values of p, q, r, s. Storage of these integrals has a memory cost equal to $3N^4$, where N is the number of orbitals that are correlated (frozen core and virtual orbitals excluded). Therefore, conventional integrals are viable for computations with at most 100-200 orbitals. For larger bases, density Fitting and Cholesky decomposition are instead recommended.

6.2 Density Fitting (DF) and Cholesky Decomposition (CD)

The density fitting and Cholesky decomposition methods approximate two-electron integrals as products of three-index tensors b_{pr}^{P}

$$\langle pq|rs\rangle = \sum_{P}^{M} b_{pr}^{P} b_{qs}^{P}$$

where M is a quantity of the order $3N$.

Caution: The equations reported here use physicist notation for the two-electron integrals, but the DF/CD literature usually adopts chemist's notation.

The main difference between DF and CD is in the way the B tensors are defined. In DF, the b tensor is defined as

$$b_{pq}^{Q} = \sum_{p}(pq|P)[(P|Q)^{-1/2}]_{PQ}$$

where the indices P and Q refer to the auxiliary basis set. The auxiliary basis is defined via the Psi4 option `DF_BASIS_MP2`.

In the CD approach, the b tensor is formed via Cholesky decomposition of the exact two-electron integrals in the atomic basis. The accuracy of this decomposition (and the resulting two-electron integrals) is determined by a user defined tolerance selected via the option `CHOLESKY_TOLERANCE`. Both the DF and CD algorithms store the b tensor in memory, and therefore, they require $MN^2 \approx 3N^3$ memory for storage. On a single node with 128 GB of memory, DF and CD computations allow to treat up to 1000 orbitals.

6.3 Disk-based Density Fitting (DiskDF)

Calculations with more than 1000 basis functions quickly become unfeasible as the memory requirements of density fitting grows as the cube of basis size. In this case, it is possible to switch to a disk-based implementation of DF, which assumes that the b tensor can be fully stored on disk.

6.4 Integral Selection Keywords

The following keywords control the integral class and affect all computations that run in Forte:

INT_TYPE

`INT_TYPE` selects the integral type used in the calculation

- Type: string
- Default: `CONVENTIONAL`
- Possible Values: `CONVENTIONAL`, `DF`, `CHOLESKY`, `DISKDF`

CHOLESKY_TOLERANCE

The tolerance for the cholesky decomposition. This keyword determines the accuracy of the computation. A smaller tolerance is a more accurate computation. The tolerance for the cholesky decomposition:

- Type: double in scientific notation (ie 1e-5 or 0.0001)

- Default: `1.0e-6`

DF_BASIS_MP2

The basis set used for DF. This keyword needs to be placed in the globals section of a Psi4 input. This basis should be one of the RI basis sets designed for a given primary basis, for example, when using `BASIS = cc-pVDZ` you should use `BASIS = cc-pVDZ-RI`.

- Type: string specifing basis set
- Default: none

CHAPTER 7

Full configuration interaction

7.1 Running the test cases

Forte provides test cases for most of all methods implemented. This is a good place to start if you are new to Forte. After compiling and setting up `PYTHONPATH`, you can run the test cases:

```
cd tests/methods
python run_forte_tests_travis.py
```

CHAPTER 8

Driven Similarity Renormalization Group

8.1 Basic DSRG

8.1.1 DSRG Options

CORR_LEVEL

Correlation level of MR-DSRG.

- Type: string
- Options: PT2, PT3, LDSRG2, LDSRG2_QC, LSRG2, SRG_PT2, QDSRG2, LDSRG2_P3, QDSRG2_P3
- Default: PT2

DSRG_S

The value of the flow parameter s.

- Type: double
- Default: 1.0e10

8.2 MR-LDSRG(2)

8.2.1 MR-LDSRG(2) Options

DSRG_MAXITER

Max iterations for MR-DSRG amplitudes update.

- Type: int
- Default: 50

RELAX_REF

Relax the reference for MR-DSRG.

