The `FermiFab` toolbox for fermionic many-particle quantum systems ☆

Christian B. Mendl

Center for Mathematics M7, TU München, Germany

ARTICLE INFO

ABSTRACT

This paper introduces the `FermiFab` toolbox for many-particle quantum systems. It is mainly concerned with the representation of (symbolic) fermionic wavefunctions and the calculation of corresponding reduced density matrices (RDMs). The toolbox transparently handles the inherent antisymmetrization of wavefunctions and incorporates the creation/annihilation formalism. Thus, it aims at providing a solid base for a broad audience to use fermionic wavefunctions with the same ease as matrices in Matlab, say. Leveraging symbolic computation, the toolbox can greatly simply tedious pen-and-paper calculations for concrete quantum mechanical systems, and serves as "sandbox" for theoretical hypothesis testing. `FermiFab` (including full source code) is freely available as a plugin for both Matlab and Mathematica.

Program summary

1. Introduction

The ground state energy of fermionic many-particle quantum systems can be re-expressed as a linear functional of (one- or two-body) reduced density matrices (RDMs). This notion traces back to the origins of quantum mechanics [1,2] around 1930. Since 1964, the one-body RDM has been greatly popularized by density functional theory [3,4], which is typically the most viable approximation for handling large particle numbers. The tantalizing possibility of employing RDMs (instead of many-particle wavefunctions) for *exact* groundstate energy computations is counterbalanced by the *N*-representability problem, i.e., the search for necessary and sufficient conditions a two-body density must obey to represent an *N*-electron wavefunction [5–7]. Modern applications use variational principles and semidefinite

---

Fig. 1. Schematic illustration of a Slater determinant: (filled) circles correspond to (occupied) orbitals.

programming to impose positivity constraints on the two-body RDM [8]. In any case, it is desirable to render the powerful RDM framework accessible to a broader audience, integrating it into the symbolic language of modern computer algebra systems like Mathematica, or numeric software like Matlab.

The `FermiFab` toolbox (available for download at [9]) is precisely designed for that purpose. A short "usage manual" and a brief tour of the essential features are provided in the following subsections. Note that the underlying one-particle orbitals (see below) are always assumed to be orthonormalized. In addition, the toolbox adheres to the trace-normalization convention $\mathrm{tr}_{\bigwedge^p \mathcal{H}} \gamma_{|\psi\rangle\langle\psi|} = \binom{N}{p}$ for the p-body RDM $\gamma_{|\psi\rangle\langle\psi|}$ of a normalized N-body wavefunction ψ. Here, $\bigwedge^p \mathcal{H}$ denotes the p-particle Fock-space (see following subsection).

1.1. Fermi states

Fundamental building blocks of multi-fermion quantum systems are Slater determinants (Fig. 1). These can be thought of as a collection of "orbitals" (or slots), some of which are occupied by a fermionic particle (e.g., an electron). In mathematical terms, the available number of orbitals 'orbs' is the dimension of the underlying one-particle Hilbert space $\mathcal{H}$ and the number of occupied orbitals the particle number N. Thus there are altogether $\binom{\text{orbs}}{N}$ Slater determinants. Their complex span defines the N-particle *Fock-space* $\bigwedge^N \mathcal{H}$. The N-particle Fermi states are precisely the elements of $\bigwedge^N \mathcal{H}$.

1.2. Getting started with `FermiFab`

For concreteness, the following examples are issued in the Matlab programming language. (The Mathematica version of `FermiFab` provides the same features; Section 3 contains a demonstration.) Commands typed by the user are preceded by >>, and the subsequent lines show the corresponding output. In standard Matlab syntax, `zeros(n,1)` below constructs a column vector of length n, and `nchoosek` computes binomial coefficients. We first show how to represent an $N = 4$ particle state ψ with, e.g., 6 available orbitals in total:

```
>> orbs = 6; N = 4;
>> x = zeros(nchoosek(orbs,N),1); x(1)=1/sqrt(2); x(2)=1i/sqrt(2);
>> psi = fermistate(orbs,N,x)

psi =
    Fermi State (orbs == 6, N == 4)
    (0.70711)|1234> + (0+0.70711i)|1235>
```

Needless to say, the `fermistate` command is specific to the `FermiFab` toolbox. The vector x contains the Slater determinant coefficients of ψ in lexicographical order. Let's assign more meaningful names to the orbitals of ψ:

```
>> psi = set(psi,'orbnames',{'1s' '1s~' '2s' '2s~' '2p' '2p~'})

psi =
    Fermi State (orbs == 6, N == 4)
    (0.70711)|1s 1s~ 2s 2s~> + (0+0.70711i)|1s 1s~ 2s 2p>
```

From a physics viewpoint, these orbitals could form electronic subshells in atoms. The rank-one projector $|\psi\rangle\langle\psi|$ or "density matrix" of ψ can be calculated intuitively by

```
>> psi*psi'

ans =
    Fermi Operator wedge^4 H -> wedge^4 H (orbs == 6)

Matrix representation w.r.t. ordered Slater basis
(|1s 1s~ 2s 2s~>, |1s 1s~ 2s 2p>, ... |2s 2s~ 2p 2p~>) ->
    (|1s 1s~ 2s 2s~>, |1s 1s~ 2s 2p>, ... |2s 2s~ 2p 2p~>):

  Columns 1 through 4

   0.5000                 0 - 0.5000i          0                    0
        0 + 0.5000i   0.5000                    0                    0

   ...
```

Note that the result is now a `fermiop` operator acting on $\bigwedge^4 \mathcal{H}$.

1.3. Reduced density matrices

The core feature of the toolbox is the efficient calculation of RDMs. For example, the 2-body RDM

$$\langle ij|\gamma_{|\psi\rangle\langle\psi|}kl\rangle := \langle\psi|a_k^\dagger a_l^\dagger a_j a_i|\psi\rangle$$

can be obtained by

```
>> rdm(psi,2)

ans =
    Fermi Operator wedge^2 H -> wedge^2 H (orbs == 6)

Matrix representation w.r.t. ordered Slater basis
(|1s 1s~>, |1s 2s>, ... |2p 2p~>) -> (|1s 1s~>, |1s 2s>, ... |2p 2p~>):

  Columns 1 through 4

   1.0000                 0                 0                 0
        0            1.0000                 0                 0
        0                 0            0.5000            0 - 0.5000i
        0                 0       0 + 0.5000i            0.5000
   ...
```

RDMs are reviewed in more detail in Section 2.3.

