An efficient implementation of Slater-Condon rules for determinant-driven calculations

Anthony SCEMAMA, Emmanuel GINER

Laboratoire de Chimie et Physique Quantiques, CNRS-IRSAMC, Université de Toulouse

Abstract Slater-Condon rules are at the heart of any quantum chemistry method as they allow to simplify $3N$-dimensional integrals as sums of 3- or 6-dimensional integrals. We propose an efficient implementation of those rules in order to identify very rapidly which integrals are involved in a matrix element expressed in the determinant basis set. This implementation takes advantage of the bit manipulation instructions on x86 architectures that were introduced in 2008 with the SSE4.2 instruction set. Finding which spin-orbitals are involved in the calculation of a matrix element doesn't depend on the number of electrons of the system.

Slater-Condon rules

We consider wave functions Ψ expressed as linear combinations of Slater determinants D of orthonormal spin-orbitals $\phi(\mathbf{r})$:

$$\Psi = \sum_i c_i D_i \tag{1}$$

Using the Slater-Condon rules, the matrix elements of any one-body ($\mathcal{O}_1$) or two-body ($\mathcal{O}_2$) operator expressed in the determinant space have simple expressions involving one- and two-electron integrals in the spin-orbital space. The diagonal elements are given by:

$$\begin{aligned}\langle D|\mathcal{O}_1|D\rangle &= \sum_{i\in D}\langle\phi_i|\mathcal{O}_1|\phi_i\rangle \\ \langle D|\mathcal{O}_2|D\rangle &= \frac{1}{2}\sum_{(i,j)\in D}\langle\phi_i\phi_j|\mathcal{O}_2|\phi_i\phi_j\rangle - \langle\phi_i\phi_j|\mathcal{O}_2|\phi_j\phi_i\rangle\end{aligned} \tag{2}$$

For two determinants which differ only by the substitution of spin-orbital i with spin-orbital j:

$$\begin{aligned}\langle D|\mathcal{O}_1|D_i^j\rangle &= \langle\phi_i|\mathcal{O}_1|\phi_j\rangle \\ \langle D|\mathcal{O}_2|D_i^j\rangle &= \sum_{k\in D}\langle\phi_i\phi_k|\mathcal{O}_2|\phi_j\phi_k\rangle - \langle\phi_i\phi_k|\mathcal{O}_2|\phi_k\phi_j\rangle\end{aligned} \tag{3}$$

For two determinants which differ by two spin-orbitals:

$$\begin{aligned}\langle D|\mathcal{O}_1|D_{ik}^{jl}\rangle &= 0 \\ \langle D|\mathcal{O}_2|D_{ik}^{jl}\rangle &= \langle\phi_i\phi_k|\mathcal{O}_2|\phi_j\phi_l\rangle - \langle\phi_i\phi_k|\mathcal{O}_2|\phi_l\phi_j\rangle\end{aligned} \tag{4}$$

All other matrix elements involving determinants with more than two substitutions are zero.

An efficient implementation of those rules requires:

1. to find the number of spin-orbital substitutions between two determinants
2. to find which spin-orbitals are involved in the substitution
3. to compute the phase factor if a reordering of the spin-orbitals has occured

Algorithm

We use the convention that the least significant bit of binary integers is the rightmost bit. As the position number of a bit in an integer is the exponent for the corresponding bit weight in base 2, the bit positions are numbered from the right to the left starting at position 0. To be consistent with this convention, we also represent the arrays of 64-bit integers from right to left, starting at position zero. Following with the usual notations, the spin-orbitals start with index one.

Binary representation of the determinants

The molecular spin-orbitals in the determinants are ordered by spin: the α spin-orbitals are placed before the β spin-orbitals. Each determinant is represented as a pair of bit-strings: one bit-string corresponding to the α spin-orbital occupations, and one bit-string for the β spin-orbital occupations. When the i-th orbital is occupied by an electron with spin σ in the determinant, the bit at position $(i-1)$ of the σ bit-string is set to one, otherwise it is set to zero.