- Type: string
- Options: NONE, ONCE, TWICE, ITERATE
- Default: NONE

DSRG_HBAR_SEQ

Apply the sequential transformation algorithm in evaluating the transformed Hamiltonian $\bar{H}(s)$, i.e.,

$$\bar{H}(s) = e^{-\hat{A}_n(s)} \cdots e^{-\hat{A}_2(s)} e^{-\hat{A}_1(s)} \hat{H} e^{\hat{A}_1(s)} e^{\hat{A}_2(s)} \cdots e^{\hat{A}_n(s)}.$$

- Type: bool
- Default: false

DSRG_NIVO

Apply non-interacting virtual orbital (NIVO) approximation in evaluating the transformed Hamiltonian.

- Type: bool
- Default: false

CHAPTER 9

Programmer's Manual

9.1 Psi4

9.1.1 Symmetry and the `Dimension` class

In Forte, the irreducible representations (irreps) of Abelian point groups are represented using a zero-based integer. The Cotton ordering of irreps is used, which can be found here. This ordering is convenient because the direct product of two irreps can be computed using the XOR operator. For example, consider ${C_2v} symmetry. if `ha``=A1 and ``hb` , then their direct product can be computed as:

```
// Assume C2v symmetry
// Cotton ordering: [A1, A2, B1, B2]
int ha = 1; // 1 = 01 = A2
int hb = 3; // 3 = 11 = B2
int hab = ha ^ hb; // 10 = 2 = B1
```

9.1.2 The `Vector` and `Matrix` classes

CHAPTER 10

List of Forte options

ACI_ADD_AIMED_DEGENERATE

Add degenerate determinants not included in the aimed selection

- Type: Boolean
- Default value: True

ACI_ADD_EXTERNAL_EXCITATIONS

Adds external single excitations to the final wave function

- Type: Boolean
- Default value: False

ACI_ADD_SINGLES

Adds all active single excitations to the final wave function

- Type: Boolean
- Default value: False

ACI_APPROXIMATE_RDM

Approximate the RDMs

- Type: Boolean
- Default value: False

ACI_AVERAGE_OFFSET

Offset for state averaging

- Type: Integer
- Default value: 0

ACI_BATCHED_SCREENING

Control batched screeing

- Type: Boolean
- Default value: False

ACI_CONVERGENCE

ACI Convergence threshold

- Type: Double
- Default value: 0.000000

ACI_DIRECT_RDMS

Computes RDMs without coupling lists

- Type: Boolean
- Default value: False

ACI_ENFORCE_SPIN_COMPLETE

Enforce determinant spaces to be spin-complete

- Type: Boolean
- Default value: True

ACI_EXCITED_ALGORITHM

The excited state algorithm

- Type: String
- Default value: ROOT_ORTHOGONALIZE

ACI_EXTERNAL_EXCITATION_ORDER

Order of external excitations to add

- Type: String
- Default value: SINGLES

ACI_EXTERNAL_EXCITATION_TYPE

Type of external excitations to add

- Type: String
- Default value: ALL

ACI_EX_TYPE

Type of excited state to compute

- Type: String
- Default value: CONV

ACI_FIRST_ITER_ROOTS

Compute all roots on first iteration?

- Type: Boolean
- Default value: False

ACI_INITIAL_SPACE

The initial reference space

- Type: String
- Default value: CAS

ACI_LOW_MEM_SCREENING

Use low-memory screening algorithm

- Type: Boolean
- Default value: False

ACI_MAX_CYCLE

Maximum number of cycles

- Type: Integer
- Default value: 20

ACI_MAX_MEM

Sets max memory for batching algorithm (MB)

