

Module Interface Specification for EOMEE

Gabriela Sánchez Díaz

December 9, 2020

1 Revision History

Date	Version	Notes
20-11-2020	1.0	MIS first draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [SRS](#).

Additionally, the following abbreviations were used:

abbreviation	description
N	Number of electrons
orthog	Matrix orthogonalization method
tol	Tolerance
nspino	Number of spin orbital basis
lhs	Left-hand-side
rhs	Right-hand-side
neigs	Number of eigenvalues

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	2
6	MIS of Control Module	2
6.1	Module	2
6.2	Uses	2
6.3	Syntax	2
6.3.1	Exported Constants	2
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	3
6.4.5	Local Functions	3
7	MIS of Input Module	4
7.1	Module	4
7.2	Uses	4
7.3	Syntax	4
7.3.1	Exported Constants	4
7.3.2	Exported Access Programs	4
7.4	Semantics	4
7.4.1	State Variables	4
7.4.2	Environment Variables	5
7.4.3	Assumptions	5
7.4.4	Access Routine Semantics	5
7.4.5	Local Functions	6
8	MIS of Integrals Module	7
8.1	Template Module	7
8.2	Uses	7
8.3	Syntax	7
8.3.1	Exported Constants	7
8.3.2	Exported Access Programs	7

8.4	Semantics	7
8.4.1	State Variables	7
8.4.2	Environment Variables	7
8.4.3	Assumptions	7
8.4.4	Access Routine Semantics	7
8.4.5	Local Functions	8
9	MIS of RDMs Module	9
9.1	Template Module	9
9.2	Uses	9
9.3	Syntax	9
9.3.1	Exported Constants	9
9.3.2	Exported Access Programs	9
9.4	Semantics	9
9.4.1	State Variables	9
9.4.2	Environment Variables	9
9.4.3	Assumptions	9
9.4.4	Access Routine Semantics	10
9.4.5	Local Functions	10
10	MIS of EOM Base Module	12
10.1	Interface Module	12
10.2	Uses	12
10.3	Syntax	12
10.3.1	Exported Constants	12
10.3.2	Exported Access Programs	12
10.4	Semantics	12
10.4.1	State Variables	12
10.4.2	Assumptions	12
10.4.3	Local Functions	13
10.4.4	Considerations	13
11	MIS of EOM IP Module	14
11.1	Template Module	14
11.2	Uses	14
11.3	Syntax	14
11.3.1	Exported Constants	14
11.3.2	Exported Access Programs	14
11.4	Semantics	14
11.4.1	State Variables	14
11.4.2	Environment Variables	14
11.4.3	Assumptions	14
11.4.4	Access Routine Semantics	15

11.4.5	Local Functions	16
12	MIS of EOM EA Module	17
12.1	Template Module	17
12.2	Uses	17
12.3	Syntax	17
12.3.1	Exported Constants	17
12.3.2	Exported Access Programs	17
12.4	Semantics	17
12.4.1	State Variables	17
12.4.2	Environment Variables	17
12.4.3	Assumptions	17
12.4.4	Access Routine Semantics	18
12.4.5	Local Functions	19
13	MIS of EOM Excitation Module	20
13.1	Template Module	20
13.2	Uses	20
13.3	Syntax	20
13.3.1	Exported Constants	20
13.3.2	Exported Access Programs	20
13.4	Semantics	20
13.4.1	State Variables	20
13.4.2	Environment Variables	20
13.4.3	Assumptions	20
13.4.4	Access Routine Semantics	21
13.4.5	Local Functions	22
14	MIS of EOM DIP Module	23
14.1	Template Module	23
14.2	Uses	23
14.3	Access Routine Semantics	23
14.3.1	Local Functions	23
15	MIS of EOM DEA Module	24
15.1	Template Module	24
15.2	Uses	24
15.3	Access Routine Semantics	24
15.3.1	Local Functions	24
16	MIS of Output module	25
16.1	Module	25
16.2	Uses	25