The pair of bit-strings is encoded in a 2-dimensional array of 64-bit integers. The first dimension contains N_{int} elements and starts at position zero. N_{int} is the minimum number of 64-bit integers needed to encode the bit-strings:

$$N_{\text{int}} = \lfloor N_{\text{MOs}}/64 \rfloor + 1 \tag{5}$$

where N_{MOs} is the total number of molecular spin-orbitals with spin α or β (we assume this number to be the same for both spins). The second index of the array corresponds to the α or β spin. Hence, determinant D_k is represented by an array of N_{int} 64-bit integers $I^k_{i,\sigma}$, $i \in [0, N_{\text{int}}-1], \sigma \in \{1,2\}$.

Finding the number of substitutions

The number of substitutions between determinants D_1 and D_2 is equivalent to the degree of excitation d of the operator $\hat{T}_d$ which transforms D_1 into D_2 ($D_2 = \hat{T}_d D_1$). This is the number of holes created in D_1 or the number of particles created in D_2.

On line 4 of algorithm 1, $(I^1_{i,\sigma}$ `xor` $I^2_{i,\sigma})$ returns a 64-bit integer with bits set to one where the bits differ between $I^1_{i,\sigma}$ and $I^2_{i,\sigma}$. Those correspond to the positions of the holes and the particles. The `popcnt` function returns the number of non-zero bits in the resulting integer. At line 5, `two_d` contains the sum of the number of holes and particles, so the excitation degree `d` is half of `two_d`.

Algorithm 1: Degree of excitation between D_1 and D_2.

Function `n_excitations`(I^1, I^2)

Input: I^1, I^2: lists of integers representing D_1 and D_2.

Output: d: Degree of excitation.

```
1 two_d ← 0;
2 for σ ∈ {α, β} do
3     for i ← 0 to N_int − 1 do
4         two_d ← two_d+ popcnt (I^1_{i,σ} xor I^2_{i,σ});
5 d ← two_d/2;
6 return d;
```

The `popcnt` instruction was introduced in the hardware of the x86_64 processors with the SSE4.2 instruction set. This instruction has a 3-cycle latency and a 1-cycle throughput independently of the number of bits set to one (here, independently of the number of electrons). The *popcnt* instruction may be generated by Fortran compilers via the intrinsic `popcnt` function.

Identifying the substituted spin-orbitals

Algorithm 2 creates the list of spin-orbital indices containing the holes of the excitation from D_1 to D_2. At line 4, H is is set to a 64-bit integer with ones at the positions of the holes. The loop starting at line 5 translates the positions of those bits to spin-orbital indices as follows: when $H \neq 0$, the index of the rightmost bit of H set to one is equal to the number of trailing zeros of the integer. This number can be obtained by the x86_64 *bsf* (bit scan forward) instruction with a latency of 3 cycles and a 1-cycle throughput, and may be generated by the Fortran `trailz` intrinsic function. At line 7, the spin-orbital index is calculated. At line 8, the rightmost bit set to one is cleared in H.

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0000000011111111 : D1
0010010010101111 : D2
0010010001010000 : D1 xor D2
0000000001010000 : (D1 xor D2) and D1
0010010000000000 : (D1 xor D2) and D2
```

Algorithm 2: List of orbital indices corresponding to holes in the excitation from D_1 to D_2

Function `get_holes`(I^1, I^2);

Input: I^1, I^2: lists of integers representing D_1 and D_2.

Output: Holes: List of positions of the holes.