- Type: Integer
- Default value: 1000

ACI_MAX_RDM

Order of RDM to compute

- Type: Integer
- Default value: 1

ACI_MAX_RDM

Order of RDM to compute

- Type: Integer
- Default value: 1

ACI_MAX_RDM

Order of RDM to compute

- Type: Integer
- Default value: 1

ACI_NBATCH

Number of batches in screening

- Type: Integer
- Default value: 1

ACI_NFROZEN_CORE

Number of orbitals to freeze for core excitations

- Type: Integer
- Default value: 0

ACI_NO

Computes ACI natural orbitals

- Type: Boolean
- Default value: False

ACI_NO_THRESHOLD

Threshold for active space prediction

- Type: Double
- Default value: 0.020000

ACI_NROOT

Number of roots for ACI computation

- Type: Integer
- Default value: 1

ACI_N_AVERAGE

Number of roots to averag

- Type: Integer
- Default value: 1

ACI_PERTURB_SELECT

Type of energy selection

- Type: Boolean
- Default value: False

ACI_PQ_FUNCTION

Function for SA-ACI

- Type: String
- Default value: AVERAGE

ACI_PREITERATIONS

Number of iterations to run SA-ACI before SS-ACI

- Type: Integer
- Default value: 0

ACI_PRESCREEN_THRESHOLD

The SD space prescreening threshold

- Type: Double
- Default value: 0.000000

ACI_PRINT_NO

Print the natural orbitals

- Type: Boolean
- Default value: True

ACI_PRINT_REFS

Print the P space

- Type: Boolean
- Default value: False

ACI_PRINT_WEIGHTS

Print weights for active space prediction

- Type: Boolean
- Default value: False

ACI_PROJECT_OUT_SPIN_CONTAMINANTS

Project out spin contaminants in Davidson-Liu's algorithm

- Type: Boolean
- Default value: True

ACI_QUIET_MODE

Print during ACI procedure

- Type: Boolean
- Default value: False

ACI_REF_RELAX

Do reference relaxation in ACI

- Type: Boolean
- Default value: False

ACI_RELAX_SIGMA

Sigma for reference relaxation

- Type: Double
- Default value: 0.010000

ACI_ROOT

Root for single-state computations

- Type: Integer
- Default value: 0

ACI_ROOTS_PER_CORE

Number of roots to compute per frozen occupation

- Type: Integer
- Default value: 1

ACI_SAVE_FINAL_WFN

Print final wavefunction to file

- Type: Boolean
- Default value: False

ACI_SCALE_SIGMA

Scales sigma in batched algorithm

- Type: Double
- Default value: 0.500000

ACI_SELECT_TYPE

The energy selection criteria

- Type: String
- Default value: AIMED_ENERGY

ACI_SIZE_CORRECTION

Perform size extensivity correction

- Type: String
- Default value:

ACI_SPIN_ANALYSIS

Do spin correlation analysis

- Type: Boolean
- Default value: False

ACI_SPIN_PROJECTION

Type of spin projection

- Type: Integer
- Default value: 0

ACI_SPIN_TOL

Tolerance for S^2 value

- Type: Double
- Default value: 0.020000

ACI_STREAMLINE_Q

Do streamlined algorithm

- Type: Boolean
- Default value: False

ACI_TEST_RDMS

Run test for the RDMs

- Type: Boolean
- Default value: False

ACTIVE_REF_TYPE

Initial guess for active space wave functions

- Type: String
- Default value: CAS

AO_DSRG_MRPT2

Do AO-DSRG-MRPT2 if true (not available)

- Type: Boolean
- Default value: False

AVAS_DIAGONALIZE

Allow the users to specifydiagonalization of Socc and SvirIt takes priority over thethreshold based selection.

- Type: Boolean
- Default value: True

AVAS_NUM_ACTIVE

Allows the user to specify the total number of active orbitals. It takes priority over the threshold based selection.

- Type: Integer
- Default value: 0

AVAS_NUM_ACTIVE_OCC

Allows the user to specify the number of active occupied orbitals. It takes priority over the threshold based selection.

- Type: Integer
- Default value: 0

AVAS_NUM_ACTIVE_VIR

Allows the user to specify the number of active occupied orbitals. It takes priority over the threshold based selection.

