SNEG - Mathematica package for symbolic calculations with second-quantization-operator expressions

Rok Žitko[a]

[a] *J. Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia*

Abstract

In many-particle problems involving interacting fermions or bosons, the most natural language for expressing the Hamiltonian, the observables, and the basis states is that of the second-quantization operators. It thus appears advantageous to write numerical computer codes which allow one to define the problem and the quantities of interest directly in terms of operator strings, rather than in some low-level programming language. Here I describe a Mathematica package which provides a flexible framework for performing the required symbolic calculations. It consists of a collection of transformation rules that define the algebra of operators and a comprehensive library of utility functions. While the emphasis is given on the problems from solid-state and atomic physics, the package can be easily adapted to any given problem involving non-commuting operators.

Key words: symbolic manipulation, second quantization operators, Wick's theorem, occupation number representation, bra-ket notation

PROGRAM SUMMARY

Manuscript Title: SNEG - Mathematica package for symbolic calculations with second-quantization-operator expressions
Author: Rok Žitko
Progam Title: SNEG
Journal Reference:
Catalogue identifier:
Licensing provisions: GNU Public License
Programming language: Mathematica
Computer: any computer which runs Mathematica
Operating system: any OS which runs Mathematica
RAM: problem dependent
Number of processors used: 1
Supplementary material:
Keywords: second quantization, non-commuting operators, commutators, Wick's theorem, Dirac bra-ket notation
Classification: 2.9 Theoretical methods, 5 Computer algebra, 6.2 Languages
External routines/libraries:
Subprograms used:
Nature of problem: Manipulation of expressions involving second quantization operators and other non-commuting objects. Calculation of commutators, anticommutators, expectation values. Generation of matrix representations of the Hamiltonians expressed in the second quantization language.
Solution method: Automatic reordering of operator strings in some well specified canonical order; (anti)commutation rules are used where needed. States may be represented in occupation-number representation. Dirac bra-ket notation may be intermixed with non-commuting operator expressions.

Email address: rok.zitko@ijs.si (Rok Žitko)

Restrictions: For long operator strings, the brute-force automatic reordering becomes slow, but it can be turned off. In such cases, the expectation values may still be evaluated using Wick's theorem.
Unusual features: SNEG also provides the natural notation of second-quantization operators (dagger for creation operators, etc.) when used interactively using the Mathematica notebook interface.
Additional comments:
Running time: problem dependent

1. Introduction

Computational science has emerged as the third paradigm of science, complementing experiments and theory. Computers are now used to simulate realistic physical systems which are not accessible to experiments or would be simply too expensive to study directly. They also allow numerical treatment of theoretical models which cannot be solved by analytical means nor by simple approximations. In this field, it is still common practice to quickly write ad-hoc computer codes for performing calculations for specific problems. In these rapidly developed computer programs the problem definition and the quantities of interest are typically hard-coded using the same low-level programming language which is also used to implement the method of solution. In more technical terms, the problem-domain and the solution-domain languages tend to coincide. As the scientific interests change with time, such codes often undergo successive modifications and adaptations, often leading to maintainability issues or even bugs. In software engineering, the proposed solution to such difficulties is to use a domain-specific language (DSL), i.e., a specification language adapted to a particular problem domain. Using a DSL, the problem can be expressed significantly more clearly than allowed by low-level languages. In the field of many-particle physics, such a language already exists: the language of strings of second-quantization operators (particle creation and annihilation operators) in terms of which it possible to express the problem (the Hamiltonian), the quantities of interest (the observables), and the domain of definition (the basis states defined by the creation operators applied to some vacuum state). Using an appropriate notation is equally important: the operators are usually expressed as single-character symbols, possibly with further indexes, and a dagger is used to distinguish creation from annihilation operators. The computer algebra system Mathematica makes it possible to both easily define the DSLs and to establish a suitable notation for these DSLs. In this article I describe package SNEG, which implements a DSL for second-quantization expressions and provides the corresponding natural notation. In addition to facilitating the representation of the input to numerical codes, the package is powerful enough to perform some calculations directly (e.g., evaluation of the expectation values using Wick's theorem, calculation and simplification of operator commutators, etc.).

This paper is structured as follows. Section 2 is devoted to the specification of basic elements (operators), their concatenation (non-commutative multiplication) and their automatic reordering (according to the canonical commutation/anticommutation rules or some other specification); it also introduces the Dirac bra-ket notation which can be mixed with the second-quantization operator expressions, and the occupation-number-representation vectors. Section 3 describes a number of higher-level routines for generating second-quantization expression (particle number and spin operators, etc.) and their manipulation (commutators and anticommutators, etc.). Section 4 details the utility routines for generating basis states which satisfy chosen symmetries (particle number conservation, rotational invariance in the spin space) as well as the routines for generating the matrix representations of operator expressions in given basis space; these routines are crucial for the applications of SNEG as an input preprocessor for lower-level numerical computer codes. The focus of Section 5 are symbolic sums with dummy indexes and their automated simplification using pattern matching. Finally, Section 6 describes a successful use of SNEG in the numerical renormalization group package "NRG Ljubljana".

The package SNEG comes with detailed documentation which integrates in the Mathematica interactive help system. Each SNEG function is carefully documented and examples are provided. For this reason, the syntax of function calls is not described in this article; instead, the focus here is on the basic concepts, design choices, conventions followed, and applications.

SNEG is released under the GNU Public License (GPL) and the most recently updated version is available from `http://nrgljubljana.ijs.si/sneg`. The package comes with a standard battery of test cases which may be used as a regression test, but also to verify that the possible user's custom extensions do not interfere with the expected behavior of the library.

While there are other packages for symbolic calculations with non-commuting objects for Mathematica and for other computer algebra systems (supercalc [1], ccr_car_algebra [2], NCComAlgebra [3], grassmann.m [4], grassmannOps.m [5]), none appears to have the scope or generality of SNEG. Furthermore, the goal of SNEG is different from more specialized symbolic manipulation packages such as TCE [6, 7] for performing many-body perturbation theory in quantum chemistry or FormCalc [8] for calculations in theoretical high-energy physics. Instead, SNEG is principally intended to provide a general framework in which more sophisticated solutions can be implemented or as a tool that provides a more natural interface to the user.

2. Foundations

The cornerstone of SNEG is a definition of non-commutative multiplication with automatic reordering of operators in some standard form (usually the conventional normal ordering with creation operators preceding the annihilation operators) which takes into account selected (anti)commutation rules. Use of the standard form reordering allows automatic simplifications of expressions. This section describes the low-level aspects of the library: objects representing operators and numbers (constants), non-commutative multiplication, expression reordering, vacuum state objects, occupation-number-representation objects, and support for the Dirac's bra-ket notation.

2.1. Operator objects and numeric objects

In SNEG, operators are represented as Mathematica expressions (lists) with a chosen head (typically a single-letter symbol) and containing the necessary indexes as list elements, for example

$$\texttt{a[]}, \quad \texttt{c[k, sigma]}. \tag{1}$$

The symbols need to be explicitly declared before they are used with one of the declaration routines: `snegbosonoperators`, `snegfermionoperators`, `snegmajoranaoperators`, or `snegspinoperators`. The declaration routines define the default (anti)commutation properties of the objects. They also establish the natural on-screen notation ("pretty-printing") when the package is used interactively using the Mathematica notebook interface. Both the (anti)commutation properties and the pretty-printing can be later modified according to the user's requirements. For operators declared to be bosons or fermions, the first element of the list (i.e., the first "index") has a special role: it distinguishes creation operators (`CR`=0) and annihilation operators (`AN`=1):

$$\texttt{c[CR, k]} \to c_k^\dagger, \quad \texttt{c[AN, k]} \to c_k. \tag{2}$$

In addition, for fermions, by default SNEG follows the convention that the last index is interpreted as the particle spin (`DO`=↓=0 and `UP`=↑=1); this convention is used when generating operator expressions using higher-level functions (see below) and when pretty-printing the expressions on computer display:

$$\texttt{c[CR, k, UP]} \to c_{k\uparrow}^\dagger, \quad \texttt{c[CR, k, DO]} \to c_{k\downarrow}^\dagger. \tag{3}$$

A similar convention is applied to operators that are declared to be spin operators: the last index denotes the spin component (`x`=1, `y`=2, `z`=3, `p`=4, `m`=5; the latter two denote the spin-raising and spin-lowering operators). The operator nature of operator objects can be tested using function `operatorQ`, and more specifically with `fermionQ`, `bosonQ`, `majoranaQ`, and `spinQ`.