1.4. Tensor products of operators

Given a linear operator $A : \mathcal{H} \to \mathcal{H}$, a straightforward derivation based on the antisymmetrized structure of $\bigwedge^N \mathcal{H}$ shows that

$$\langle j_1, \ldots, j_N|(A \otimes \cdots \otimes A)|i_1, \ldots, i_N\rangle = \det\langle j_k|A|i_\ell\rangle_{k,\ell}$$

for all $1 \leqslant i_1 < \cdots < i_N \leqslant \dim\mathcal{H}$ and $1 \leqslant j_1 < \cdots < j_N \leqslant \dim\mathcal{H}$. That is, we obtain a matrix representation of $A \otimes \cdots \otimes A$ acting on $\bigwedge^N \mathcal{H}$. The tensor_op command implements precisely this operation. The following code lines are taken from the "natural orbitals" example in test/norbs.m:

```
>> orbs = 6; N = 4;
>> psi = fermistate(orbs,N,crand(nchoosek(orbs,N),1));
>> [U,D] = eig(rdm(psi,1));
```

crand generates pseudorandom complex numbers (similar to rand), and eig computes eigenvalues and -vectors. Thus, the eigenvectors of the 1-body RDM of ψ are stored in U. Performing a corresponding base change on $\bigwedge^N \mathcal{H}$ using these eigenvectors should result in a diagonal 1-body RDM [5]:

```
>> psi = tensor_op(U,N)'*psi;
>> G = get(rdm(psi,1),'data');
>> err = norm(G-diag(diag(G)))

err =

  1.6512e-015
```

In many physical applications, one can take advantage of unitary base changes on $\mathcal{H}$ such that subsequent computations are simplified, e.g., by choosing single-particle eigenstates of the L_z angular momentum operator. The above code shows how to implement the according base change on $\bigwedge^N \mathcal{H}$.

1.5. State configurations

For performance and memory efficiency reasons, FermiFab has built-in "configurations", i.e., we can subdivide the available orbitals into several groups, each of which contains a fixed number of particles. (Physically speaking, the groups could be interpreted as atomic subshells 1s, 2s, 2p, 3s, for example.) Let's say our system involves a total of 3 particles in 9 orbitals, with exactly 2 particles in the first 5 orbitals and 1 particle in the remaining 4 orbitals. Then a fermistate reflecting this configuration is specified by

```
>> orbs = [5,4]; N = [2,1];
>> psi = fermistate(orbs,N)
```

```
psi =
    Fermi State (orbs == 9, N == 3)
    |126>
```

Note that $|126\rangle$ is the lexicographically first base vector respecting the configuration constraints, and that ψ requires only $\binom{5}{2} \cdot \binom{4}{1} = 40$ rather than $\binom{9}{3} = 84$ complex entries:

```
>> length(psi)

ans =

    40
```

The `rdm` command works transparently for any configuration, so ψ behaves like a standard 9-orbital, 3-particle state.

What happens if we add two `fermistates` with different but compatible configurations (i.e., the total number of orbitals and particles is the same)?

```
>> orbs = [2,7]; N = [1,2];
>> phi = fermistate(orbs,N)

phi =
    Fermi State (orbs == 9, N == 3)
    |134>

>> length(phi)

ans =

    42

>> chi = psi+phi

chi =
    Fermi State (orbs == 9, N == 3)
    |126> + |134>
```

as expected – so how is this accomplished? `FermiFab` has detected that it needs to combine the two configurations, resulting in the full-fledged 9-orbital, 3-particle state. This fact can be checked by

```
>> length(chi)

ans =

    84
```

1.6. Symbolic computations

The Mathematica version of `FermiFab` is – quite naturally – inherently based on symbolic language. Considering Matlab, the (optionally available) Symbolic Math Toolbox integrates seamlessly into `FermiFab`, too. Taking advantage of symbolic computations is thus easily accomplished. That is, in the above examples, we may as well insert symbolic variables:

```
>> syms a b c
>> y = sym(zeros(1,nchoosek(orbs,N)));
>> y(1) = a; y(3) = 1i*b^2; y(4) = 1/c;
>> psi = set(psi,'data',y)

psi =
    Fermi State (orbs == 6, N == 4)
    (a)|1s 1s~ 2s 2s~> + (b^2*i)|1s 1s~ 2s 2p~> + (1/c)|1s 1s~ 2s~ 2p>

>> rdm(psi,2)

ans =
    Fermi Operator wedge^2 H -> wedge^2 H (orbs == 6)
```

Algorithm 1 NextFermi

Input: s: **bitfield**
1: $t \Leftarrow (s \vee (s-1)) + 1$
2: **return** $t \vee (((\text{LastBit}(t) - 1)/\text{LastBit}(s)) \gg 1)$

```
Matrix representation w.r.t. ordered Slater basis
(|12>, |13>, ... |56>) -> (|12>, |13>, ... |56>):

[ (c*b^2*conj(b)^2 + a*c*conj(a))/c + 1/(c*conj(c)),
...
```

2. Implementation details

The algorithmic implementation is based on the canonical mapping from Slater determinants to *bitfields*. That is, each Slater determinant corresponds to an unsigned integer s, where the ith bit is set to 1 precisely when the ith orbital is occupied. To remain unambiguous in terms of bitlength, the first orbital is stored in the LSB (least significant bit). Now, our task consists of re-expressing the creation/annihilation and RDM formalism in terms of bit operations. Note that, for example, testing whether all occupied orbitals in s_1 are also occupied in s_2 amounts to the pretty simple line of code $s_1 \wedge s_2 = s_1$, where we have used the bitwise AND operator $\wedge$. The following table summarizes all required bit operations:

bitwise AND:	$x \wedge y$
bitwise OR:	$x \vee y$
bitwise XOR:	$x \oplus y$
bit shift left:	$x \ll n$
bit shift right:	$x \gg n$
bit count:	$\#(x)$