16.3	Syntax	25
16.3.1	Exported Constants	25
16.3.2	Exported Access Programs	25
16.4	Semantics	25
16.4.1	State Variables	25
16.4.2	Environment Variables	25
16.4.3	Assumptions	25
16.4.4	Access Routine Semantics	25
16.4.5	Local Functions	25
17	MIS of Solver Module	26
17.1	Module	26
17.2	Uses	26
17.3	Syntax	26
17.3.1	Exported Constants	26
17.3.2	Exported Access Programs	26
17.4	Semantics	26
17.4.1	State Variables	26
17.4.2	Environment Variables	26
17.4.3	Assumptions	26
17.4.4	Access Routine Semantics	26
17.4.5	Local Functions	26
18	Appendix	27

3 Introduction

The following document details the Module Interface Specifications of EOMEE, a set of tools to implement and solve the Equation-of-Motion methods for excited states.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/gabrielasd/eomee/tree/cas741>.

4 Notation

The structure of the MIS for modules comes from ?, with the addition that template modules have been adapted from ?. The mathematical notation comes from Chapter 3 of ?. For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by EOMEE.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of EOMEE uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, EOMEE uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

The following simplifications will be made in the mathematical notation for the sake of understandability:

- $\text{seq}(l_1, l_2, \dots, l_n; T)$, will be used instead of sequence $[l_1, l_2, \dots, l_n]$ of type T . For example $\text{seq}(n, m; \mathbb{R})$, where $n, m > 0$, would map to sequence $[n, m]$ of type \mathbb{R} . This type will generally be used to indicate NumPy.ndarray data types.
- Variables that are of type sequence will be denoted in bold font, i.e, the parameter \mathbf{x} denotes a sequence.
- Subscripts will be used for indexing sequences, for instance, x_i will represent the i th element of \mathbf{x} , the same as $x[i]$ from ?.

- str will be used instead of string.
- bool will be used instead of boolean.

Also, the absence of value will be defined by Python’s data type NoneType, denoted as None.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2	Level 3
Hardware-Hiding Module		
Behaviour-Hiding Module	Control	
	Input	
	Integrals	
	RDMs	
	EOM Interface	IP EOM; EA EOM; DIP EOM; DEA EOM; Excitation EOM
	Output	
Software Decision Module	Solver	

Table 1: Module Hierarchy

6 MIS of Control Module

6.1 Module

main

6.2 Uses

input (7), Integrals (8), WfnRDMs (9), EOMIP (11), EOMEA (12), EOMExc (13), EOMDIP (14), EOMDEA (15), solver (17), output (16)

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	str	-	-

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

None

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

main():

- transition: The following steps are performed:
Get a file containing the input parameters from the user (inputFile).

Parse the file's content and verify all required input parameters are present.

Load and verify the electron integrals (\mathbf{h}, \mathbf{v}) and RDMs (γ, Γ)

Define an EOM type equation from the parameters $\mathbf{h}, \mathbf{v}, \gamma$ and Γ .

Solve the EOM eigenvalue problem and evaluate the TDMs

Output the results of the computations:

- exception: None

6.4.5 Local Functions

None

7 MIS of Input Module

7.1 Module

input

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
parse_inputfile	str	ParsedParams	FileNotFoundError
check_inputs	ParsedParams	-	FileNotFoundError, ValueError
N		$(n1, n2 : \mathbb{Z})$	
<i>one_int_file</i>		str	
<i>two_int_file</i>		str	
<i>dm1_file</i>		str	
<i>dm2_file</i>		str	
<i>eom</i>		str	
<i>orthog</i>		str	
<i>tol</i>		\mathbb{R}	

7.4 Semantics

7.4.1 State Variables

$N: \mathbb{Z} \vee (n1, n2 : \mathbb{Z})$

one_int_file: str

two_int_file: str

dm1_file: str

dm2_file: str

eom: str $\in \{“ip”, “dip”, “ea”, “dea”, “exc”\}$ which selects the EOM method

orthog: str $\in \{“symmetric”, “asymmetric”\}$

tol: $\mathbb{R} > 0$

7.4.2 Environment Variables

inputFile: string representing a file or file path.

7.4.3 Assumptions

The first function called will be `parse_infile`, followed by `check_inputs`.