Fermionic operators with different symbols are assumed to anticommute, bosonic operators with different symbols are assumed to commute, and bosonic and fermionic are assumed to commute. If necessary, this default behavior can be overridden.

SNEG allows to explicitly declare certain symbols to be numeric quantities of specific kinds; this information is used to correctly factor out numeric objects from the operator strings. The related functions

are `snegintegerconstants`, `snegrealconstants`, `snegcomplexconstants`, `sneggrassmanconstants`, and `snegfreeindexes`. The first four are self-explaining, while the last one declares indexes which are allowed to appear as dummy summation indexes in symbolic sum expressions; this is required to facilitate the automatic simplification of such expressions. The numeric nature of declared symbols can be tested using the function `isnumericQ`. The distinction between real and complex numeric quantities is used in the function `conj` to correctly conjugate an operator expression. Finally, it should be noted that the Grassman variables are correctly anticommuted and that $z^2 = 0$.

2.2. Non-commutative multiplication

In SNEG, the non-commutative multiplication is denoted by `nc`. In interactive sessions `nc` multiplications are pretty printed with centered dots between the terms, for example:

$$\texttt{nc}[\texttt{c}[\texttt{CR},\texttt{k},\texttt{UP}],\texttt{c}[\texttt{AN},\texttt{k},\texttt{UP}]] \rightarrow c^{\dagger}_{k\uparrow}\cdot c_{k\uparrow}. \tag{4}$$

The dot is displayed in order to permit easy detection of possible errors arising from an inadvertent replacement of non-commutative with the usual commutative multiplication. (In this article, the centered dot will be used for a similar purpose: to indicate places where noncommutitave multiplication is performed. This will only be done, however, where this is necessary for clarity or in order to avoid any possible confusion.) Function `nc` is linear in all its arguments, for example:

$$\texttt{nc}[\texttt{a}+\texttt{b},\texttt{c}] = \texttt{nc}[\texttt{a},\texttt{c}] + \texttt{nc}[\texttt{b},\texttt{c}], \quad \texttt{nc}[\texttt{z a},\texttt{b}] = \texttt{z nc}[\texttt{a},\texttt{b}], \tag{5}$$

where `z` is a numeric quantity, and it is associative:

$$\texttt{nc}[\texttt{nc}[\texttt{a},\texttt{b}],\texttt{c}] = \texttt{nc}[\texttt{a},\texttt{b},\texttt{c}]. \tag{6}$$

Furthermore, `nc[ ]`=1 and `nc[c]`=`c`; these two rules mimic the behavior of the standard Mathematica product function `Times` and imply the property of idempotency of multiplication. Note that `nc`, unlike `Times`, does not have the attributes `Flat` and `OneIdentity`. Instead, the associativity property is explicitly implemented. This design choice was motivated by reasons of efficiency in the pattern matching. Unlike `Times`, `nc` also does not have the attribute `Orderless`, for obvious reasons.

Finally, it should be mentioned that the exponentiation of an operator should be performed using the SNEG function `pow`, rather than with the Mathematica built-in `Power`, i.e. one should write `pow[a[],2]` rather than `a[]`$\hat{2}$.

2.3. Expression reordering

Computer algebra systems simplify expressions by ordering them in some canonical manner; in this way, the equivalent parts can be combined (or canceled out, if the prefactors sum to zero). This is how some simplifications are automatically effected in Mathematica. For this reason, SNEG also attempts to reorder multiplicands in the `nc` operator strings according to some canonical order, using the associated canonical anticommutation and commutation rules for the operators.

By default, fermionic operators are sorted canonically (in the sense that creation operators are permuted to the left and annihilation operators to the right – note that this simply corresponds to sorting according to the first index, `CR=0` or `AN=1`) and then by the remaining indexes, including spin as the last index. It has to be remarked, however, that the canonical order depends on the definition of the vacuum state. SNEG supports either an "empty band" vacuum with no particles present, or a "Fermi sea" vacuum with levels filled up to the Fermi level; this is controlled by setting the value of `ordering` to `EMPTY` or `SEA` for the specific symbol. The "Fermi sea" case can also be applied to atomic systems where one distinguishes between core, virtual and valence orbitals, and a quasi-vacuum state with occupied core orbitals is used.

The default ordering is that of an "empty band", thus the value of the first index (`CR` or `AN`) fully determines whether an operator is a creation or an annihilation operator. The type (annihilation vs. creation) of the operator can be tested using the functions `isannihilation` and `iscreation`. In the case of "Fermi sea"

ordering, the second index of the operator is tested by default. This index is assumed to be a "momentum" or "energy" index, with the Fermi sea fixed at zero. Thus

$$\begin{aligned} &\texttt{iscreation[c[CR, k]]} \to \texttt{True}, \quad k > 0, \\ &\texttt{iscreation[c[CR, k]]} \to \texttt{False}, \quad k < 0. \end{aligned} \tag{7}$$

If a different convention is required, the user has to redefine the functions `isannihilation` and `iscreation`. If necessary, it is also possible to turn off the automatic reordering for a fermionic operator by setting `ordering` to `NONE`. This is useful for very long expressions which can be simplified more efficiently by explicitly using Wick's theorem (see below) rather than by automatic operator reordering.

Internally, the operator ordering is tested with `snegOrderedQ`, while the necessary transformations on the operator strings (when out-of-order parts are detected) are implemented as transformation rules for the function `nc`.

By default, the canonical commutation relations (CCR) apply for bosonic operators:

$$[a_i, a_j] = 0, \quad [a_i^\dagger, a_j^\dagger] = 0, \quad [a_i, a_j^\dagger] = \delta_{i,j}, \tag{8}$$

while the canonical anti-commutation relations (CAR) apply for fermionic operators:

$$\{a_i, a_j\} = 0, \quad \{a_i^\dagger, a_j^\dagger\} = 0, \quad \{a_i, a_j^\dagger\} = \delta_{i,j}. \tag{9}$$

These can be overridden by redefining SNEG functions `cmt` and `acmt`, for example to generalize the relations to the case of non-orthonormal basis sets.

The Pauli rule for (complex) Dirac fermionic operators is implemented as a pattern which sets to zero all expressions in which two identical fermionic operators appear consecutively. For (real) Majorana fermionic operators, the default behavior is to replace pairs of identical consecutive operators by 1/2. It should be noted that $\psi^2 = 1/2$ is not the only convention and that, instead, some people define $\psi^2 = 1$.

Bosonic and spin operators are reordered according to the indexes; for bosons, this implies canonical ordering with the creation operators to the left of the annihilation operators. When operators of different types appear in the same product, they are disentangled. For example, bosonic operators are by default always commuted to the left of fermionic operators, Majorana fermionic operators are anti-commuted to the left of the Dirac fermionic operators, etc.

Finally, it should be recalled that automatic reordering is also implemented for the anti-commuting Grassman numbers if they are declared using `sneggrassmanconstants`.