For example, $1001101_2 \gg 3 = 1001_2$ and $\#(18) = \#(10010_2) = 2$. Note that bit operations are typically very "cheap" on CPUs. (In particular, refer to the SSE4 [10] POPCNT "population count" instruction for bit counting.) Diving a little bit further down into CPU intrinsics, we will make use of two's-complement arithmetic for negating numbers [11], e.g.,

$$x = \cdots 001011\mathbf{1}00_2 \quad \rightsquigarrow$$
$$-x = \cdots 110100\mathbf{1}00_2. \tag{1}$$

Interestingly, precisely all bits flip which are more significant than the least significant 1-bit (marked in bold). Thus, we can use this property to extract the last 1-bit from a bitfield $x \neq 0$ simply by

$$\text{LastBit}(x) := x \wedge (-x).$$

(An less universal alternative is the BSF "bit scan forward" instruction [12], which returns the index of the least significant 1-bit.)

2.1. Enumerating Slater determinants

The basic task we set out to accomplish in this subsection is lexicographically enumerating all Slater determinants of a fixed particle number N and number of orbitals 'orbs'. This amounts to computing the lexicographically next bit permutation (denoted by 'NextFermi'). For example,

$$s = \mathbf{0}1111000_2 \quad \rightsquigarrow$$
$$\text{NextFermi}(s) = \mathbf{1}0000111_2.$$

Closer inspection reveals the general rule that the leading 1-bit (marked in bold) in the least significant block of 1s gets shifted to the left by one position, and the remaining 1-bits are shifted to the end. Algorithm 1 is adopted from [13] and performs exactly this computation. In line 1, $s \vee (s-1)$ sets the trailing zeros in s to 1, so for example, $s = \mathbf{0}1111000_2 \rightsquigarrow s \vee (s-1) = \mathbf{0}1111111_2$ and $t = \mathbf{1}0000000_2$. The second term in line 2 performs the shifting of the remaining 1-bits to the end.

As an extension of Algorithm 1, we want to take into account "configurations", i.e., a subdivision of the available orbitals into several groups, each of which contains a fixed number of particles. For example, we compartmentalize a total of 11 orbitals such that exactly 4 particles are in the first 6 orbitals and 2 in the remaining 5 orbitals, written as $(\text{orbs}_1, \text{orbs}_2) = (6, 5)$ and $(N_1, N_2) = (4, 2)$. Then a sequence of patterns – respecting the configuration restrictions – would be

$$0|01010|110110_2,$$
$$0|01010|11\mathbf{1}00\mathbf{1}_2,$$
$$0|01010|1110\mathbf{1}0_2,$$
$$0|01010|111\mathbf{1}00_2,$$
$$0|01\mathbf{1}00|00\mathbf{1111}_2, \tag{2}$$

where we have highlighted the currently changing 1-bits in bold.

Algorithm 2 NextFermiConfig

Input: s: **bitfield**, orbs: **int array**
1: mask $\Leftarrow (1 \ll \text{orbs}[0]) - 1$
2: **if** $(((s \vee (s - 1)) \wedge \text{mask}) \neq \text{mask})$ **then**
3:     **return** NextFermi(s)
4: **else**
5:     **if** orbs.length $= 1$ **then**
6:         **return** -1
7:     **end if**
8:     $t \Leftarrow$ NextFermiConfig($s \gg \text{orbs}[0]$, orbs[1, . . . , end])
9:     **if** $t = -1$ **then**
10:         **return** -1
11:     **end if**
12:     **return** (mask/LastBit(s)) $\vee (t \ll \text{orbs}[0])$
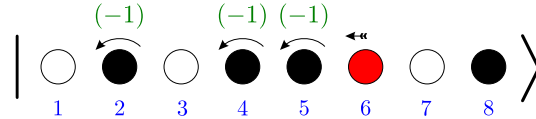13: **end if**

Fig. 2. Annihilation of a single orbital (red). Figuratively, the red orbital moves to the front before being removed, such that each flip (curved arrows) with an occupied preceding orbital contributes a sign factor of (-1). In terms of quantum mechanics, $a_{|6\rangle}|24568\rangle = -|2458\rangle$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

More formally, given $(\text{orbs}_1, \ldots, \text{orbs}_k)$, the compartmentalization may be written as $V_j := \text{span}\{|i\rangle: b_{j-1} < i \leqslant b_j\} \subset \mathcal{H}$ with $b_j := \sum_{\ell=1}^{j} \text{orbs}_\ell$. In other words, $\mathcal{H} = \bigoplus_j V_j$. In the example above, $V_1 = \text{span}\{|1\rangle, \ldots, |6\rangle\}$ and $V_2 = \text{span}\{|7\rangle, \ldots, |11\rangle\}$. Now, mathematically speaking, a *configuration* of an N-particle state is a subspace of $\bigwedge^N \mathcal{H}$ of the following form:

$$\mathcal{C}^{N_1,\ldots,N_k} := \text{span}\{|i_1, \ldots, i_N\rangle: \sharp\{\ell: |i_\ell\rangle \in V_j\} = N_j\} \tag{3}$$

where $(N_1, \ldots, N_k)$ is a partition of N (i.e. $0 \leqslant N_j \leqslant \text{orbs}_j$, $\sum_j N_j = N$). A quantum chemist could interpret the V_j as atomic subshells $1s, 2s, 2p, 3s, \ldots$ and the N_j as occupation numbers. An interesting consequence of definition (3) is the recovery of a tensor product structure, namely

$$\mathcal{C}^{N_1,\ldots,N_k} \cong \bigotimes_{j=1}^{k} \bigwedge^{N_j} V_j. \tag{4}$$

This follows from the observation that a configuration is constructed by the lexicographical enumeration of Slater determinants within orbital groups, as illustrated in (2).