7.4.4 Access Routine Semantics

`parse_infile(filename)`:

- transition: The input file *filename* is read sequentially and the state variables get assigned
- output: *out* := ParsedParams
- exception: FileNotFoundError

`check_inputs(ParsedParams)`:

- output: None
- exception: *exc* :=

$\neg(N \in (n1, n2 : \mathbb{Z}))$	\Rightarrow TypeError
"one_int_file" not in working directory	\Rightarrow FileNotFoundError
"two_int_file" not in working directory	\Rightarrow FileNotFoundError
"dm1_file" not in working directory	\Rightarrow FileNotFoundError
"dm2_file" not in working directory	\Rightarrow FileNotFoundError
$\neg(eom \in \{"ip", "dip", "ea", "dea", "exc"\})$	\Rightarrow ValueError
$\neg(orthog \in \{"symmetric", "asymmetric"\})$	\Rightarrow ValueError
$\neg(tol \in \mathbb{R})$	\Rightarrow TypeError
$\neg(tol > 0)$	\Rightarrow ValueError

`ParsedParams.N`:

- output: *out* := *N*
- exception: None

`ParsedParams.tol`:

- output: *out* := *tol*
- exception: None

ParsedParams.*orthog*:

- output: *out* := *orthog*
- exception: None

ParsedParams.*com*:

- output: *out* := *com*
- exception: None

ParsedParams.*one_int_file*:

- output: *out* := *one_int_file*
- exception: None

ParsedParams.*two_int_file*:

- output: *out* := *two_int_file*
- exception: None

ParsedParams.*dm1_file*:

- output: *out* := *two_int_file*
- exception: None

ParsedParams.*dm2_file*:

- output: *out* := *two_int_file*
- exception: None

7.4.5 Local Functions

None

8 MIS of Integrals Module

8.1 Template Module

Integrals

8.2 Uses

input (7)

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
new Integrals	str, str	Integrals	-
h	-	$\text{seq}(m, m : \mathbb{R})$	-
v	-	$\text{seq}(m, m, m, m : \mathbb{R})$	-
<i>nspino</i>	-	\mathbb{Z}	-

8.4 Semantics

8.4.1 State Variables

h: $\text{seq}(m, m : \mathbb{R})$

v: $\text{seq}(m, m, m, m : \mathbb{R})$

nspino: \mathbb{Z}

8.4.2 Environment Variables

intfile1: binary file in NumPy .npy format.

intfile2: binary file in NumPy .npy format.

8.4.3 Assumptions

The constructor of Integrals will be called before any state variable is invoked.

8.4.4 Access Routine Semantics

new Integrals(*one_int_file*, *two_int_file*):

- transition: Call `load_integrals(one_int_file, two_int_file)`
- output: *out* := self

- exception: None

Integrals.h:

- output: *out* := **h**
- exception: None

Integrals.v:

- output: *out* := **v**
- exception: None

Integrals.nspino:

- output: *out* := *nspino*
- exception: None

8.4.5 Local Functions

load_integrals(*one_int_file*, *two_int_file*):

- transition:
Read the binary files *one_int_file* and *two_int_file*
verify_integrals()
If no exception is raised, assign the state variables **h** and **v**
- exception: *exc* := FileNotFoundError

verify_integrals():

- output: *out* := None
- exception: *exc* :=

$\neg(\mathbf{h} \in \text{sequence of } \mathbb{R})$	$\Rightarrow \text{TypeError}$
$\neg(\mathbf{v} \in \text{sequence of } \mathbb{R})$	$\Rightarrow \text{TypeError}$
h is not a bidimensional array	$\Rightarrow \text{ValueError}$
v is not a 4 dimensional array	$\Rightarrow \text{ValueError}$
$\neg(\mathbf{h}[0] = \mathbf{v}[0])$	$\Rightarrow \text{ValueError}$
$\neg(h_{ij} = h_{ji})$	$\Rightarrow \text{ValueError}$
$\neg((v_{ijkl} = v_{jilk}) \wedge (v_{ijkl} = v_{klij}))$	$\Rightarrow \text{ValueError}$
$\neg((v_{ijkl} = -v_{jikl}) \wedge (v_{ijkl} = -v_{ijlk}))$	$\Rightarrow \text{ValueError}$