- Type: Integer
- Default value: 0

AVAS_SIGMA

Threshold that controls the size of the active space

- Type: Double
- Default value: 0.980000

CCVV_ALGORITHM

Algorithm to compute the CCVV term in DSRG-MRPT2 (only used in three-dsrg-mrpt2 code)

- Type: String
- Default value: FLY_AMBIT
- Allowed values: CORE, FLY_AMBIT, FLY_LOOP, BATCH_CORE, BATCH_VIRTUAL, BATCH_CORE_GA, BATCH_VIRTUAL_GA, BATCH_VIRTUAL_MPI, BATCH_CORE_MPI, BATCH_CORE_REP, BATCH_VIRTUAL_REP

CCVV_BATCH_NUMBER

Batches for CCVV_ALGORITHM

- Type: Integer
- Default value: -1

CCVV_SOURCE

Special treatment for the CCVV term in DSRG-MRPT2 (used in three-dsrg-mrpt2 code)

- Type: String
- Default value: NORMAL

- Allowed values: ZERO, NORMAL

CHOLESKY_TOLERANCE

The tolerance for cholesky integrals

- Type: Double
- Default value: 0.000001

CINO

Do a CINO computation?

- Type: Boolean
- Default value: False

CINO_AUTO

Allow the users to choosewhether pass frozen_doccactice_docc and restricted_doccor not

- Type: Boolean
- Default value: False

CINO_NROOT

The number of roots computed

- Type: Integer
- Default value: 1

CINO_ROOTS_PER_IRREP

The number of excited states per irreducible representation

- Type: Array
- Default value: []

CINO_THRESHOLD

The fraction of NOs to include in the active space

- Type: Double
- Default value: 0.990000

CINO_TYPE

The type of wave function.

- Type: String
- Default value: CIS
- Allowed values: CIS, CISD

CORR_LEVEL

Correlation level of MR-DSRG (used in mrdsrg code, LDSRG2_P3 and QDSRG2_P3 not implemented)

- Type: String
- Default value: PT2
- Allowed values: PT2, PT3, LDSRG2, LDSRG2_QC, LSRG2, SRG_PT2, QDSRG2, LDSRG2_P3, QDSRG2_P3

DL_GUESS_SIZE

Set the initial guess space size for DL solver

- Type: Integer
- Default value: 100

DSRGPT

Renormalize (if true) the integrals (only used in toy code mcsrgpt2)

- Type: Boolean
- Default value: True

DSRG_DIPOLE

Compute (if true) DSRG dipole moments

- Type: Boolean
- Default value: False

DSRG_HBAR_SEQ

Evaluate H_bar sequentially if true

- Type: Boolean
- Default value: False

DSRG_MAXITER

Max iterations for MR-DSRG amplitudes update

- Type: Integer
- Default value: 50

DSRG_MRPT2_DEBUG

Excssive printing for three-dsrg-mrpt2

- Type: Boolean
- Default value: False

DSRG_MULTI_STATE

Multi-state DSRG options (MS and XMS recouple states after single-state computations)

- Type: String
- Default value: SA_FULL
- Allowed values: SA_FULL, SA_SUB, MS, XMS

DSRG_OMIT_V3

Omit blocks with >= 3 virtual indices if true

- Type: Boolean
- Default value: False

DSRG_TRANS_TYPE

DSRG transformation type

- Type: String

- Default value: UNITARY
- Allowed values: UNITARY, CC

DWMS_ALGORITHM

DWMS algorithms

- Type: String
- Default value: DWMS-0
- Allowed values: DWMS-0, DWMS-1, DWMS-AVG0, DWMS-AVG1

DWMS_ZETA

Gaussian width cutoff for the density weights

- Type: Double
- Default value: 0.000000

ESNOS

Compute external single natural orbitals

- Type: Boolean
- Default value: False

ESNO_MAX_SIZE

Number of external orbitals to correlate

- Type: Integer
- Default value: 0

FCIMO_ACTV_TYPE

The active space type

- Type: String
- Default value: COMPLETE
- Allowed values: COMPLETE, CIS, CISD, DOCI