Algorithm 2 implements precisely this enumeration. In accordance with the lexicographical scheme, it first iterates through all Slater determinants within the least significant orbital group (line 3), then resets this group (first term in line 12) and recursively computes the next bit pattern for the remaining groups (line 8). The mask in line 1 is required for testing whether the last bit permutation within the least significant group has been reached (line 2). In the example above, we would have mask $= 0|111111_2$.

2.2. Creation/annihilation operators

The creation/annihilation operator formalism is an essential ingredient of many-particle quantum mechanics and quantum field theory [14]. For a very brief sketch, let $\varphi \in \bigwedge^p \mathcal{H}$ be a p-particle wavefunction with $1 \leqslant p \leqslant N$. Then, the linear *annihilation operator* a_φ acting on $\bigwedge^N \mathcal{H}$ removes or "annihilates" the state φ from $\bigwedge^N \mathcal{H}$. More precisely, a_φ is uniquely determined by its antilinearity in φ,

$$a_{c\varphi_1 + \varphi_2} = \bar{c}a_{\varphi_1} + a_{\varphi_2} \quad \forall c \in \mathbb{C}, \ \varphi_1, \varphi_2 \in \bigwedge^p \mathcal{H}$$

together with the decomposition for Slater determinants,

$$a_{|i_1, i_2, \ldots, i_p\rangle} := a_{|i_p\rangle} \cdots a_{|i_2\rangle} a_{|i_1\rangle} \quad \forall 1 \leqslant i_1 < \cdots < i_p \leqslant \dim \mathcal{H},$$

as well as the definition

$$a_{|i\rangle}|j_1, \ldots, j_N\rangle := \begin{cases} (-1)^{k-1}|j_1, \ldots, j_{k-1}, j_{k+1}, \ldots, j_N\rangle, & i = j_k, \\ 0, & i \notin \{j_1, \ldots, j_N\}, \end{cases}$$

for all $1 \leqslant j_1 < \cdots < j_N \leqslant \dim \mathcal{H}$. The sign factor can be interpreted as number of orbital "flips" illustrated in Fig. 2.

So far we have considered annihilation operators only. The *creation operator* $a_\varphi^\dagger$ is by definition the adjoint (conjugate transpose) of a_φ, as the notation already suggests. It can be shown that the following relations hold, where the anticommutator bracket is defined by $\{A, B\} := AB - BA$ and $\varphi, \chi \in \bigwedge^p \mathcal{H}$ are arbitrary wavefunctions:

$$\{a_\varphi, a_\chi\} = 0, \qquad \{a_\varphi^\dagger, a_\chi^\dagger\} = 0, \qquad \{a_\varphi, a_\chi^\dagger\} = \langle \varphi | \chi \rangle.$$

Algorithm 3 AnnihilSignMask

Input: s: **bitfield**
1: **bitfield** $m \Leftarrow 0$
2: **while** $s \neq 0$ **do**
3:     $t \Leftarrow \text{LastBit}(s)$
4:     $m \Leftarrow m \oplus (-t)$
5:     $s \Leftarrow s - t$
6: **end while**
7: **return** m

In the remainder of this subsection, we want to detail an efficient algorithmic implementation of the annihilation operation, w.l.o.g. for Slater determinants only. More precisely, let $|s\rangle \in \bigwedge^N \mathcal{H}$ be a fixed Slater determinant, then our task is the calculation of $a_{|t\rangle}|s\rangle$ for arbitrary Slater determinants $|t\rangle \in \bigwedge^p \mathcal{H}$ and $1 \leqslant p \leqslant N$. The result will be nonzero only if all occupied orbitals in t are also occupied in s, which can be tested by $t \wedge s = t$ as already mentioned in the beginning. Given this holds true, the bit pattern describing the Slater determinant $a_{|t\rangle}|s\rangle$ is simply $s - t$, so what essentially remains is the calculation of the sign factor.

For that purpose, we define the *annihilation sign mask* of s such that each bit stores the integer parity of the number of less or equal significant 1-bits in s. That is, if s has binary representation

$$s = \ldots a_2 a_1 a_0, \quad a_i \in \{0, 1\}, \quad \text{then}$$

$$\text{AnnihilSignMask}(s) := \ldots b_2 b_1 b_0 \quad \text{with } b_i \equiv \sum_{j=0}^{i} a_j \mod 2.$$

For example, $s = \overset{7\,6\,5\,4\,3\,2\,1\,0}{00101100}_2$ results in $\text{AnnihilSignMask}(s) = \ldots \overset{7\,6\,5\,4\,3\,2\,1\,0}{11100100}_2$ where overhead numbers label bit positions. Algorithm 3 implements this calculation. It has a running time of $\mathcal{O}(\#(s))$ since the last statement (line 5) in the while loop removes the least significant 1-bit from s. In line 4, the $\oplus(-t)$ operation flips all bits which are less or equal significant than the current least significant 1-bit.

Finally, we define the *reverse permutation sign* $\sigma_{\text{revperm}}(n)$ for all $n \in \mathbb{N}_{\geqslant 1}$ by the sign of the permutation $i \mapsto n - i + 1$ ($i = 1, \ldots, n$). A moment's thought reveals that

$$\sigma_{\text{revperm}}(n) = (-1)^{\frac{1}{2}(n-1)n}.$$

Altogether, our devised algorithm is illustrated in Fig. 3. More formally, we obtain $a_{|t\rangle}|s\rangle = \zeta \cdot |s - t\rangle$ with the sign factor ζ equal to

$$\zeta = \sigma_{\text{revperm}}\big(\#(t)\big) \cdot (-1)^{\#(a_{\text{mask}} \wedge t)}, \tag{5}$$

where we have set $a_{\text{mask}} := \text{AnnihilSignMask}(s) \ll 1$. Eq. (5) will be the basic building block for calculating reduced density matrices in Algorithm 4 below, as described in the next subsection.