9 MIS of RDMS Module

9.1 Template Module

WfnRDMS

9.2 Uses

input (7)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
new WfnRDMS	$\mathbb{Z} \vee (n1, n2 : \mathbb{Z}), \text{ str},$ str	WfnRDMS	-
γ	-	$\text{seq}(m, m : \mathbb{R})$	-
Γ	-	$\text{seq}(m, m, m, m : \mathbb{R})$	-
N	-	$(n1, n2 : \mathbb{Z})$	-
<i>nspino</i>	-	\mathbb{Z}	-

9.4 Semantics

9.4.1 State Variables

$N: (n1, n2 : \mathbb{Z})$

nspino: \mathbb{Z}

γ : $\text{seq}(m, m : \mathbb{R})$, where $0 \leq \gamma_{ij} \leq 1$

Γ : $\text{seq}(m, m, m, m : \mathbb{R})$, where $0 \leq \Gamma_{ijkl} \leq 1$

9.4.2 Environment Variables

file1: binary file in NumPy .npy format.

file2: binary file in NumPy .npy format.

9.4.3 Assumptions

The constructor of WfnRDMS will be called before invoking any state variable.

9.4.4 Access Routine Semantics

new WfnRDMS($n1$, $dm1_file$, $dm2_file$):

- transition:
 $N := n1$
 Call `assign_rdms($dm1_file$, $dm2_file$)`
- output: $out := self$
- exception: None

WfnRDMS.dm1:

- output: $out := \gamma$
- exception: None

WfnRDMS.dm2:

- output: $out := \Gamma$
- exception: None

WfnRDMS.N:

- output: $out := N$
- exception: None

WfnRDMS.nspino:

- output: $out := nspino$
- exception: None

9.4.5 Local Functions

`assign_rdms($dm1_file$, $dm2_file$):`

- transition: Read the binary files $dm1_file$ and $dm2_file$.
 `verify_rdms()`
 If no exception is raised, assign the state variables γ and Γ
- exception: $exc := FileNotFoundError$

`verify_rdms():`

- output: $out := None$
- exception: $exc :=$

$\neg(\boldsymbol{\gamma} \in \text{sequence of } \mathbb{R})$	$\Rightarrow \text{TypeError}$
$\neg(\boldsymbol{\Gamma} \in \text{sequence of } \mathbb{R})$	$\Rightarrow \text{TypeError}$
$\boldsymbol{\gamma}$ is not a bidimensional array	$\Rightarrow \text{ValueError}$
$\boldsymbol{\Gamma}$ is not a 4 dimensional array	$\Rightarrow \text{ValueError}$
$\neg(\gamma_{ij} = \gamma_{ji})$	$\Rightarrow \text{ValueError}$
$\neg(\Gamma_{ijkl} = \Gamma_{jilk}) \vee \neg(\Gamma_{ijkl} = \Gamma_{klij})$	$\Rightarrow \text{ValueError}$
$\neg(\Gamma_{ijkl} = -\Gamma_{jikl}) \vee \neg(\Gamma_{ijkl} = -\Gamma_{ijlk})$	$\Rightarrow \text{ValueError}$
$\text{Tr}(\boldsymbol{\gamma}) \neq N$	$\Rightarrow \text{ValueError}$
$\text{Tr}(\boldsymbol{\Gamma}) \neq N(N - 1)$	$\Rightarrow \text{ValueError}$

10 MIS of EOM Base Module

10.1 Interface Module

EOMBase

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
<i>neigs</i>	-	\mathbb{Z}	NotImplementedError
<code>compute_tdm</code>	$\text{seq}(k, k: \mathbb{R})$	$\text{seq}(k, m, m: \mathbb{R})$	NotImplementedError
<i>lhs</i>	-	$\text{seq}(k, k: \mathbb{R})$	-
<i>rhs</i>	-	$\text{seq}(k, k: \mathbb{R})$	-
<i>nspino</i>	-	\mathbb{Z}	-
h	-	$\text{seq}(m, m: \mathbb{R})$	-
v	-	$\text{seq}(m, m, m, m: \mathbb{R})$	-
γ	-	$\text{seq}(m, m: \mathbb{R})$	-
Γ	-	$\text{seq}(m, m, m, m: \mathbb{R})$	-