FCIMO_CISD_NOHF

Ground state: HF; Excited states: no HF determinant in CISD space

- Type: Boolean
- Default value: True

FCIMO_IAO_ANALYSIS

Intrinsic atomic orbital analysis

- Type: Boolean
- Default value: False

FCIMO_IPEA

Generate IP/EA CIS/CISD space

- Type: String
- Default value: NONE

- Allowed values: NONE, IP, EA

FCIMO_LOCALIZE_ACTV

Localize active orbitals before computation

- Type: Boolean
- Default value: False

FCIMO_PRINT_CIVEC

The printing threshold for CI vectors

- Type: Double
- Default value: 0.050000

FCI_MAXITER

Maximum number of iterations for FCI code

- Type: Integer
- Default value: 30

FCI_MAX_RDM

The number of trial guess vectors to generate per root

- Type: Integer
- Default value: 1

FCI_NROOT

The number of roots computed

- Type: Integer
- Default value: 1

FCI_NTRIAL_PER_ROOT

The number of trial guess vectors to generate per root

- Type: Integer
- Default value: 10

FCI_PRINT_NO

Print the NO from the rdm of FCI

- Type: Boolean
- Default value: False

FCI_ROOT

The root selected for state-specific computations

- Type: Integer
- Default value: 0

FCI_TEST_RDMS

Test the FCI reduced density matrices?

- Type: Boolean

- Default value: False

FORM_HBAR3

Form 3-body Hbar (only used in dsrg-mrpt2 with SA_SUB for testing)

- Type: Boolean
- Default value: False

FORM_MBAR3

Form 3-body mbar (only used in dsrg-mrpt2 for testing)

- Type: Boolean
- Default value: False

GAMMA

The reference space selection threshold

- Type: Double
- Default value: 1.000000

H0TH

Zeroth-order Hamiltonian of DSRG-MRPT (used in mrdsrg code)

- Type: String
- Default value: FDIAG
- Allowed values: FDIAG, FFULL, FDIAG_VACTV, FDIAG_VDIAG

INTEGRAL_SCREENING

The screening for JK builds and DF libraries

- Type: Double
- Default value: 0.000000

INTERNAL_AMP

Include internal amplitudes for VCIS/VCISD-DSRG

- Type: String
- Default value: NONE
- Allowed values: NONE, SINGLES_DOUBLES, SINGLES, DOUBLES

INTERNAL_AMP_SELECT

Excitation types considered when internal amplitudes are included

- Type: String
- Default value: AUTO
- Allowed values: AUTO, ALL, OOVV

INTRUDER_TAMP

Threshold for amplitudes considered as intruders for warning

- Type: Double
- Default value: 0.100000

INT_TYPE

The integral type

- Type: String
- Default value: CONVENTIONAL
- Allowed values: CONVENTIONAL, DF, CHOLESKY, DISKDF, DISTDF, ALL, OWNINTEGRALS

ISA_B

Intruder state avoidance parameter when use ISA to form amplitudes (only used in toy code mcsrgpt2)

- Type: Double
- Default value: 0.020000

JOB_TYPE

Specify the job type

- Type: String
- Default value: NONE
- Allowed values: NONE, ACI, PCI, CAS, DMRG, SR-DSRG, SR-DSRG-ACI, SR-DSRG-PCI, TENSORSRG, TENSORSRG-CI, DSRG-MRPT2, DSRG-MRPT3, MR-DSRG-PT2, THREE-DSRG-MRPT2, SOMRDSRG, MRDSRG, MRDSRG_SO, CASSCF, ACTIVE-DSRGPT2, DWMS-DSRGPT2, DSRG_MRPT, TASKS, CC, NOJOB, DOCUMENTATION

MAXITER_RELAX_REF

Max macro iterations for DSRG reference relaxation

- Type: Integer
- Default value: 15

MINAO_BASIS

The basis used to define an orbital subspace

- Type: String
- Default value: STO-3G

MRCINO

Do a MRCINO computation?

- Type: Boolean
- Default value: False

MRCINO_AUTO

Allow the users to choosewhether pass frozen_doccactice_docc and restricted_doccor not

- Type: Boolean
- Default value: False

MRCINO_NROOT

The number of roots computed

- Type: Integer
- Default value: 1

MRCINO_ROOTS_PER_IRREP

The number of excited states per irreducible representation

- Type: Array
- Default value: []

MRCINO_THRESHOLD

The fraction of NOs to include in the active space

- Type: Double
- Default value: 0.990000

MRCINO_TYPE

The type of wave function.