2.4. Advanced automatic simplifications

SNEG will attempt to simplify expressions involving exponential functions of operators. Using the Baker-Hausdorff relations, one has for example

$$\begin{aligned} e^A e^B &\to e^{A+B} \quad \text{if } [A,B] = 0, \\ e^A e^B &\to e^{A+B} e^{[A,B]/2} \quad \text{if } [A,[A,B]] = 0 \text{ and } [B,[B,A]] = 0, \\ e^A B e^{-A} &\to B \quad \text{if } [A,B] = 0, \\ e^A B e^{-A} &\to B + [A,B] \quad \text{if } [A,[A,B]] = 0, \\ e^A B e^{-A} &\to e^c B \quad \text{if } [A,B] = cB \text{ and } c \text{ is numeric}, \\ e^A B e^{-A} &= \cos(i\sqrt{c})B + \left[\sin(i\sqrt{c})\right]/(i\sqrt{c})[A,B], \quad \text{if } [A,[A,B]] = cB \text{ and } c \text{ is numeric}. \end{aligned} \tag{10}$$

while using the Mendaš-Milutinović relations [9] one has

$$\begin{aligned} e^A B e^{-A} &\to B e^{-2A} \quad \text{if } \{A,B\} = 0, \\ e^A B e^{-A} &\to B e^{-2A} + \{A,B\} e^{-2A} \quad \text{if } \{A,\{A,B\}\} = 0. \end{aligned} \tag{11}$$

In addition

$$\begin{aligned} e^B &\to 1 + B \quad \text{if } c = B^2 = 0, \\ e^B &\to \cosh\sqrt{c} + B \sinh\sqrt{c}/\sqrt{c} \quad \text{if } c = B^2 \neq 0. \end{aligned} \tag{12}$$

2.5. Occupation-number representation

For fermionic operators, SNEG allows working with the occupation-number representation (ONR) of the states in a given Fock space. Second-quantized expressions can be applied to these states, one can compute the matrix elements of operators between pairs of states, etc.

The single-particle Hilbert space is defined by a call to function `makebasis`, which takes as the argument a list of all orbitals (without spin indexes, expansion over spin is performed automatically) that appear in the problem, for example

$$\texttt{makebasis}[\{\texttt{c}[1],\texttt{c}[2]\}] \to \left\{c^\dagger_{1,\uparrow}, c^\dagger_{1,\downarrow}, c^\dagger_{2,\uparrow}, c^\dagger_{2,\downarrow}\right\}. \tag{13}$$

The resulting list of all creation operators is stored in a global variable `BASIS`. It is equally possible to define `BASIS` directly, or to manipulate `BASIS` later on, for example to project out certain parts of the space. [The basis from Eq. (13) will be used throughout this subsection for all examples.]

The states in the ONR are expressed in the form of Mathematica lists ("vectors") with head `vc`, which contain the occupancies of all orbitals represented by zeros and ones. For example,`vc[0,1,0,1]` corresponds to a state with one particle with spin down in orbital 1, and one particle with spin down in orbital 2. In interactive Mathematica notebooks, the ONR vectors are shown in the Dirac-ket-like format with boxes which are either empty or filled, according to the occupancy of various orbitals:

$$\texttt{vc}[0,1,0,1] \to |\square\blacksquare\square\blacksquare\rangle\,. \tag{14}$$

If a vector is conjugated using `conj`, it behaves as the corresponding bra. If a bra and a ket are multiplied by `nc`, the corresponding scalar product is computed.

The vacuum state is generated by a call to SNEG function `vacuum`. Alternatively, one may use the nonspecific form, `VACUUM`, which corresponds to a generic state with no particles. By definition,

$$\texttt{nc}[\texttt{conj}[\texttt{VACUUM}],\texttt{VACUUM}] = 1. \tag{15}$$

Function `ap` can be used to apply a second-quantization operator (or a full operator expression) to a vector:

$$\begin{aligned}
\texttt{ap}[1,\texttt{VACUUM}] &\to \texttt{vc}[0,0,0,0],\\
\texttt{ap}[\texttt{c}[\texttt{CR},1,\texttt{UP}],\texttt{vacuum}[]] &\to \texttt{vc}[1,0,0,0],\\
\texttt{ap}[\texttt{c}[\texttt{CR},1,\texttt{UP}],\texttt{vc}[0,1,0,1]] &\to \texttt{vc}[1,1,0,1],\\
\texttt{ap}[\texttt{c}[\texttt{CR},1,\texttt{UP}],\texttt{vc}[1,0,0,0]] &\to 0.
\end{aligned} \tag{16}$$

It should be noted that

$$\begin{aligned}
\texttt{ap}[\texttt{nc}[\texttt{c}[\texttt{CR},1,\texttt{UP}],\texttt{c}[\texttt{CR},1,\texttt{DO}]],\texttt{VACUUM}] &\to \texttt{vc}[1,1,0,0],\\
\texttt{ap}[\texttt{nc}[\texttt{c}[\texttt{CR},1,\texttt{DO}],\texttt{c}[\texttt{CR},1,\texttt{UP}]],\texttt{VACUUM}] &\to -\texttt{vc}[1,1,0,0],
\end{aligned} \tag{17}$$

i.e., the fermionic sign is correctly evaluated. If one attempts to apply an operator which does not correspond to one of the orbitals in the single-particle basis, the function call is returned unevaluated.

In addition, `ap` is automatically called when an ONR vector appears as the last element of a non-commutative multiplication expression:

$$\texttt{nc}[\texttt{c}[\texttt{CR},1,\texttt{UP}],\texttt{vc}[0,0,0,0]] \to \texttt{vc}[1,0,0,0]. \tag{18}$$

It is possible to convert an ONR vector to the string of creation operators which, applied to the VACUUM state, would give back the same vector:

$$\texttt{vc2ops}[\texttt{vc}[1,1,0,0]] \to -c^\dagger_\downarrow c^\dagger_\uparrow. \tag{19}$$

2.6. Dirac's bra-ket notation

SNEG provides support for calculations with the Dirac bra-ket notation, which can be intermixed with the second-quantization expressions. This is convenient, for example, for mixed electron-phonon systems, where the fermions can be described using the second-quantization language, but the oscillator using some other convenient representation, such as coherent states. In mixed expressions, the bras and kets are by default always commuted to the right of the fermionic operators. In interactive Mathematica sessions, the bras and kets are displayed enclosed by appropriate angled brackets and bars.

A ket can be expressed using function `ket`, which can take one or several arguments (quantum numbers):

$$\begin{aligned} \texttt{ket[m]} &\rightarrow |m\rangle\,, \\ \texttt{ket[m, n]} &\rightarrow |m, n\rangle\,. \end{aligned} \tag{20}$$

Quantum numbers may also remain unspecified; this is signaled by the value `Null`; in interactive sessions it is displayed as a small centered circle:

$$\texttt{ket[m, Null, n]} \rightarrow |m, \circ, n\rangle\,. \tag{21}$$

This functionality can be used to multiply kets from orthogonal Hilbert spaces, as in the following example:

$$\texttt{nc[ket[m, Null], ket[Null, n]]} \rightarrow |m, n\rangle\,. \tag{22}$$

If the patterns of specified arguments are such that a conflict arises, as in `nc[ket[m,Null], ket[n,Null]]`, the expression is returned unevaluated.

All the preceding rules also apply to bras, defined using `bra`. With the Hermitian conjugation function `conj`, a bra can be transformed into ket and vice versa. It should be noted that the order of quantum numbers in the argument is maintained, rather than reversed:

$$\texttt{conj[ket[m, n]]} \rightarrow \langle m, n|\,. \tag{23}$$

When a bra and a ket are multiplied by `nc`, a scalar product is computed by comparing the quantum numbers in equivalent positions using the Kronecker delta:

$$\texttt{nc[bra[m, n], ket[i, j]]} \rightarrow \delta_{m,i}\delta_{n,j}, \tag{24}$$

If necessary, this default behavior can be altered by redefining the function `braketrule`. Unspecified arguments are ignored in evaluations of scalar products if they occur in equivalent positions in bra and ket, as in

$$\texttt{nc[bra[m, Null, n], ket[i, Null, j]]} \rightarrow \delta_{m,i}\delta_{n,j}, \tag{25}$$

otherwise the scalar product remains unevaluated

$$\texttt{nc[bra[m, Null], ket[i, j]]} \rightarrow \langle m, \circ|i, j\rangle. \tag{26}$$

In the bra-ket notation, the operators may be expressed as products of kets and bras, for example:

$$\texttt{nc[ket[1], bra[1]]} - \texttt{nc[ket[-1], bra[-1]]} \rightarrow |1\rangle\langle 1| - |-1\rangle\langle -1|\,. \tag{27}$$

It is possible to mix occupation-number-representation vectors and Dirac bra-ket vectors. The two subspaces are assumed to be unrelated (i.e., tensor product space). SNEG embeds the Dirac kets inside ONR vectors as the last element. When `ap` is used to apply an operator featuring both non-commuting operators and Dirac ket-bras, the operators are applied to the ONR part of the vector, and the ket-bras to the ket only.