2.3. Reduced density matrices

In this subsection we briefly recall the relevant abstract formalism, and then describe the algorithmic implementation in the `FermiFab` toolbox. Let $1 \leqslant p_k \leqslant N_k$ ($k = 1, 2$) and denote orthonormal basis sets of $\bigwedge^{p_k} \mathcal{H}$ by $(\varphi_{ki})_i$. For wavefunctions $\psi_k \in \bigwedge^{N_k} \mathcal{H}$ ($k = 1, 2$), define the *reduced density matrix* $\gamma_{|\psi_1\rangle\langle\psi_2|} : \bigwedge^{p_2} \mathcal{H} \to \bigwedge^{p_1} \mathcal{H}$ by

$$\langle \varphi_{1j} | \gamma_{|\psi_1\rangle\langle\psi_2|} | \varphi_{2i} \rangle := \big\langle \psi_2 \big| a_{\varphi_{2i}}^\dagger a_{\varphi_{1j}} \psi_1 \big\rangle = \langle a_{\varphi_{2i}} \psi_2 | a_{\varphi_{1j}} \psi_1 \rangle \quad \forall i, j, \tag{6}$$

where we have employed the creation/annihilation operators defined in the last subsection. The significance of this definition can be seen as follows. Any linear map $b : \bigwedge^{p_1} \mathcal{H} \to \bigwedge^{p_2} \mathcal{H}$ with matrix representation (b_{ij}) may be "lifted" to an operator $B : \bigwedge^{N_1} \mathcal{H} \to \bigwedge^{N_2} \mathcal{H}$ by

$$B := \sum_{i,j} b_{ij} a_{\varphi_{2i}}^\dagger a_{\varphi_{1j}}. \tag{7}$$

(A prominent example is the Coulomb operator ($p_1 = p_2 = 2$), which describes the *pairwise* interaction between charged particles.) Now, the B expectation value with respect to $|\psi_1\rangle\langle\psi_2|$ equals

$$\langle \psi_2 | B \psi_1 \rangle \overset{\text{def}}{=} \sum_{i,j} b_{ij} \big\langle \psi_2 \big| a_{\varphi_{2i}}^\dagger a_{\varphi_{1j}} \psi_1 \big\rangle = \text{tr}_{\bigwedge^{p_2} \mathcal{H}} (b \gamma_{|\psi_1\rangle\langle\psi_2|}). \tag{8}$$

In other words, this equation switches from $\bigwedge^{N_k} \mathcal{H}$ to $\bigwedge^{p_k} \mathcal{H}$ ($k = 1, 2$). For many applications, this is the only possibility to avoid the "curse of dimensionality" induced by the N_1, N_2-particle systems. In terms of `FermiFab`, (7) is implemented by the `p2N` command.

In the rest of this subsection, we focus on the calculation of $\gamma_{|\psi_1\rangle\langle\psi_2|}$ in Algorithm 4. Due to linearity, it suffices to restrict ourselves to Slater determinants. That is, ψ_1 and ψ_2 are (w.l.o.g.) replaced by Slater determinants s_1 and s_2, respectively, and it is assumed that the (φ_{ki}) are Slater determinants, too. So the last term in (6) can be concisely written as $\langle a_{t_2} s_2 | a_{t_1} s_1 \rangle$ with Slater determinants $t_k \in \bigwedge^{p_k} \mathcal{H}$ ($k = 1, 2$). Note that the particle number conservation law imposes $N_1 - p_1 = N_2 - p_2$, otherwise all terms will be zero; so we calculate p_2 from given N_1, N_2 and p_1.

The basic algorithmic idea is exemplified in Fig. 4. Namely, we subsume all orbitals occupied either in s_1 or s_2, but not in both, as "force" group, and all orbitals occupied in both s_1 and s_2 as "choice" group. The corresponding bit patterns f_{mask} and c_{mask} are computed in lines 1 and 13 of Algorithm 4 by a single bit operation. Since $\langle a_{t_2} s_2 | a_{t_1} s_1 \rangle$ is nonzero only if $a_{t_1} s_1 = \pm a_{t_2} s_2$, all occupied "force" orbitals
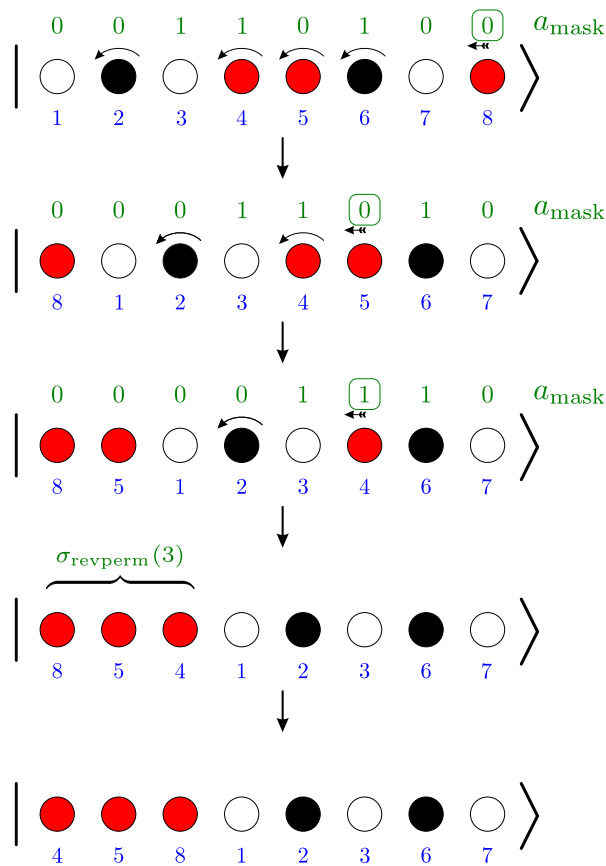
Fig. 3. The cumulative sign factors incurred during rearrangement of the to-be annihilated (red) orbitals to the front. In terms of quantum mechanics, the corresponding operation reads $a_{|458\rangle}|24568\rangle = |26\rangle$. The contribution from all flips (curved arrows) during each step can be obtained from the marked bit in a_{mask}. Note that this mask needs to be calculated only once. The permutation sign for sorting the three red orbitals in the last step equals $\sigma_{revperm}(3) = -1$, so the overall sign factor is 1. Algorithmically, the whole schematic is implemented by Eq. (5). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 4 SlaterRDM