- Type: String
- Default value: CIS
- Allowed values: CIS, CISD

MRPT2

Compute full PT2 energy

- Type: Boolean
- Default value: False

MS

Projection of spin onto the z axis

- Type: Double
- Default value: 0.000000

NTAMP

Number of amplitudes printed in the summary

- Type: Integer
- Default value: 15

N_GUESS_VEC

Number of guess vectors for Sparse CI solver

- Type: Integer
- Default value: 10

PCI_ADAPTIVE_BETA

Use an adaptive time step?

- Type: Boolean
- Default value: False

PCI_CHEBYSHEV_ORDER

The order of Chebyshev truncation

- Type: Integer

- Default value: 5

PCI_COLINEAR_THRESHOLD

The minimum norm of orthogonal vector

- Type: Double
- Default value: 0.000001

PCI_DL_COLLAPSE_PER_ROOT

The number of trial vector to retain after Davidson-Liu collapsing

- Type: Integer
- Default value: 2

PCI_DL_SUBSPACE_PER_ROOT

The maxim number of trial Davidson-Liu vectors

- Type: Integer
- Default value: 8

PCI_DYNAMIC_PRESCREENING

Use dynamic prescreening

- Type: Boolean
- Default value: False

PCI_ENERGY_ESTIMATE_FREQ

Iterations in between variational estimation of the energy

- Type: Integer
- Default value: 1

PCI_ENERGY_ESTIMATE_THRESHOLD

The threshold with which we estimate the variational energy. Note that the final energy is always estimated exactly.

- Type: Double
- Default value: 0.000001

PCI_EVAR_MAX_ERROR

The max allowed error for variational energy

- Type: Double
- Default value: 0.000000

PCI_E_CONVERGENCE

The energy convergence criterion

- Type: Double
- Default value: 0.000000

PCI_FAST_EVAR

Use a fast (sparse) estimate of the energy

- Type: Boolean

- Default value: False

PCI_FUNCTIONAL

The functional for determinant coupling importance evaluation

- Type: String
- Default value: MAX
- Allowed values: MAX, SUM, SQUARE, SQRT, SPECIFY-ORDER

PCI_FUNCTIONAL_ORDER

The functional order of PCI_FUNCTIONAL is SPECIFY-ORDER

- Type: Double
- Default value: 1.000000

PCI_GENERATOR

The propagation algorithm

- Type: String
- Default value: WALL-CHEBYSHEV
- Allowed values: LINEAR, QUADRATIC, CUBIC, QUARTIC, POWER, TROTTER, OLSEN, DAVIDSON, MITRUSHENKOV, EXP-CHEBYSHEV, WALL-CHEBYSHEV, CHEBYSHEV, LANCZOS, DL

PCI_GUESS_SPAWNING_THRESHOLD

The determinant importance threshold

- Type: Double
- Default value: -1.000000

PCI_INITIATOR_APPROX

Use initiator approximation

- Type: Boolean
- Default value: False

PCI_INITIATOR_APPROX_FACTOR

The initiator approximation factor

- Type: Double
- Default value: 1.000000

PCI_KRYLOV_ORDER

The order of Krylov truncation

- Type: Integer
- Default value: 5

PCI_MAXBETA

The maximum value of beta

- Type: Double
- Default value: 1000.000000

PCI_MAX_DAVIDSON_ITER

The maximum value of Davidson generator iteration

- Type: Integer
- Default value: 12

PCI_MAX_GUESS_SIZE

The maximum number of determinants used to form the guess wave function

- Type: Double
- Default value: 10000.000000

PCI_NROOT

The number of roots computed

- Type: Integer
- Default value: 1

PCI_PERTURB_ANALYSIS

Do result perturbation analysis

- Type: Boolean
- Default value: False

PCI_POST_DIAGONALIZE

Do a post diagonalization?

- Type: Boolean
- Default value: False

PCI_PRINT_FULL_WAVEFUNCTION

Print full wavefunction when finish

- Type: Boolean
- Default value: False

PCI_REFERENCE_SPAWNING

Do spawning according to reference

- Type: Boolean
- Default value: False

PCI_SCHWARZ_PRESCREENING

Use schwarz prescreening

- Type: Boolean
- Default value: False

PCI_SIMPLE_PRESCREENING

Prescreen the spawning of excitations

- Type: Boolean
- Default value: False

PCI_SPAWNING_THRESHOLD

The determinant importance threshold

- Type: Double
- Default value: 0.001000

PCI_STOP_HIGHER_NEW_LOW

Stop iteration when higher new low detected

- Type: Boolean
- Default value: False

PCI_SYMM_APPROX_H

Use Symmetric Approximate Hamiltonian

- Type: Boolean
- Default value: False

PCI_TAU

The time step in imaginary time (a.u.)