2.6.1. Bra-ket notation for spin operators

SNEG includes functions for performing calculations with spin operators in the bra-ket notation. The spin kets and bras for an arbitrary spin S are generated using functions `spinket` and `spinbra`:

$$\texttt{spinket}[3/2] \rightarrow \{|3/2\rangle, |1/2\rangle, |-1/2\rangle, |-3/2\rangle\}. \tag{28}$$

If there is more than one Hilbert space, the position of the spin operator may be specified as the second argument to `spinket` and `spinbra` by a position list with number one indicating the chosen position, and zeros elsewhere:

$$\texttt{spinket}[1/2, \{0, 1, 0\}] \rightarrow \{|\circ, 1/2, \circ\rangle, |\circ, -1/2, \circ\rangle\}. \tag{29}$$

Spin operators may be generated using functions `spinketbra?`, where the question mark stands for X,Y,Z,P,M, i.e., X, Y, or Z component of spin, or the spin raising (Plus) or lowering (Minus) operator. The first argument is the spin S, while the second is the optional position list. For example

$$\begin{aligned}
\texttt{spinketbraX}[1/2] &\rightarrow \frac{1}{2}|-1/2\rangle\langle 1/2| + \frac{1}{2}|1/2\rangle\langle -1/2|, \\
\texttt{spinketbraY}[1/2] &\rightarrow \frac{i}{2}|-1/2\rangle\langle 1/2| - \frac{i}{2}|1/2\rangle\langle -1/2|, \\
\texttt{spinketbraZ}[1/2] &\rightarrow \frac{1}{2}|1/2\rangle\langle 1/2| - \frac{1}{2}|-1/2\rangle\langle -1/2|, \\
\texttt{spinketbraP}[1/2] &\rightarrow |1/2\rangle\langle -1/2|, \\
\texttt{spinketbraM}[1/2] &\rightarrow |-1/2\rangle\langle 1/2|.
\end{aligned} \tag{30}$$

For completeness, there is also a function for generating the identity operator, `spinketbraI`.

The spin-spin coupling (exchange interaction) operator $\boldsymbol{S}_1 \cdot \boldsymbol{S}_2$ can be generated using SNEG function `spinketbraspinspin`. As an example, to generate the Hamiltonian for a two-site Heisenberg model in external magnetic field, one could write

$$\begin{aligned}
\texttt{H} = \texttt{J}\ &\texttt{spinketbraspinspin}[\{1/2, 1/2\}] \\
&+\texttt{B}(\texttt{nc}[\texttt{spinketbraZ}[1/2, \{1, 0\}], \texttt{spinketbraI}[1/2, \{0, 1\}]] \\
&+\texttt{nc}[\texttt{spinketbraZ}[1/2, \{0, 1\}], \texttt{spinketbraI}[1/2, \{1, 0\}]]).
\end{aligned} \tag{31}$$

2.6.2. Bra-ket notation for phonon operators

For convenience, SNEG also includes a small number of functions for calculations with oscillators (phonons). Since oscillator states are unbounded, all functions take an argument which specifies the phonon cutoff `Nph`. A set of states up to `Nph` is generated, for example, as

$$\texttt{phononbasis}[2] \rightarrow \{|0\rangle, |1\rangle, |2\rangle\}. \tag{32}$$

The phonon number, raising, lowering, and displacement operators are generated as follows:

$$\begin{aligned}
\texttt{phononnumber}[2] &\rightarrow |1\rangle\langle 1| + 2|2\rangle\langle 2|, \\
\texttt{phononplus}[2] &\rightarrow |1\rangle\langle 0| + \sqrt{2}|2\rangle\langle 1|, \\
\texttt{phononminus}[2] &\rightarrow |0\rangle\langle 1| + \sqrt{2}|1\rangle\langle 2|, \\
\texttt{phononx}[2] &\rightarrow |1\rangle\langle 0| + |0\rangle\langle 1| + \sqrt{2}(|2\rangle\langle 1| + |1\rangle\langle 2|).
\end{aligned} \tag{33}$$

2.7. Miscellaneous

There is some rudimentary support for calculating Berezin integrals over Grassman numbers: `int[z]` stands for $\int \mathrm{d}z$ and one has, for example,

$$\texttt{nc}[\texttt{int}[\texttt{z1}]] = 0, \quad \texttt{nc}[\texttt{int}[\texttt{z1}], \texttt{z1}] = 1, \quad \texttt{nc}[\texttt{int}[\texttt{z1}, \texttt{z2}], \texttt{z1}, \texttt{z2}] = 1. \tag{34}$$

3. Generation of expressions and operations on expressions

In this section I describe the higher-level SNEG functions for generating various operators which can be expressed in terms of the second-quantization operators (occupancy, spin, on-site electron-electron repulsion, exchange coupling, etc.) and for performing various operations upon the expressions (calculation of the expectation values, Hermitian conjugation, etc.). In many of the application of SNEG, the library can be used at this higher level and the user does not need to be concerned with the inner working of the library at the lower levels.

3.1. Number (level occupancy) operator

The number (occupancy) operator $n = c^\dagger c$ can be generated with the function `number` which comes in different flavors depending on the function argument(s). The simplest case is the generation of the number operator corresponding to an operator pair $c^\dagger$, c using

$$\texttt{number[c]} \rightarrow \sum_\sigma c^\dagger_\sigma \cdot c_\sigma. \tag{35}$$

Note that the appropriate product of creation and annihilation operators is returned and that a summation of the spin degrees of freedom is performed. This behavior corresponds to fermionic operators declared to be spin-1/2 operators (this is the default behavior for operators declared using `snegfermionoperators`; see the documentiation of `spinof` for a way to redefine the spin of the operator). More generally, if additional indexes are required, one may use

$$\texttt{number[c[k]]} \rightarrow \sum_\sigma c^\dagger_{k,\sigma} \cdot c_{k,\sigma}. \tag{36}$$

If a single spin component is required, it may be specified as the second argument:

$$\texttt{number[c, UP]} \rightarrow c^\dagger_\uparrow \cdot c_\uparrow. \tag{37}$$

For bosonic operators, there is no sum over spin components:

$$\texttt{number[a]} \rightarrow a^\dagger \cdot a. \tag{38}$$

It is also possible to generate the number operator for more complex objects such as linear combinations of levels. As an example, if the even (bonding) orbital is defined to be $|e\rangle = \frac{1}{2}(|1\rangle + |2\rangle)$, the corresponding occupancy operator may be generated using an abstract-function argument to `number`:

$$\texttt{number[(c[\#1, 1, \#2] + c[\#1, 2, \#2])/Sqrt[2]\&]} \rightarrow \frac{1}{2}\sum_\sigma \left(c^\dagger_{1\sigma}c_{1\sigma} + c^\dagger_{1\sigma}c_{2\sigma} + c^\dagger_{2\sigma}c_{1\sigma} + c^\dagger_{2\sigma}c_{2\sigma}\right). \tag{39}$$

3.2. Hopping operator

The inter-site hopping operator may be generated using `hop`:

$$\texttt{hop[c[1], c[2]]} \rightarrow \sum_\sigma c^\dagger_{1,\sigma}c_{2,\sigma} + c^\dagger_{2,\sigma}c_{1,\sigma}. \tag{40}$$

If a single spin component is required, it may be specified as the third argument:

$$\texttt{hop[c[1], c[2], UP]} \rightarrow c^\dagger_{1,\uparrow}c_{2,\uparrow} + c^\dagger_{2,\uparrow}c_{1,\uparrow}. \tag{41}$$

Furthermore, there is a function `spinfliphop` which generates an operator that describes a process where the electron's spin is flipped as it hops between the sites, and a function `hopphi`, where a given phase change is associated with hopping (as in the Peierls substitution for describing the effect of a magnetic field piercing the plaquettes).