10.4 Semantics

10.4.1 State Variables

nspino: \mathbb{Z}

h: $\text{seq}(m, m: \mathbb{R})$

v: $\text{seq}(m, m, m, m: \mathbb{R})$

γ : $\text{seq}(m, m: \mathbb{R})$

Γ : $\text{seq}(m, m, m, m: \mathbb{R})$

lhs: `_compute_lhs()`

rhs: `_compute_rhs()`

10.4.2 Assumptions

The EOMBase module can't be instantiated, it is inherited by EOMIP, EOMEA, EOMExc, EOMDIP and EOMDEA.

10.4.3 Local Functions

`_compute_lhs()`:

- exception: `NotImplementedError`

`_compute_rhs()`:

- exception: `NotImplementedError`

10.4.4 Considerations

EOMBase is an abstract class (ABC) defining an interface for the different EOM methods (Subsections (11), (12), (13), (14) and (15)). Each state variable has a corresponding access program. Only the methods *neigs*, `compute_tdm`, `_compute_lhs` and `_compute_rhs` are abstract.

11 MIS of EOM IP Module

11.1 Template Module

EOMIP inherits EOMBase

11.2 Uses

EOMBase (10), Integrals (8), WfnRDMs (9)

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
new EOMIP	$\text{seq}(m, m : \mathbb{R}),$ $\text{seq}(m, m, m, m : \mathbb{R}),$ $\text{seq}(m, m : \mathbb{R}),$ $\text{seq}(m, m, m, m : \mathbb{R})$	EOMIP	-

11.4 Semantics

11.4.1 State Variables

nspino: \mathbb{Z}

h: $\text{seq}(m, m : \mathbb{R})$

v: $\text{seq}(m, m, m, m : \mathbb{R})$

γ : $\text{seq}(m, m : \mathbb{R})$

Γ : $\text{seq}(m, m, m, m : \mathbb{R})$

lhs: `_compute_lhs()`

rhs: `_compute_rhs()`

11.4.2 Environment Variables

None

11.4.3 Assumptions

The EOMIP constructor is called before any other access program in the class.

11.4.4 Access Routine Semantics

new EOMIP($\mathbf{h}, \mathbf{v}, \text{dm1}, \text{dm2}$):

- transition: $\mathbf{h}, \mathbf{v}, \boldsymbol{\gamma}, \boldsymbol{\Gamma} := \mathbf{h}, \mathbf{v}, \text{dm1}, \text{dm2}$,
 $lhs := _compute_lhs()$
 $rhs := _compute_rhs()$
 $nspino := |\mathbf{h}[0]|$

- output: $out := self$

- exception: None

neigs():

- output: $out := |\mathbf{h}[0]|$
- exception: None

compute_tdm(\mathbf{c}):

- output: $out := \sum_n \gamma_{mn} c_n, \{n : \mathbb{Z} | 0 \leq n < nspino\}$
- exception: None

EOMIP.nspino:

- output: $out := nspino$
- exception: None

EOMIP.h:

- output: $out := \mathbf{h}$
- exception: None

EOMIP.v:

- output: $out := \mathbf{v}$
- exception: None

EOMIP.dm1:

- output: $out := \boldsymbol{\gamma}$
- exception: None

EOMIP.dm2:

- output: $out := \mathbf{\Gamma}$
- exception: None

EOMIP.lhs:

- output: $out := lhs \in \text{seq}(m, m : \mathbb{R})$
- exception: ValueError

EOMIP.rhs:

- output: $out := rhs \in \text{seq}(m, m : \mathbb{R})$
- exception: ValueError

11.4.5 Local Functions

`_compute_lhs()`:

- output: $out := -\mathbf{h}\boldsymbol{\gamma} + 0.5 \sum_{qrs} v_{qnrs} \Gamma_{mqr s}$
- exception: None

`_compute_rhs()`:

- output: $out := \boldsymbol{\gamma}$
- exception: None

12 MIS of EOM EA Module

12.1 Template Module

EOMEA inherits EOMBase

12.2 Uses

EOMBase (10), Integrals (8), WfnRDMs (9)

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
new	$\text{seq}(m, m : \mathbb{R})$,	EOMEA	-
EOMEA	$\text{seq}(m, m, m, m : \mathbb{R})$, $\text{seq}(m, m : \mathbb{R})$, $\text{seq}(m, m, m, m : \mathbb{R})$		

12.4 Semantics

12.4.1 State Variables

nspino: \mathbb{Z}

h: $\text{seq}(m, m : \mathbb{R})$

v: $\text{seq}(m, m, m, m : \mathbb{R})$

γ : $\text{seq}(m, m : \mathbb{R})$

Γ : $\text{seq}(m, m, m, m : \mathbb{R})$

lhs: `_compute_lhs()`

rhs: `_compute_rhs()`

12.4.2 Environment Variables

None

12.4.3 Assumptions

The EOMEA constructor is called before any other access program in that class.

12.4.4 Access Routine Semantics

new EOMEA($\mathbf{h}, \mathbf{v}, \mathbf{dm1}, \mathbf{dm2}$):

- transition: $\mathbf{h}, \mathbf{v}, \boldsymbol{\gamma}, \boldsymbol{\Gamma} := \mathbf{h}, \mathbf{v}, \mathbf{dm1}, \mathbf{dm2}$,
 $lhs := _compute_lhs()$,
 $rhs := _compute_rhs()$
 $nspino := |\mathbf{h}[0]|$

- output: $out := self$

- exception: None

neigs():

- output: $out := |\mathbf{h}[0]|$
- exception: None

compute_tdm(\mathbf{c}):

- output: $out := \sum_n (\delta_{mn} - \gamma_{mn}) c_n, \{n : \mathbb{Z} | 0 \leq n < nspino\}$
- exception: None

EOMEA.nspino:

- output: $out := nspino$
- exception: None

EOMEA.h:

- output: $out := \mathbf{h}$
- exception: None

EOMEA.v:

- output: $out := \mathbf{v}$
- exception: None

EOMEA.dm1:

- output: $out := \boldsymbol{\gamma}$
- exception: None

EOMEA.dm2:

- output: $out := \mathbf{\Gamma}$
- exception: None

EOMEA.lhs:

- output: $out := lhs \in \text{seq}(m, m : \mathbb{R})$
- exception: ValueError

EOMEA.rhs:

- output: $out := rhs \in \text{seq}(m, m : \mathbb{R})$
- exception: ValueError

12.4.5 Local Functions

`_compute_lhs()`:

- output: $out := \mathbf{h} - \mathbf{h}\boldsymbol{\gamma} + \sum_{ps} v_{mpns} \gamma_{ps} + 0.5 \sum_{pqs} v_{pqns} \Gamma_{pqsm}$
- exception: None

`_compute_rhs()`:

- output: $out := \mathbf{I} - \boldsymbol{\gamma}$, where \mathbf{I} represents the identity matrix
- exception: None

13 MIS of EOM Excitation Module

13.1 Template Module

EOMExc inherits EOMBase

13.2 Uses

EOMBase (10), Integrals (8), WfnRDMs (9)

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
new EOMExc	$\text{seq}(m, m : \mathbb{R}),$ $\text{seq}(m, m, m, m : \mathbb{R}),$ $\text{seq}(m, m : \mathbb{R}),$ $\text{seq}(m, m, m, m : \mathbb{R})$	EOMExc	-

13.4 Semantics

13.4.1 State Variables

nspino: \mathbb{Z}

h: $\text{seq}(m, m : \mathbb{R})$

v: $\text{seq}(m, m, m, m : \mathbb{R})$

γ : $\text{seq}(m, m : \mathbb{R})$

Γ : $\text{seq}(m, m, m, m : \mathbb{R})$

lhs: `_compute_lhs()`

rhs: `_compute_rhs()`

13.4.2 Environment Variables

None

13.4.3 Assumptions

The EOMExc constructor is called before any other access program in that class.