Input: s_1, s_2: **bitfield**, p_1: **int**
1: $f_{mask} \Leftarrow s_1 \oplus s_2$ // "force" mask
2: $s_{force,k} \Leftarrow (f_{mask} \wedge s_k)$ $(k = 1, 2)$
3: $n_{choice,1} \Leftarrow p_1 - \#(s_{force,1})$
4: **if** $n_{choice,1} < 0$ **then**
5:     **return** 0
6: **end if**
7: $p_2 \Leftarrow \#(s_2) - \#(s_1) + p_1$
8: $a_{mask,k} \Leftarrow \text{AnnihilSignMask}(s_k) \ll 1, k = 1, 2$
9: $\zeta \Leftarrow \prod_{k=1}^{2} \sigma_{revperm}(p_k) \cdot (-1)^{\#(a_{mask,k} \wedge s_{force,k})}$ // sign factor
10: **if** $n_{choice,1} = 0$ **then**
11:     **return** $\zeta \cdot |s_{force,1}\rangle\langle s_{force,2}|$
12: **end if**
13: $c_{mask} \Leftarrow s_1 \wedge s_2$ // "choice" mask
14: $k_{choice} \Leftarrow \#(c_{mask})$
15: $r \Leftarrow \{\}$
16: $t \Leftarrow (1 \ll n_{choice,1}) - 1$
17: **while** $(t \gg k_{choice}) = 0$ **do** // iterate Fermi map of 'choice' orbitals
18:     $s_{choice} \Leftarrow \text{BitDistribute}(t, c_{mask})$
19:     append $r \Leftarrow \zeta \cdot \prod_{k=1}^{2} (-1)^{\#(a_{mask,k} \wedge s_{choice})} |s_{force,1} + s_{choice}\rangle\langle s_{force,2} + s_{choice}|$
20:     $t \Leftarrow \text{NextFermi}(t)$
21: **end while**
22: **return** r

---

have to be annihilated by a_{t_1} and a_{t_2}, respectively. On the other hand, each "choice" orbital annihilated by a_{t_1} must also be annihilated by a_{t_2} and vice versa, but there's a freedom in exactly which of these orbitals to select, hence the "choice" designator. In our example, the only force orbital occupied in s_1 is 7, so t_1 must contain 7 but may "choose" between 5, 6 and 9. If $p_1 = 3$, we obtain t_1 equal to one of $|567\rangle$, $|579\rangle$ or $|679\rangle$. The respective t_2 states are then $|12456\rangle$, $|12459\rangle$ and $|12469\rangle$. After the obligatory annihilation sign factor calculations (5), the final result (for $p_1 = 3$) is

$$\gamma_{|5679\rangle\langle124569|} = |567\rangle\langle12456| - |579\rangle\langle12459| - |679\rangle\langle12469|.$$
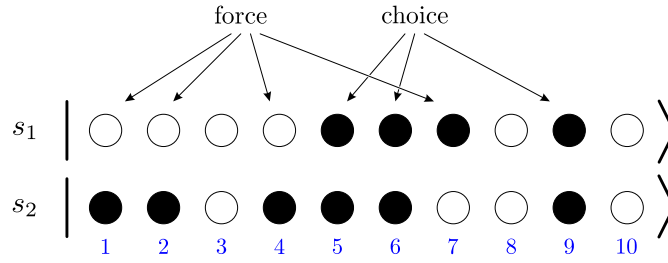
Fig. 4. Alignment of two Slater determinants for the annihilation operation. "Force" labels the orbitals which are either occupied in s_1 or s_2, but not in both, whereas "choice" labels all orbitals occupied in s_1 as well as s_2.
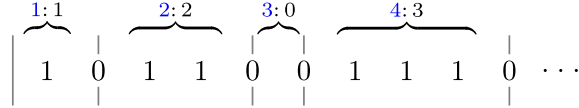

Fig. 5. Bit-encoding of a bosonic state. Blue numbers label modes, and 0-bits serve as delimiters between modes. The shown state consists of one boson in the 1st mode, two in the 2nd mode, zero in the 3rd and three in the 4th. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 4 implements Eq. (5) in line 9 and the first term of line 19. $s_{\text{force},k}$ stores the orbitals which must be annihilated in s_k ($k = 1, 2$), and the number of to-be annihilated "choice" orbitals in s_1 is computed in line 3. The while loop accumulates the return value list r containing the ket-bra's as in the above example. In line 18, the algorithm uses the 'BitDistribute' command, which basically just shifts bits to the positions designated by the 1-bits in c_{mask}.

2.4. Bosons

As a short outlook, we want to illustrate how the developed methods can easily be adapted to bosonic systems as well. In quantum mechanics, bosons are subatomic particles which obey Bose–Einstein statistics, like, for example, photons. For our purposes, we replace fermionic "orbitals" by bosonic "modes", which can be multiply occupied (i.e., the Pauli exclusion principle no longer holds for bosons). That is, the bosonic analogue of a fermionic Slater determinant differs only by the unrestricted number of particles in each mode. The central observation of this subsection states that a bit-encoding (equivalent to Slater determinants) works for bosons as well. The idea is detailed in Fig. 5, where 0-bits serve as delimiters between modes. Lexicographical enumeration of bosonic states with a fixed total particle number N and number of modes m is accomplished via enumeration of the bit-encoded Slater determinants with $(m + N - 1)$ orbitals and N particles! That is, Algorithm 1 may be employed without modifications.