3.3. Hubbard's local electron-electron repulsion operator

The electron-electron repulsion operator $n_\downarrow n_\uparrow$ may be generated using the SNEG function `hubbard`:

$$\texttt{hubbard[c]} \to -c_\downarrow^\dagger c_\uparrow^\dagger c_\downarrow c_\uparrow. \tag{42}$$

Notice that due to the automatic operator reordering in SNEG, the creation operator $c_\uparrow^\dagger$ is anticommuted to the left of the annihilation operator $c_\downarrow$. There is also a closely associated function `isozsq`, defined as $(n-1)^2$ with $n = n_\uparrow + n_\downarrow$, i.e., the square of the isospin operator along the z-axis (see below for more details on the isospin operators).

For operators with spin different from 1/2, the definition of `hubbard` is

$$\texttt{hubbard[c]} \to \sum_{s_1=-S}^{S} \sum_{s_2=s_1+1}^{S} n_{s_1} n_{s_2}, \tag{43}$$

where S is the spin.

To describe inter-orbital and inter-site electron-electron charge repulsion, one can use function `chargecharge`:

$$\texttt{chargecharge[d[m], d[n]]} \to n_m n_n, \tag{44}$$

where n_i is the occupancy operator of orbital annihilated by the operator d_i.

3.4. Spin operators

Spin operator for orbitals described by fermionic operators can be generated by SNEG functions `snegx`, `snegy`, and `snegz`, which are defined for all values of particle spin (parameter `spinof`). For example:

$$\texttt{spinx[c]} \to \frac{1}{2}\left(c_\downarrow^\dagger c_\uparrow + c_\uparrow^\dagger c_\downarrow\right) \tag{45}$$

for a spin-1/2 operators (`spinof[c] ≙ 1/2`), and

$$\texttt{spinx[d]} \to \frac{\sqrt{3}}{2}\left(d_{-3/2}^\dagger d_{-1/2} + d_{-1/2}^\dagger d_{-3/2} + d_{1/2}^\dagger d_{3/2} + d_{3/2}^\dagger d_{1/2}\right) + d_{-1/2}^\dagger d_{1/2} + d_{1/2}^\dagger d_{-1/2} \tag{46}$$

for a spin-3/2 operators (`spinof[d] ≙ 3/2`). The spin raising and spin lowering operators can be generated with `spinplus` and `spinminus`, respectively, while the total spin squared operator S^2 can be obtained using `spinss`. It is also possible to generate all three Cartesian spin operators at the same time using `spinxyz`.

The spin operator generation functions use low-level routines for the generation of spin matrices (`spinmatrixX`, `spinmatrixY`, etc.) which are defined for any (half-)integer spin and may also be used to other purposes.

3.5. Exchange coupling operators

The exchange coupling (i.e., the scalar product of two spin operators, $\boldsymbol{S}_1 \cdot \boldsymbol{S}_2$) can be generated using `spinspin`:

$$\begin{aligned}\texttt{spinspin[c[1], c[2]]} \to \frac{1}{4}(&-c_{1,\downarrow}^\dagger c_{2,\downarrow}^\dagger\ c_{1,\downarrow} c_{2,\downarrow} + c_{1,\downarrow}^\dagger\ c_{2,\uparrow}^\dagger c_{1,\downarrow}\ c_{2,\uparrow} - 2c_{1,\downarrow}^\dagger c_{2,\uparrow}^\dagger\ c_{1,\uparrow} c_{2,\downarrow} - 2c_{1,\uparrow}^\dagger\ c_{2,\downarrow}^\dagger c_{1,\downarrow}\ c_{2,\uparrow} \\ &+ c_{1,\uparrow}^\dagger c_{2,\downarrow}^\dagger\ c_{1,\uparrow} c_{2,\downarrow} - c_{1,\uparrow}^\dagger\ c_{2,\uparrow}^\dagger c_{1,\uparrow} c_{2,\uparrow}).\end{aligned} \tag{47}$$

It is also possible to separately generate the transverse and the longitudinal parts (that is, xx+yy and zz components, respectively) of the spin coupling, as well as the $S_1^- S_2^+$ and $S_1^- S_2^+$ terms:

$$\begin{aligned}\texttt{spinspinxy[c[1], c[2]]} &\to \frac{1}{2}(-c_{1,\downarrow}^\dagger c_{2,\uparrow}^\dagger c_{1,\uparrow} c_{2,\downarrow} - c_{1,\uparrow}^\dagger\ c_{2,\downarrow}^\dagger c_{1,\downarrow} c_{2,\uparrow}), \\ \texttt{spinspinz[c[1], c[2]]} &\to \frac{1}{4}(-c_{1,\downarrow}^\dagger c_{2,\downarrow}^\dagger\ c_{1,\downarrow} c_{2,\downarrow} + c_{1,\downarrow}^\dagger\ c_{2,\uparrow}^\dagger c_{1,\downarrow}\ c_{2,\uparrow} + c_{1,\uparrow}^\dagger c_{2,\downarrow}^\dagger\ c_{1,\uparrow} c_{2,\downarrow} - c_{1,\uparrow}^\dagger\ c_{2,\uparrow}^\dagger c_{1,\uparrow} c_{2,\uparrow}), \\ \texttt{spinspinpm[c[1], c[2]]} &\to -c_{1,\uparrow}^\dagger c_{2,\downarrow}^\dagger c_{1,\downarrow}\ c_{2,\uparrow}, \\ \texttt{spinspinmp[c[1], c[2]]} &\to -c_{1,\uparrow}^\dagger c_{2,\downarrow}^\dagger c_{1,\downarrow}\ c_{2,\uparrow}.\end{aligned} \tag{48}$$

3.6. Nambu spinors and isospin operators

The Nambu spinor can be generated using `nambu`:

$$\begin{aligned} \texttt{nambu[c[i]]} &\to \begin{pmatrix} c^\dagger_{i,\uparrow} \\ c_{i,\downarrow} \end{pmatrix} \\ \texttt{nambu[c[i], i]} &\to \begin{pmatrix} c^\dagger_{i,\uparrow} \\ (-1)^i c_{i,\downarrow} \end{pmatrix} \end{aligned} \tag{49}$$

The parity of the optional second argument, which must be an integer, specifies the sign of the isospin-down component of the Nambu spinor. The isospin operators may be generated as

$$\begin{aligned} \texttt{isospinx[c]} &\to \frac{1}{2}(c_\downarrow c_\uparrow - c^\dagger_\downarrow c^\dagger_\uparrow), \\ \texttt{isospiny[c]} &\to \frac{1}{2}(ic^\dagger_\downarrow c^\dagger_\uparrow + ic_\downarrow\ c_\uparrow), \\ \texttt{isospinz[c]} &\to \frac{1}{2}(c^\dagger_\downarrow c_\downarrow + c^\dagger_\uparrow\ c_\uparrow - 1), \\ \texttt{isospinplus[c]} &\to -c^\dagger_\downarrow c^\dagger_\uparrow, \\ \texttt{isospinminus[c]} &\to c_\downarrow c_\uparrow. \end{aligned} \tag{50}$$

Like `nambu`, these five functions can take a sign-specifying second argument.

3.7. Vacuum expectation values

An important application area of SNEG is the computation of the vacuum expectation values (VEV) of second-quantization-operator strings. Two main functions are provided for this purpose, `vev` and `vevwick`.

To speed up the evaluations, a number of simplification rules are defined in SNEG. For example, terms with an odd number of operators in a string will always give zero when the VEV is calculated, thus such terms are immediately dropped. Furthermore, the VEV of a normal ordered expression is zero by definition. The terms where the right-most operator annihilates the vacuum (or the left-most operators annihilates the vacuum when applied to the left, i.e. to the bra) are also discarded. For the "Fermi sea" ordering, a similar rule is defined for the action of the two-operator occupancy expressions $n_k = c^\dagger_k c_k$ and $1 - n_k = 1 - c^\dagger_k c_k$ to the left and to the right, where the result depends on the value of the momentum index k. Finally, since it assumed that the vacuum state for the "Fermi sea" ordering has a well defined number of particles and magnetization, SNEG tests if the charge and spin quantum numbers are conserved; it not, such terms are dropped when the VEV is computed. The user can also provide additionnal simplification rules if further symmetries are known to be present in the problem being studied.