13.4.4 Access Routine Semantics

new EOMExc(h,v,dm1,dm2):

- transition: $\mathbf{h}, \mathbf{v}, \boldsymbol{\gamma}, \boldsymbol{\Gamma} := \mathbf{h}, \mathbf{v}, \mathbf{dm1}, \mathbf{dm2}$,
 $lhs := _compute_lhs()$,
 $rhs := _compute_rhs()$
 $nspino := |\mathbf{h}[0]|$
 $neigs := |\mathbf{h}[0]|$

- output: $out := self$

- exception: None

neigs():

- output: $out := |\mathbf{h}[0]|^2 \in \mathbb{Z}$
- exception: None

compute_tdm(c):

- output: $out := \sum_{ij} (\delta_{li} \gamma_{kj} - \Gamma_{kijl}) c_{ij}, \{(i, j) | (i \in [0..nspino - 1]) \wedge (j \in [0..nspino - 1])\}$
- exception: None

EOMExc.nspino:

- output: $out := nspino$
- exception: None

EOMExc.h:

- output: $out := \mathbf{h}$
- exception: None

EOMExc.v:

- output: $out := \mathbf{v}$
- exception: None

EOMExc.dm1:

- output: $out := \boldsymbol{\gamma}$
- exception: None

EOMExc.dm2:

- output: $out := \Gamma$
- exception: None

EOMExc.lhs:

- output: $out := lhs \in \text{seq}(m^2, m^2 : \mathbb{R})$
- exception: ValueError

EOMExc.rhs:

- output: $out := rhs \in \text{seq}(m^2, m^2 : \mathbb{R})$
- exception: ValueError

13.4.5 Local Functions

`_compute_lhs()`:

- output: $out :=$

$$h_{li}\gamma_{kj} + h_{jk}\gamma_{il} - \sum_q (h_{jq}\delta_{li}\gamma_{kq} + h_{qi}\delta_{jk}\gamma_{ql})$$

$$+ \sum_{qs} (v_{lqis}\Gamma_{kqjs} + v_{jqks}\Gamma_{iqls})$$

$$+ 0.5 \sum_{rs} (v_{jlrs}\Gamma_{kirs} + \sum_q v_{qjrs}\delta_{li}\Gamma_{kqrs})$$

$$+ 0.5 \sum_{pq} (v_{pqik}\Gamma_{pqlj} + \sum_s v_{pqsi}\delta_{jk}\Gamma_{pqsl})$$
- exception: None

`_compute_rhs()`:

- output: $out := \delta_{li}\gamma_{kj} - \Gamma$
- exception: None

14 MIS of EOM DIP Module

The MIS of EOM DIP is equivalent to the one for EOM Excitation (Section 13), therefore only the semantics of the methods that change will be declared.

14.1 Template Module

EOMDIP inherits EOMBase

14.2 Uses

EOMBase (10), Integrals (8), WfnRDMs (9)

14.3 Access Routine Semantics

`compute_tdm(c)`:

- output: $out := \sum_{ij} \Gamma_{klji} c_{ij}, \{(i, j) | (i \in [0..nspino - 1]) \wedge (j \in [0..nspino - 1])\}$
- exception: None

14.3.1 Local Functions

`_compute_lhs()`:

- output: $out :=$

$$2(h_{jk}\delta_{il} - h_{jl}\delta_{ik} + h_{ik}\gamma_{lj} - h_{il}\gamma_{kj})$$

$$+ 2 \sum_q h_{jq}(\delta_{ik}\gamma_{lq} - \delta_{il}\gamma_{kq}) + \mathbf{v}$$

$$+ 2 \sum_q v_{qjkl}\gamma_{qi} + \sum_r (v_{jilr}\gamma_{kr} - v_{jikr}\gamma_{lr})$$

$$+ 2 \sum_{qr} (v_{iqrk}\delta_{lj} + v_{iqlr}\delta_{kj})\gamma_{qr}$$

$$+ 2 \sum_{qr} (v_{jqrk}\Gamma_{qlri} + v_{jqlr}\Gamma_{qkri})$$

$$+ \sum_{qrs} v_{qjrs}(\delta_{ki}\Gamma_{qlrs} - \delta_{li}\Gamma_{qkrs})$$
- exception: None

`_compute_rhs()`:

- output: $out := 2\delta_{jk}\gamma_{li} + 2\delta_{il}\gamma_{kj} - 2\delta_{jk}\delta_{il}$
- exception: None

15 MIS of EOM DEA Module

The MIS of EOM DEA is equivalent to the one for EOM Excitation (Section 13), therefore only the methods that change are declared.