3. Application to transition metal atoms

The application example is based on the series [15–17], in which [17] makes use of the FermiFab toolbox to calculate ground state approximations for transition metal atoms (employing the so-called configuration-interaction (CI) methods). The underlying quantum mechanical (non-relativistic, Born–Oppenheimer) Hamiltonian $H = H_0 + V_{ee}$ with

$$H_0 = \sum_{i=1}^{N} \left(-\frac{1}{2} \Delta_{\boldsymbol{x}_i} - \frac{Z}{|\boldsymbol{x}_i|} \right), \qquad V_{ee} = \sum_{1 \leqslant i < j \leqslant N} \frac{1}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}$$

governs atoms/ions with N electrons and nuclear charge Z. The two terms in H_0 are the single-particle kinetic energy and nuclear potential, respectively, whereas the Coulomb operator V_{ee} describes the pairwise inter-electron Coulomb repulsion. The Hamiltonian leaves the simultaneous eigenspaces of the well-known angular momentum, spin and parity ('LS') operators invariant, so calculating these eigenspaces *first* leads to a huge dimension reduction. Specifically, the FermiFab toolbox automates the LS-eigenspace computation by combining configurations (4) with Clebsch–Gordan coefficients. We skip further details here; instead, for the purpose of this section, we provide two orthonormal LS-eigenstates of neutral Chromium ($N = Z = 24$) with symmetry level $_5D$:

$$\psi_1 := \frac{1}{\sqrt{10}} \big(|3d_0\, 3d_m\, 3d_x\, 4s\, \overline{4s}\, 4d_x\rangle - |3d_0\, 3d_m\, 3d_y\, 4s\, \overline{4s}\, 4d_y\rangle - |3d_0\, 3d_z\, 3d_x\, 4s\, \overline{4s}\, 4d_y\rangle$$

$$- |3d_0\, 3d_z\, 3d_y\, 4s\, \overline{4s}\, 4d_x\rangle + \sqrt{3}|3d_z\, 3d_m\, 3d_x\, 4s\, \overline{4s}\, 4d_y\rangle - \sqrt{3}|3d_z\, 3d_m\, 3d_y\, 4s\, \overline{4s}\, 4d_x\rangle \big)$$

and

$$\psi_2 := \frac{1}{\sqrt{21}} \Big(-\sqrt{3}/2\,|3d_0\, 4s\, \overline{4s}\, 4p_z\, 4p_x\, 4d_y\rangle - \sqrt{3}/2\,|3d_0\, 4s\, \overline{4s}\, 4p_z\, 4p_y\, 4d_x\rangle$$

$$+ 2|3d_m\, 4s\, \overline{4s}\, 4p_x\, 4p_y\, 4d_z\rangle + \frac{1}{2}|3d_m\, 4s\, \overline{4s}\, 4p_z\, 4p_x\, 4d_y\rangle - \frac{1}{2}|3d_m\, 4s\, \overline{4s}\, 4p_z\, 4p_y\, 4d_x\rangle$$

$$+ \frac{1}{2}|3d_x\, 4s\, \overline{4s}\, 4p_x\, 4p_y\, 4d_y\rangle + |3d_x\, 4s\, \overline{4s}\, 4p_z\, 4p_x\, 4d_z\rangle + \sqrt{3}|3d_x\, 4s\, \overline{4s}\, 4p_z\, 4p_y\, 4d_0\rangle$$

$$-|3d_x\,4s\,\overline{4s}\,4p_z\,4p_y\,4d_m\rangle - \frac{1}{2}|3d_y\,4s\,\overline{4s}\,4p_x\,4p_y\,4d_x\rangle + \sqrt{3}|3d_y\,4s\,\overline{4s}\,4p_z\,4p_x\,4d_0\rangle$$

$$+\,|3d_y\,4s\,\overline{4s}\,4p_z\,4p_x\,4d_m\rangle + |3d_y\,4s\,\overline{4s}\,4p_z\,4p_y\,4d_z\rangle - 2|3d_z\,4s\,\overline{4s}\,4p_x\,4p_y\,4d_m\rangle$$

$$+\,\frac{1}{2}|3d_z\,4s\,\overline{4s}\,4p_z\,4p_x\,4d_x\rangle + \frac{1}{2}|3d_z\,4s\,\overline{4s}\,4p_z\,4p_y\,4d_y\rangle\Big).$$

In this notation, $\overline{}$ means spin down $\downarrow$, otherwise up $\uparrow$, and the s, p, d subshell orbitals are labeled s, $p_z\,p_x\,p_y$ and $d_0\,d_z\,d_m\,d_x\,d_y$, respectively. The numbers 3 and 4 denote the third and fourth shell. Since all spin-orbitals up to $3p$ are fully occupied, they are not shown here for conciseness of notation.

The following paragraph demonstrates how to translate the expectation value $\langle\psi_2|V_{ee}\psi_1\rangle$ into a list of Coulomb integral symbols

$$(ab|cd) := \int_{\mathbb{R}^6} a^*(\boldsymbol{x}_1)b(\boldsymbol{x}_1)\frac{1}{|\boldsymbol{x}_1-\boldsymbol{x}_2|}c^*(\boldsymbol{x}_2)d(\boldsymbol{x}_2)\,\mathrm{d}\boldsymbol{x}_1\,\mathrm{d}\boldsymbol{x}_2, \tag{9}$$

where $a, b, c, d \in L^2(\mathbb{R}^3)$ are *spatial* orbitals and * denotes complex conjugation. As shown in (8), the essential step is the calculation of the 2-body reduced density matrix $\gamma_{|\psi_1\rangle\langle\psi_2|}$. Using the Mathematica version of `FermiFab`, this is accomplished by the first line of the following code sample (see `mathematica/RDMdemo.nb`); the subsequent code just displays the result:

```
ρ₂ = RDM[ψ₁, ψ₂, 2];

SlaterToPureString[orbnames_, s_] :=
 StringJoin @@ Flatten[{#, " "} & /@ orbnames⟦FermiToCoords[s]⟧]⟦1 ;; -2⟧

(* display RDM *)
ρ₂ /. IXI[ket_, bra_] :> "|" <> SlaterToPureString[allShells, ket] <>
   ")⟨" <> SlaterToPureString[allShells, bra] <> "|"
```

$$\frac{|3d0\ 3dx\rangle\langle 4pz\ 4py|}{\sqrt{210}} + \frac{|3d0\ 3dy\rangle\langle 4pz\ 4px|}{\sqrt{210}} + \frac{|3dz\ 3dm\rangle\langle 4px\ 4py|}{\sqrt{70}}$$

The `FermiToCoords` command converts any bit-encoded Slater determinant to a vector of integers enumerating the occupied orbitals.