For the "empty band" ordering of fermionic operators, the VEV of an expression is trivially given by the possible remaining purely numeric term after all the operators in the strings had been fully ordered, since all other terms that involve some operators are (by the very definition of this ordering type) such that they annihilate the vacuum to the right. This is, in fact, the rationale for using this ordering type by default in SNEG.

For the "Fermi sea" ordering, the default rule for the VEV of bilinear forms is

$$\begin{aligned} \left\langle c^\dagger_{k,m} c_{k',m'} \right\rangle &\to \theta(-k)\delta_{k,k'}\delta_{m,m'}, \\ \left\langle c_{k,m} c^\dagger_{k',m'} \right\rangle &\to \theta(k)\delta_{k,k'}\delta_{m,m'}. \end{aligned} \tag{51}$$

3.8. Other applications of Wick's theorem

The function `wick` may be used to rewrite an expression in terms of normal ordered products and contractions using Wick's theorem [10]. Contraction are evaluated using `contraction`, which is defined as $\overline{AB} = AB - :AB:$, where double dots denote the normal ordering. The results depend, of course, on the

vacuum state, i.e., on the chosen ordering of the fermionic operators (see above). For "Fermi sea ordering", we have, for example:

$$\begin{aligned}
\texttt{wick}[\texttt{nc}[\texttt{c}[\texttt{CR},\texttt{k}],\texttt{c}[\texttt{AN},\texttt{k}]]] &\rightarrow : c_k^\dagger c_k : + \theta(-k), \\
\texttt{wick}[\texttt{nc}[\texttt{c}[\texttt{CR},\texttt{k}],\texttt{c}[\texttt{CR},\texttt{l}],\texttt{c}[\texttt{AN},\texttt{m}],\texttt{c}[\texttt{AN},\texttt{n}]]] &\rightarrow : c_k^\dagger c_l^\dagger c_m c_n : - : c_l^\dagger c_n : \delta_{km}\theta(-k) \\
&+ : c_l^\dagger c_m : \delta_{kn}\theta(-k) + : c_k^\dagger c_n : \delta_{lm}\theta(-l) - c_k^\dagger c_m : \delta_{ln}\theta(-l) + \delta_{kn}\delta_{lm}\theta(-k)\theta(-l) - \delta_{km}\delta_{ln}\theta(-k)\theta(-l).
\end{aligned} \tag{52}$$

3.9. Conjugation

Hermitian conjugates of operator strings can be computed using the function `conj`. The numerical constants and parameters are handled correctly depending on their nature (real or complex commuting numbers, or Grassman anticommuting numbers). For complex fermionic and bosonic operator objects, the first index is modified (creation to annihilation, and vice versa), while real fermionic operator objects are left unchanged. Exponential functions is conjugated according to the rule

$$\left(e^A\right)^\dagger = e^{A^\dagger}. \tag{53}$$

SNEG also performs the simplification

$$\left(A^\dagger\right)^\dagger = A. \tag{54}$$

This rule is usually valid, since a continuous linear operator defined on all of a Hilbert space has a unique adjoint operator, however the rule does not hold for operators that are defined on a subset of a Hilbert space and for unbounded operators. In the case of finite-dimensional Hilbert space, which are the main application area of SNEG, there is no difference between adjoint operators and Hermitian conjugate operators, thus there are no reasons for concern and the repeated conjugation may be automatically eliminated. These issues are discussed in depth in Ref. [11].

3.10. (Anti)commutators

Commutators and anticommutators can be computed trivially by forming the sums or differences of the products, or with the help of the provided auxiliary functions `commutator` and `anticommutator`. To obtain the final results, it is often necessary to expand and recollect the common terms, for example by using the `Simplify` function. Nested (anti)commutators can be computed using `supercommutator` and `superanticommutator`; for efficiency, at each order of the iterative calculation the expression is simplified to avoid redundant computations.

For example:

$$\begin{aligned}
\texttt{commutator}[\texttt{spinx}[\texttt{c}],\texttt{spiny}[\texttt{c}]] &\rightarrow i\frac{1}{2}\left(c_\uparrow^\dagger c_\uparrow - c_\downarrow^\dagger c_\downarrow\right), \\
\texttt{supercommutator}[\texttt{spinx}[\texttt{c}],\texttt{spiny}[\texttt{c}],2] &\rightarrow \frac{1}{2}\left(ic_\downarrow^\dagger c_\uparrow - ic_\uparrow^\dagger c_\downarrow\right).
\end{aligned} \tag{55}$$

3.11. Normal ordering

Expressions can be "normal ordered" by subtracting their vacuum expectation values. Two functions are provided for this purpose: `normalorder` and `normalorderwick`. As indicated by the name, they differ in how the vacuum expectation value is computed (function `vev` or `vevwick`, see above). For convenience, SNEG also allows to mark expressions as "normal ordered" without actually performing the evaluation. The conventional double dot notation is used for this purpose; to surround a (sub)expression with double dots, `dd` has to be applied to the string:

$$\texttt{dd}[\texttt{nc}[\texttt{c}[\texttt{CR}],\texttt{c}[\texttt{AN}]]] \rightarrow : c^\dagger c : . \tag{56}$$

3.12. Operations with states

As discussed previously, the states can be represented in SNEG either as strings of second-quantization operators (implicitly applied to a vacuum state in which no particles are present) or as occupation-number-representation vectors. To compute a matrix element of an operator between two states, $\langle a| O |b\rangle$ one can use SNEG functions `braketop` or `braketvc`, depending on the representation used. To compute diagonal matrix elements (i.e., the expectation values), one can also use functions `expvop` and `expvvc`. For example, the expression

$$\texttt{expvop[spinss[c], c[CR, UP]]} \to 3/4 \tag{57}$$

calculates the expectation value of the spin squared operator $\boldsymbol{S}^2$ in the doublet state $c_\uparrow^\dagger |0\rangle$, i.e., $S(S+1) = 3/4$. Finally, there are two "norm" functions that compute $\sqrt{\langle\psi|\psi\rangle}$, `normop` and `normvc`; these are useful to generate (ortho)normalized sets of states.

The braket calculations can be generalized to lists of states to generate matrix representations of operators; the corresponding functions are named `matrixrepresentationvc` and `matrixrepresentationop`.

A state may be decomposed into components with respect to a given basis (list of states) using `decomposevc` and `decomposeop`. Furthermore, a list of states may be orthogonalized using `orthogvc` and `orthogop`.

3.13. Projection operators

A number of projection operators can be generated by SNEG:

$$\begin{aligned}
\texttt{projector0[c]} &\to (1-n_\uparrow)(1-n_\downarrow),\\
\texttt{projectorUP[c]} &\to n_\uparrow(1-n_\downarrow),\\
\texttt{projectorDO[c]} &\to n_\downarrow(1-n_\uparrow),\\
\texttt{projector2[c]} &\to n_\uparrow n_\downarrow,\\
\texttt{projector1[c]} &\to n_\uparrow(1-n_\downarrow) + n_\downarrow(1-n_\uparrow),\\
\texttt{projector02[c]} &\to (1-n_\uparrow)(1-n_\downarrow) + n_\uparrow n_\downarrow.
\end{aligned} \tag{58}$$

In calculations involving projection operators, it is often useful to simplify expression by dropping terms which would yield zero when applied to a vacuum state; the SNEG function to achieve this is called `zeroonvac`. For example:

$$\begin{aligned}
\texttt{nc[projector1[c], c[CR, UP]]} &\to c_\uparrow^\dagger + c_\downarrow^\dagger c_\uparrow^\dagger c_\downarrow,\\
\texttt{zeroonvac[nc[projector1[c], c[CR, UP]]]} &\to c_\uparrow^\dagger.
\end{aligned} \tag{59}$$

3.14. Simplification of expressions

Previously in this section I have described a number of function for generating operators in the form of strings of second-quantization operators. Often it is necessary to proceed in the opposite direction: given a long complex expression, one has to rewrite it in terms of higher-level functions, such as number, hopping, repulsion, or spin operators. This might be used, for example, after performing a change-of-basis transformation on the creation and annihilation operators, if one requires a physical interpretation of the resulting expression.