15.1 Template Module

EOMDEA inherits EOMBase

15.2 Uses

EOMBase (10), Integrals (8), WfnRDMs (9)

15.3 Access Routine Semantics

`compute_tdm(c)`:

- output: $out := \sum_{ij} (2\delta_{li}\delta_{kj} + 2\delta_{lj}\gamma_{ik} + 22\delta_{ki}\gamma_{jl} + \Gamma_{ijkl})c_{ij}, \{(i, j) | (i \in [0..nspino-1]) \wedge (j \in [0..nspino-1])\}$
- exception: None

15.3.1 Local Functions

`_compute_lhs()`:

- output: $out := 2(h_{li}\delta_{kj} - h_{ki}\delta_{lj} + h_{ki}\gamma_{jl} - h_{li}\gamma_{jk}) + 2\sum_p (h_{pi}\delta_{lj}\gamma_{pk} + h_{pj}\delta_{ki}\gamma_{pl}) + \mathbf{v} + 2\sum_r v_{lkjr}\gamma_{ir} + \sum_q (v_{qlij}\gamma_{qk} - v_{qkij}\gamma_{ql}) + 2\sum_{qr} (v_{qljr}\delta_{ki} - v_{qkjr}\delta_{li})\gamma_{qr} + 2\sum_{qr} (v_{qlir}\Gamma_{qjrk} - v_{qkir}\Gamma_{qjrl}) + \sum_{pqr} v_{pqjr}(\delta_{li}\Gamma_{pqrk} - \delta_{ki}\Gamma_{pqrl})$
- exception: None

`_compute_rhs()`:

- output: $out := 2\delta_{li}\delta_{kj} - 2\delta_{li}\gamma_{jk} - 2\delta_{kj}\gamma_{il}$
- exception: None

16 MIS of Output module

16.1 Module

output

16.2 Uses

input (7)

16.3 Syntax

16.3.1 Exported Constants

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
output	fname: ΔE :seq($k:\mathbb{R}$), c =seq($k,n:\mathbb{R}$), $\gamma_{n;0k}$ =seq($k,n,n:\mathbb{R}$)	str, -	-

16.4 Semantics

16.4.1 State Variables

None

16.4.2 Environment Variables

outputFile: A text file

16.4.3 Assumptions

16.4.4 Access Routine Semantics

output(fname, $\Delta E,c,\gamma_{n;0k}$):

- transition: Write to fname the input parameters from ParsedParams and the results of the calculations: ΔE , c and $\gamma_{n;0k}$
- exception: None

16.4.5 Local Functions

None

17 MIS of Solver Module

17.1 Module

solve

17.2 Uses

input (7)

17.3 Syntax

17.3.1 Exported Constants

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
dense	\mathbf{A} : $\text{seq}(k, k: \mathbb{R})$, $\text{seq}(k, k: \mathbb{R})$, $\mathbb{R} > 0$, orthog: str <i>in</i> {“ <i>symm</i> ”, “ <i>asymm</i> ”}	\mathbf{B} : $\Delta \mathbf{E}: \text{seq}(k: \mathbb{R})$, tol: $\mathbf{c} = \text{seq}(k, k: \mathbb{R})$	DivideByZero

17.4 Semantics

17.4.1 State Variables

17.4.2 Environment Variables

17.4.3 Assumptions

17.4.4 Access Routine Semantics

dense($\mathbf{A}, \mathbf{B}, \text{tol}, \text{orthog}$):

- output: $\text{out} := \Delta \mathbf{E}, \mathbf{c}$ that satisfies $\mathbf{A}c_i = \Delta E_i \mathbf{B}c_i, \{i | 0 \leq i \leq k\}$
- exception: DivideByZero

17.4.5 Local Functions

None

18 Appendix