- Type: Double
- Default value: 1.000000

PCI_USE_INTER_NORM

Use intermediate normalization

- Type: Boolean
- Default value: False

PCI_USE_SHIFT

Use a shift in the exponential

- Type: Boolean
- Default value: False

PCI_VAR_ESTIMATE

Estimate variational energy during calculation

- Type: Boolean
- Default value: False

PI_ACTIVE_SPACE

Active space type

- Type: Boolean
- Default value: False

PRINT_1BODY_EVALS

Print eigenvalues of 1-body effective H

- Type: Boolean
- Default value: False

PRINT_DENOM2

Print (if true) renormalized denominators in DSRG-MRPT2

- Type: Boolean
- Default value: False

PRINT_IAOS

Print IAOs

- Type: Boolean
- Default value: True

PRINT_INTS

Print the one- and two-electron integrals?

- Type: Boolean
- Default value: False

PRINT_TIME_PROFILE

Print detailed timings in dsrg-mrpt3

- Type: Boolean
- Default value: False

PT2_MAX_MEM

Maximum size of the determinant hash (GB)

- Type: Double
- Default value: 1.000000

RELAX_REF

Relax the reference for MR-DSRG (used in dsrg-mrpt2/3, mrdsrg)

- Type: String
- Default value: NONE
- Allowed values: NONE, ONCE, TWICE, ITERATE

R_CONVERGENCE

Convergence criteria for amplitudes

- Type: Double
- Default value: 0.000001

SAVE_FINAL_WFN

Save the final wavefunction to a file

- Type: Boolean
- Default value: False

SIGMA

The energy selection threshold

- Type: Double

- Default value: 0.010000

SMART_DSRG_S

Automatic adjust the flow parameter according to denominators

- Type: String
- Default value: DSRG_S
- Allowed values: DSRG_S, MIN_DELTA1, MAX_DELTA1, DAVG_MIN_DELTA1, DAVG_MAX_DELTA1

SOURCE

Source operator used in DSRG (AMP, EMP2, LAMP, LEMP2 only available in toy code mcsrgpt2)

- Type: String
- Default value: STANDARD
- Allowed values: STANDARD, LABS, DYSON, AMP, EMP2, LAMP, LEMP2

SPIN_BASIS

Basis for spin analysis

- Type: String
- Default value: LOCAL

SPIN_MAT_TO_FILE

Save spin correlation matrix to file

- Type: Boolean
- Default value: False

SPIN_PROJECT_FULL

Project solution in full diagonalization algorithm

- Type: Boolean
- Default value: False

SUBSPACE

A list of orbital subspaces

- Type: Array
- Default value: []

T1_AMP

The way of forming T1 amplitudes (used in toy code mcsrgpt2)

- Type: String
- Default value: DSRG
- Allowed values: DSRG, SRG, ZERO

TAYLOR_THRESHOLD

Taylor expansion threshold for small denominator

- Type: Integer
- Default value: 3

THREEPDC_ALGORITHM

Algorithm for evaluating 3-body cumulants in three-dsrg-mrpt2

- Type: String
- Default value: CORE
- Allowed values: CORE, BATCH

THREE_MRPT2_TIMINGS

Detailed printing (if true) in three-dsrg-mrpt2

- Type: Boolean
- Default value: False

T_ALGORITHM

The way of forming amplitudes (DSRG_NOSEMI, SELEC, ISA only available in toy code mcsrgpt2)

- Type: String
- Default value: DSRG
- Allowed values: DSRG, DSRG_NOSEMI, SELEC, ISA

UNPAIRED_DENSITY

Compute unpaired electron density

- Type: Boolean
- Default value: False

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Index

I

Integrals, 13
    theory, 13

T

theory
    Integrals, 13