There is, clearly, no unique mapping from an expression to the corresponding generation functions. Therefore, there are several specialized routines which apply heuristic rules in an attempt to rewrite the expression: `SnegSimplifyNumber` identifies parts of the expression which may be rewritten in terms of `number`, `SnegSimplifyHubbard` does the same for `hubbard` and `SnegSimplifyHop` for `hop`, while `SnegSimplifySpin` will find instances of `spinx`, `spiny`, `spinz` and some of their products (including `spinspin` expressions). The whole set of the rules can also be applied by `SnegSimplify` and `SnegFullSimplify`, although experience shows that such brute-force approach is not very efficient and that a guided consecutive application of suitably chosen specialized routines gives better results.

3.15. Miscellaneous

Operators can be raised to the n-th power using the function `pow`, for example

$$\texttt{pow[number[a],3]} \rightarrow a^\dagger a + 3(a^\dagger)^2 a^2 + (a^\dagger)^3 a^3, \tag{60}$$

for a bosonic operator a. There is an additional function, `fastpow`, which performs a simplification of the intermediate expression at each step of the iterative calculation.

Series expansions of functions with operator arguments can be performed using `snegSeries`. For example,

$$\texttt{snegSeries[Exp,nc[c[CR],c[AN]],10]} \rightarrow 1 + \frac{6235301}{3628800} c_\uparrow^\dagger c, \tag{61}$$

which is a good approximation to $e^{c^\dagger c} = 1 + (e-1)c^\dagger c$.

For convenience, the inner and outer products of vectors of operator expressions can be calculated using the SNEG functions `inner` and `outer`, which are the analogues of the Mathematica functions `Inner` and `Outer` with `nc` playing the role of multiplication. For example, the function `spinspin` (see above) for calculating $\boldsymbol{S}_1 \cdot \boldsymbol{S}_2$ is implemented as `inner[spinxyz[c[1]],spinxyz[c[2]]]`. Furthermore, function `VMV` can be used to calculate the vector-matrix-vector products using the `nc` multiplication. This function is very useful for generating operator expressions and it is frequently used internally in the package SNEG. Finally, the cross product between two vectors using the `nc` multiplication can be calculated using `mcross`. For example, $\boldsymbol{S}_1 \times \boldsymbol{S}_2$ can be generated using `mcross[spinxyz[c[1]], spinxyz[c[2]]]`.

Function `SimplifyKD` can be used to simplify expressions involving Kronecker delta function (`KroneckerDelta`) and the unit step function (`UnitStep`). The following rules are implemented:

$$\begin{aligned}
\theta(k) + \theta(-k) &\rightarrow 1, \\
1 - \theta(-k) &\rightarrow \theta(k), \\
\theta(k)\theta(-k) &\rightarrow 0, \\
\delta_{k,l}\theta(k)\theta(-l) &\rightarrow 0, \\
(\delta_{i,j})^n &\rightarrow \delta_{i,j}, \quad n \in \mathcal{N}, \\
\theta(k)^n &\rightarrow \theta(k), \quad n \in \mathcal{N}.
\end{aligned} \tag{62}$$

(See below for simplification of symbolic sum expressions featuring Kronecker deltas and unit steps.)

4. Generation of sets of basis states

One of the principal application areas of SNEG is the transformation of the operators expressed in the second-quantized notation into the corresponding matrix representation in a given Hilbert space. To simplify numerics, it is often important to take into account various symmetries of the problem, i.e., to determine the Hamiltonian matrices in the different invariant Hilbert subspaces. SNEG provides a number of functions for generating the basis-state sets with chosen well-defined quantum numbers (charge conservation, etc.). In this section we describe both the low-level functions for basis-set manipulation, as well as the higher-level routines which provide "canned solution" for a number of different symmetry-adapted basis sets. These are convenvient for calculations defined on finite clusters, such as small Hubbard chains.

4.1. Full Fock space basis set representation

The basis-state set that spans the full Fock space is represented in SNEG as a list of pairs – the first member of each pair is a list of quantum numbers which fully characterizes the invariant subspace, while the second member of the pair is a list of all the basis states in the given subspace. The basis states can be specified either as operator expressions (which are to be applied on the vacuum state) or as occupation-number-representation (ONR) vectors. An example of a basis with well-defined charge and spin for a single-orbital problem is

$$\left\{ \quad \{\{-1,0\},\{1\}\}, \quad \left\{\{0,1/2\}, \left\{c_\uparrow^\dagger\right\}\right\}, \quad \left\{\{1,0\}, \left\{c_\downarrow^\dagger c_\uparrow^\dagger\right\}\right\} \quad \right\}. \tag{63}$$

Table 1: Basis-set-generation functions in SNEG. Q stands for the total occupancy (charge) quantum number, S_z is the projection of total spin along the z-axis, while S is the total spin quantum number. Function `spinlessbasis` generates states for a problem without spin, while all other functions pertain to spin-1/2 problems.

Function	Conserved quantum numbers
`nonebasis`	$-$
`qbasis`	Q
`szbasis`	S_z
`sbasis`	S
`qszbasis`	Q, S_z
`qsbasis`	Q, S
`spinlessbasis`	Q

The first quantum number is the charge (with respect to half filling), while the second is the spin. If the states in a subspace have multiplicity higher than 1, only a single representative state appears for each multiplet. In the last example, the spin doublet is represented, e.g., by $c^\dagger_\uparrow$.

It is possible to transform from the creation-operator representation to the occupation-number representation and back using functions `bzop2bzvc` and `bzvc2bzop`. These functions can be used to perform the manipulations of basis states in the representation where the implementation is easier and/or the calculation is faster.

There is a number of functions for manipulating the sets of states: `basistensorproduct` combines two basis sets by generatic all possible combinations (i.e., a tensor product basis), `transformbasis` applies a given rule (a transformation) to each state, `mergebasis` can be used to collect states in subspaces with equivalent quantum numbers, and `dropemptysubspaces` truncates subspaces containing no states (which may arise from projecting out states according to some chosen rule).

4.2. High-level basis generation functions

SNEG functions for generation of basis sets are tabulated in Table 1. These functions generate basis sets in the creation-operator representation. There are also corresponding functions which generate basis sets in the occupation-number representation; their names are suffixed by `vc`.

From the basis sets for the cases with conserved charge (Q) quantum number, one can generate basis sets with conserved isospin quantum number I using `transformQStoIS`.

In addition, there is support for generating basis states for calculations with spin objects in the bra-ket notation. The relevant SNEG function is named `spinbasis`.

4.3. Generation of operator matrices

Once the desired basis sets have been generated, the operators in the second-quantization language can be transformed into the corresponding matrix representations using the functions `makematricesbzop` and `makematricesbzvc`. It should be remarked that the latter function is typically much faster and, in fact, it is often advantageous to transform the basis set from the creation-operator to the occupation-number representation in order to use it. These functions generate matrices only within the invariant subspaces, i.e., it is assumed that the operator in question commutes with the Hamiltonian. There are also more general functions, `makeallmatricesbzop` and `makeallmatricesbzvc`, which generate matrices between all paris of invariant subspaces.

5. Symbolic sums

SNEG allows calculations with symbolic sums over dummy indexes, which remain in their unevaluated forms. They are defined using the function `sum` taking two arguments: the first one is the expression that

is being summed over, while the second one is the list of all summation indexes. For example

$$\begin{aligned} \texttt{sum[c[CR,k],\{k\}]} &\to \sum_k c^\dagger_k, \\ \texttt{sum[nc[c[CR,k,sigma],c[AN,k,sigma]],\{k,sigma\}]} &\to \sum_{k,\sigma} c^\dagger_{k,\sigma} c_{k,\sigma}. \end{aligned} \tag{64}$$

The list of indexes is automatically sorted; this allows some automatic simplifications. Numeric quantities which do not depend on any of the summation indexes are factored out of the `sum` expression. While not strictly necessary, it is good practice to declare the symbols which will be used as summation indexes using `snegfreeindexes`.