Since the Coulomb operator is independent of spin, we may effectively "trace out" the spin coordinate from the employed spin-orbitals. Specifically, consider single-particle wavefunctions

$$\chi_i(\boldsymbol{x}, s) = \varphi_i(\boldsymbol{x})\alpha_i(s), \quad \boldsymbol{x} \in \mathbb{R}^3,\ s \in \{\uparrow, \downarrow\},\ i = 1, \dots, 4$$

which factor into the spatial part φ_i and spin part α_i. Endowing particle i with coordinates $(\boldsymbol{x}_i, s_i)$, the antisymmetrized 2-body Slater determinants read

$$|\chi_i\chi_j\rangle \equiv \frac{1}{\sqrt{2}}\big(\varphi_i(\boldsymbol{x}_1)\alpha_i(s_1)\varphi_j(\boldsymbol{x}_2)\alpha_j(s_2) - \varphi_j(\boldsymbol{x}_1)\alpha_j(s_1)\varphi_i(\boldsymbol{x}_2)\alpha_i(s_2)\big).$$

Plugged into the following equation for the Coulomb expectation value yields

$$\left\langle\chi_1\chi_2\left|\frac{1}{|\boldsymbol{x}_1-\boldsymbol{x}_2|}\chi_3\chi_4\right.\right\rangle = (\varphi_1\varphi_3|\varphi_2\varphi_4)\langle\alpha_1|\alpha_3\rangle\langle\alpha_2|\alpha_4\rangle - (\varphi_1\varphi_4|\varphi_2\varphi_3)\langle\alpha_1|\alpha_4\rangle\langle\alpha_2|\alpha_3\rangle.$$

Translating this equation to alternating spin up $\uparrow$ and down $\downarrow$ orbitals (and taking symmetries of $(ab|cd)$ into account) is accomplished by the `SpinTraceCoulomb` command in the first line of the following code sample:

```
ρs₂ = SpinTraceCoulomb[(List @@ ρ₂) /. c_ IXI[ket_, bra_] :> {c, {ket, bra}}];

BosonToString[orbnames_, s_] :=
 StringJoin @@ Flatten[{#, " "} & /@ orbnames⟦BosonToCoords[s]⟧]⟦1 ;; -2⟧

(* display Coulomb integrals *)
ρs₂ /. CoulInt[a_, b_] :> "(" <> BosonToString[allShellsSpatial, a] <>
   "|" <> BosonToString[allShellsSpatial, b] <> ")"
```

$$-\frac{(3d0\ 4px|3dy\ 4pz)}{\sqrt{210}} - \frac{(3d0\ 4py|3dx\ 4pz)}{\sqrt{210}} + \frac{(3d0\ 4pz|3dx\ 4py)}{\sqrt{210}} +$$

$$\frac{(3d0\ 4pz|3dy\ 4px)}{\sqrt{210}} + \frac{(3dz\ 4px|3dm\ 4py)}{\sqrt{70}} - \frac{(3dz\ 4py|3dm\ 4px)}{\sqrt{70}}$$

Note that spatial orbitals can appear twice within a Coulomb integral symbol, e.g., $(aa|bc)$. Thus, a bosonic encoding of these spatial orbitals is used to accommodate multiple occurrences, and hence the `BosonToCoords` command.

Concluding, we have obtained the desired list of Coulomb integral symbols, which may then be evaluated by inserting concrete functions into (9).

Acknowledgements

References

[1] J. von Neumann, Mathematische Grundlagen der Quantenmechanik, Springer, 1932.

[2] P. Dirac, Note on exchange phenomena in the Thomas atom, Mathematical Proceedings of the Cambridge Philosophical Society 26 (1930) 376–385.

[3] P. Hohenberg, W. Kohn, Inhomogeneous electron gas, Physical Review 136 (1964) 864–871.

[4] W. Kohn, L.J. Sham, Self-consistent equations including exchange and correlation effects, Physical Review 140 (4A) (1965) 1133–1138.

[5] P.-O. Loewdin, Quantum theory of many-particle systems. I. Physical interpretations by means of density matrices, natural spin-orbitals, and convergence problems in the method of configurational interaction, Physical Review 97 (6) (1955) 1474–1489.

[6] A.J. Coleman, Structure of Fermion density matrices, Reviews of Modern Physics 35 (3) (1963) 668–686.

[7] T. Ando, Properties of Fermion density matrices, Reviews of Modern Physics 35 (3) (1963) 690–702.

[8] D.A. Mazziotti, Reduced-Density-Matrix Mechanics: With Application to Many-Electron Atoms and Molecules, Advances in Chemical Physics, vol. 134, Wiley, 2007.

[9] C.B. Mendl, FermiFab project at SourceForge, URL http://sourceforge.net/projects/fermifab, 2010.

[10] Intel, Intel SSE4 Programming Reference, URL software.intel.com, 2007.

[11] Intel, Intel 64 and IA-32 Architectures Software Developers Manuals: Basic Architecture, URL http://developer.intel.com/products/processor/manuals/index.htm, 2010.

[12] Intel, Intel 64 and IA-32 Architectures Software Developer's Manuals: Instruction Set Reference, URL http://developer.intel.com/products/processor/manuals/index.htm, 2010.

[13] S.E. Anderson, Bit Twiddling Hacks, URL http://graphics.stanford.edu/~seander/bithacks.html.

[14] M.E. Peskin, D.V. Schroeder, An Introduction To Quantum Field Theory, Frontiers in Physics, Westview Press, 1995.

[15] G. Friesecke, B.D. Goddard, Explicit large nuclear charge limit of electronic ground states for Li, Be, B, C, N, O, F, Ne and basic aspects of the periodic table, SIAM Journal on Mathematical Analysis 41 (2) (2009) 631–664, doi:10.1137/080729050.

[16] G. Friesecke, B.D. Goddard, Asymptotics-based CI models for atoms: properties, exact solution of a minimal model for Li to Ne, and application to atomic spectra, SIAM Multiscale Modeling and Simulation 7 (4) (2009) 1876–1897, doi:10.1137/080736648.

[17] C.B. Mendl, G. Friesecke, Efficient algorithm for asymptotics-based configuration-interaction methods and electronic structure of transition metal atoms, Journal of Chemical Physics 133 (2010) 184101.