```
1  for σ ∈ {α, β} do
2      k ← 0;
3      for i ← 0 to N_int − 1 do
4          H ← (I^1_{i,σ} xor I^2_{i,σ}) and I^1_{i,σ};
5          while H ≠ 0 do
6              position ← trailing_zeros(H);
7              Holes[k, σ] ← 1 + 64 × i + position;
8              H ← bit_clear(H, position);
9              k ← k + 1;
10 return Holes;
```

Computing the phase

In our representation, the spin-orbitals are always ordered by increasing index. Therefore, a reordering may occur during the spin-orbital substitution, involving a possible change of the phase.

As no more than two substitutions between determinants D_1 and D_2 give a non-zero matrix element, we only consider in algorithm 3 single and double substitutions. The phase is calculated as $-1^{N_{\text{perm}}}$, where N_{perm} is the number permutations necessary to bring the spin-orbitals on which the holes are made to the positions of the particles. This number is equal to the number of occupied spin-orbitals between these two positions.

We create a bit mask to extract the occupied spin-orbitals placed between the hole and the particle, and we count them using the `popcnt` instruction. We have to consider that the hole and the particle may or may not not belong to the same 64-bit integer. For a double excitation, if the realization of the first excitation introduces a new orbital between the hole and the particle of the second excitation (crossing of the two excitations), an additional permutation is needed.

Algorithm 3: Compute the phase factor of $\langle D_1|\mathcal{O}|D_2\rangle$

Function `GetPhase`(Holes, Particles)

Input: Holes and Particles obtained with alorithm 2

Output: phase $\in \{-1, 1\}$.

```
1  Requires: n_excitations(I^1, I^2) ∈ {1, 2}. Holes and Particles are sorted.
2  nperm ← 0;
3  for σ ∈ {α, β} do
4      n_σ ← Number of excitations of spin σ;
5      for i ← 0 to n_σ − 1 do
6          high ← max(Particles[i, σ], Holes[i, σ]);
7          low ← min(Particles[i, σ], Holes[i, σ]);
8          k ← ⌊high/64⌋;
9          m ← high (mod 64);
10         j ← ⌊low/64⌋;
11         n ← low (mod 64);
12         for l ← j to k − 1 do
13             mask[l] ←not(0);
14         mask[k] ← 2^m − 1;
15         mask[j] ← mask[j] and (not(2^{n+1}) + 1)
16         for l ← j to k do
17             nperm ← nperm+popcnt(I_1^{j,σ} and mask[l]);
18     if (n_σ = 2) and ( Holes[2, σ] < Particles[1, σ] or Holes[1, σ]
       > Particles[2, σ] ) then
19         nperm ← nperm + 1;
20 return −1^nperm;
```

Results

The presented algorithms were implemented in Fortran. Two systems made of 10 000 determinants were benchmarked. One is a water molecule in the cc-pVTZ basis set ($N_{\text{int}} = 2$) and the other is is a Copper atom in the cc-pVDZ basis set ($N_{\text{int}} = 1$). The benchmark consists in comparing each determinant with all the derminants (10^8 determinant comparisons). These comparisons are central in determinant driven calculations, such as the calculation of the Hamiltonian in the determinant basis set. As a practical example, we benchmark the calculation of the one-electron density matrix on the MO basis.

	CPU Time (s)		Average # CPU Cycles	
	AVX	SSE2	AVX	SSE2
H_2O (10 e^-, 105 MOs)				
`n_excitations`	0.33	2.33	10.2	72.6
`get_excitation`	0.60	2.63	18.4	81.4
`get_excitation`, $d=0$			6.2	58.7
`get_excitation`, $d=1$			53.0	126.3
`get_excitation`, $d=2$			88.9	195.5
`get_excitation`, $d>2$			6.7	63.6
Density matrix	0.19	1.23	11.7	75.7
Cu (29 e^-, 49 MOs)				
`n_excitations`	0.17	1.13	5.3	35.2
`get_excitation`	0.28	1.27	8.7	39.1
`get_excitation`, $d=0$			4.9	30.4
`get_excitation`, $d=1$			47.0	88.1
`get_excitation`, $d=2$			78.8	145.5
`get_excitation`, $d>2$			5.5	29.3
Density matrix	0.10	0.63	6.5	38.9
For comparison, on the same system				
L1 latency			4	
L2 latency			12	
32-bit FP division			15	
64-bit FP division			23	
32-bit Integer division			23	
64-bit Integer division			45	
L3 latency			45	
RAM latency			260	

Benchmark system: Intel Xeon CPU E3-1220 @ 3.10GHz.
The source files of this benchmark are available for download here:
https://github.com/scemama/slater_condon