A comment is in order about the design choice to introduce a new symbolic sum function `sum` instead of using the Mathematica built-in `Sum`, which can also be used to define indefinite sums as `Sum[f,i]`. The reason is that Mathematica `Sum` attempts to evaluate indefinite summations where it can, while `sum` is merely a notation for symbolic sums which remain unevaluated at all times. This is especially important in situations where the operator expression can be simplified to a constant value x. While `Sum[x,k,sigma]` evaluates to $xk\sigma$, which is meaningless, `sum[x,{k,sigma}]` evaluates to $x\sum_{k,\sigma} 1$, which can be be naturally interpreted as the number of states, including the spin degeneracy, multipled by a constant x.

5.1. Operations on symbolic sums

Products of sums can be calculated using `nc`. SNEG automatically handles summation index collisions and renames the duplicated indexes by appending a number to the name symbol (and automatically declaring the new name using `snegfreeindexes`). It is thus perfectly safe to use the same dummy index in different expressions:

$$\texttt{nc[sum[number[c[k],UP],\{k\}],sum[number[c[k],DO],\{k\}]]} \to -\texttt{sum}[c^\dagger_{k,\uparrow} c^\dagger_{k1,\downarrow} c_{k,\uparrow} c_{k1,\downarrow}, \{\texttt{k},\texttt{k1}\}]. \tag{65}$$

When commutators of sums are computed, the name replacement is always performed on the same sum in order to maximize the opportunities for automatic cancellation of equal terms.

Conjugation (`conj`) and VEV operation (`vev` and `vevwick`) can be applied to a symbolic sum:

$$\texttt{vevwick[expr]} \to \texttt{sum}[\theta(-k)\theta(-k1), \{\texttt{k},\texttt{k1}\}], \tag{66}$$

where `expr` is the expression in Eq. (65).

5.2. Manipulation of symbolic sum expressions

There is a number of SNEG functions for symbolic manipulation of sum expressions. Lists of expressions in the first argument can be "threaded" over using `sumThread`:

$$\texttt{sumThread[sum[c[CR,k,UP],c[CR,k,DO],\{k\}]]} \to \left\{ \sum_k c^\dagger_{k\uparrow}, \sum_k c^\dagger_{k\downarrow} \right\}. \tag{67}$$

Explicit sums in the first argument can be expanded using `sumExpand`:

$$\texttt{sumExpand[sum[a+b,\{k\}]]} \to \sum_k a + \sum_k b, \tag{68}$$

and expression with (partially) overlapping index lists can be collected using `sumCollect`:

$$\begin{aligned} \texttt{sumCollect[2sum[a,\{k\}]+3sum[b,\{k\}]]} &\to \sum_k 2a + 3b, \\ \texttt{sumCollect[2sum[a,\{k,m\}]+3sum[b,\{k,n\}]]} &\to \sum_k \left(\sum_m 2a + \sum_n 3b \right). \end{aligned} \tag{69}$$

It should be also noted that `sumCollect` puts possible leading or trailing operators inside the summation argument:

$$\texttt{sumCollect[nc[a, sum[b, \{k\}], c]]} \rightarrow \sum_k a \cdot b \cdot c. \tag{70}$$

For convenience, a set of rules for simplifying expressions featuring Kronecker delta δ_{n_1,n_2} are defined as `rulesumSimplifyKD`; they can be applied using `sumSimplifyKD`. They handle situations where the Kronecker delta picks out elements from the sum:

$$\texttt{sumSimplifyKD[sum[KroneckerDelta[sigma, UP]c[CR, sigma], \{sigma\}]]} \rightarrow c^\dagger_\uparrow, \tag{71}$$

and work even for more complex expressions. Furthermore, products and sums over non-overlapping index sets are automatically factored/distributed out, for example:

$$\texttt{sumSimplifyKD[sum[a[k] b[l], \{k, l\}]]} \rightarrow \sum_k a[k] \sum_l b[l], \tag{72}$$

where a is assumed to depend only on k, and b only on l.

Finally, expressions with symbolic sums can be automatically simplified using `sumSimplify` which uses Mathematica function `Simplify` with additional transformation functions for expanding, simplifying, and collecting the terms.

6. Applications

SNEG has found many applications in the field of theoretical condensed matter physics. It has been applied to perform exact diagonalizations on Hubbard clusters, small Heisenberg chains and similar lattice models, calculation of commutators of complex operator expressions (to establish the presence of various symmetries, in the equation of motion method, etc.), and perturbation theory to higher orders. It is best suited for problems where the complexity is too high for paper and pencil calculations, yet still sufficiently low for a brute-force computer algebra approach (which SNEG essentially is). The package makes otherwise tedious calculations a routine operation. Most importantly, it prevents inauspicious sign errors which commonly arrise when fermionic operators are (anti)commuted. For this reason, the package is also suitable for educational purposes, i.e., as a way of verifying the correctness of elementary calculations with operator quantities.

The main major application of SNEG in its role of an "interface" between the user and the low-level numerical codes is the package "NRG Ljubljana" for performing the numerical renormalization group (NRG) calculations for quantum impurity models [12, 13, 14]. Using SNEG as the underlying library, both the model (Hamiltonian) and the observables (operators) may be defined in terms of high-level expressions. This enabled a clear separation between the problem domain (coded in Mathematica and SNEG) and the solution domain (coded in C++). This is advantageous not only for reasons of performance, but especially for maintainability of the code. During the lifetime of the project, no major rewrites or design changes were necessary in either part of the code and the development could proceed incrementally without breaking the existing features. Furthermore, adapting the package to different problems and symmetries, or to calculate new quantities, is rather trivial.

7. Conclusion

The natural language of many-particle physics is the quantum field theory, more particularly the formalism of the second quantization. I have argued that the computational many-body physics should strive towards creating computer codes which would allow defining problems – whenever possible – in their natural problem-domain language. It is hoped that the approach (and the specific implementation, SNEG) will improve the productivity of users and the quality of scientific software (in particular reliability, reusability, maintainability, and correctness).

References

[1] Werner M. Seiler. Supercalc - a reduce package for commutator calculations. *Comput. Phys. Commun.*, 66:363, 1991.

[2] Georg Lang. ccr_car_algebra. http://library.wolfram.com/infocenter/TechNotes/4269/, 2002.

[3] Nam-Anh Nguyen and T. T. Nguyen-Dang. Symbolic calculations of unitary transformations in quantum dynamics. *Comput. Phys. Commun.*, 115:183, 1998.

[4] M. Headrick. grassmann.m. http://people.brandeis.edu/ headrick/physics/grassmann.m, 2009.

[5] J. Michelson. grassmannops.m : A mathematica package for grassmann and other non-commuting numbers and operators. http://www.physics.ohio-state.edu/ jeremy/mathematica/grassmannOps/, 2007.

[6] So Hirata. Tensor contraction engine: Abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *J. Phys. Chem. A*, 107:9887, 2003.

[7] So Hirata. Symbolic algebra in quantum chemistry. *Theor. Chem. Acc.*, 116:2, 2006.

[8] T. Hahn and M. Pérez-Victoria. Automated one-loop calculations in four and d dimensions. *Comput. Phys. Commun.*, 118:153, 1999.

[9] I. Mendaš and P. Milutinović. Anticommutator analogue of the baker-hausdorff lemma. *J. Phys. A: Math. Gen.*, 22:L687, 1989.

[10] G. C. Wick. The evaluation of the collision matrix. *Phys. Rev.*, 80:268, 1950.

[11] Francois Gieres. Mathematical surprises and dirac's formalism in quantum mechanics. quant-ph/990069, 1999.

[12] K. G. Wilson. The renormalization group: Critical phenomena and the kondo problem. *Rev. Mod. Phys.*, 47:773, 1975.

[13] H. R. Krishna-murthy, J. W. Wilkins, and K. G. Wilson. Renormalization-group approach to the anderson model of dilute magnetic alloys. i. static properties for the symmetric case. *Phys. Rev. B*, 21:1003, 1980.

[14] Ralf Bulla, Theo Costi, and Thomas Pruschke. The numerical renormalization group method for quantum impurity systems. *Rev. Mod. Phys.*, 80:395, 2008.