

# General block-tensor library for high-level electronic structure calculations

Evgeny Epifanovsky,<sup>1,2,3</sup> Michael Wormit,<sup>4</sup> Andreas Dreuw,<sup>4</sup> Anna Krylov<sup>1</sup>

**SciDAC Partnership:** Martin Head-Gordon<sup>2,3</sup> (PI), Daniel Haxton,<sup>3</sup> C. William McCurdy,<sup>3</sup> FASTMath (Esmond Ng, Xiaoye Li, Chao Yang), SUPER (Samuel Williams)

<sup>1</sup> Department of Chemistry, University of Southern California, Los Angeles, CA 90089

<sup>2</sup> Department of Chemistry, University of California, Berkeley, CA 94720

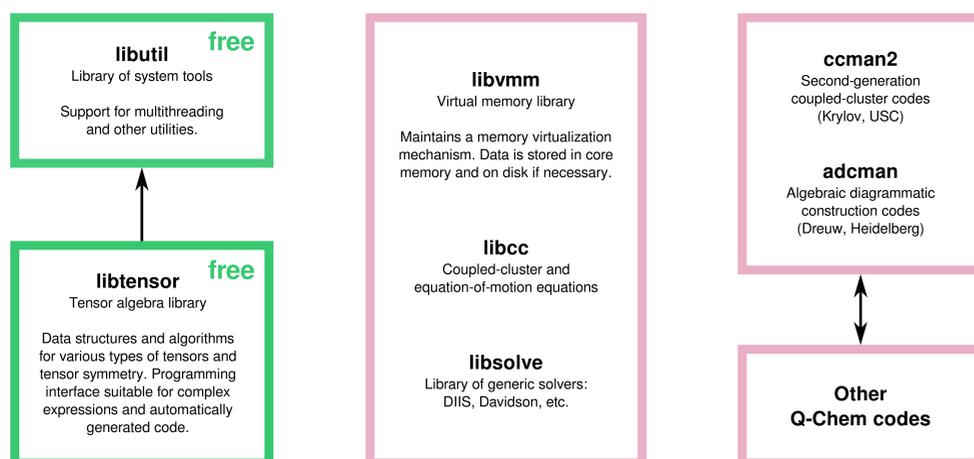
<sup>3</sup> Lawrence Berkeley National Laboratory, Berkeley, CA 94720

<sup>4</sup> Interdisziplinäres Zentrum für Wissenschaftliches Rechnen Ruprecht-Karls Universität Heidelberg, Heidelberg 69120, Germany

**Objective:** to design and implement a set of tools that enable the development of high-level post-Hartree-Fock methods.

**Requirements:** support for multi-core parallelism (and MPI support in the future), support for large tensors, portability, simple programming interface.

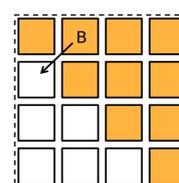
<http://iopshell.usc.edu/downloads/tensor>



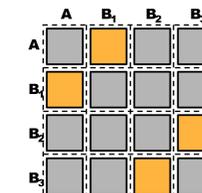
## Block tensors

Large tensors are broken down into smaller blocks, which have a manageable size. The blocks are of the same dimensionality as the original tensor. Parallel divide-and-conquer algorithms are applied to evaluate tensor algebra on these structures.

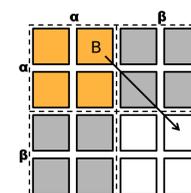
Symmetry of block tensors is recorded as relations between the blocks accompanied by a permutation and a scalar transformation (optional factor and complex conjugate). All tensor algorithms account for symmetry and operate only on non-zero canonical (unique) blocks.



Permutational symmetry connects blocks across a general tensor diagonal. Blocks on the diagonal contain redundant data, but that simplifies algebra algorithms.



Blocks can be assigned labels and multiplication rules that allow or disallow certain label combinations. This general idea is the basis for molecular point group symmetry in tensors.



Groups of blocks with similar tiling patterns can form partitions. Special symmetry rules between whole partitions form restricted HF (spin) symmetry.

## Programming interface

Computational expressions are coded using a simple C++ interface. It is easy to create, read and modify new and existing routines. Low-level details are hidden, which increases the portability of the codes.

Operation	Example
Multiplication by a scalar	$c(i j k) = 2.0 * a(i j k);$
Addition of two tensors	$c(i j k) = a(i j k) + b(k j i);$
Dot (inner) product of two tensors	$c = \text{dot\_product}(a(i j k), b(k j i));$
Direct (outer) product of two tensors	$c(i j k l) = a(i k) * b(j l);$
Contraction of two tensors	$c(i j k l) = \text{contract}(m, a(i j l), b(m l k));$
Elementwise product of two tensors	$c(i j k l) = \text{omult}(k l, a(i k l), b(j k l));$
Direct summation of two tensors	$c(i j k l) = \text{dirsum}(a(i k), b(j l));$
General diagonal of a tensor	$c(i j) = \text{diag}(j k, j, a(i j k));$
Symmetrization	$c(i j) = \text{symm}(i, j, a(i j));$
Antisymmetrization	$c(i j) = \text{asymm}(i, j, a(i j));$

### Example: MP2 energy

$$E_{MP2} = \sum_{ia} t_{ia}^a f_{ia} + \frac{1}{4} \sum_{ijkl} t_{ij}^{ab} (ij|ab), \text{ where } t_{ia}^a = \frac{f_{ia}}{\epsilon_i - \epsilon_a} \text{ and } t_{ij}^{ab} = \frac{(ij|ab)}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}$$

```
double mp2_energy(btensor<2> &t1, btensor<4> &t2,
                 btensor<2> &f_ov, btensor<4> &i_ovov) {
    letter i, j, a, b;
    return dot_product(t1(i|a), f_ov(i|a)) +
           0.25 * dot_product(t2(i|j|a|b), i_ovov(i|j|a|b));
}
```

### Example: CCD amplitudes

```
void ccd_t2_update(btensor<2> &f_oo, btensor<2> &f_vv, ...) {
    letter i, j, k, l, a, b, c, d;
    btensor<2> f1_oo, f1_vv;
    btensor<4> ii_oooo, ii_ovov;

    // Compute intermediates
    f1_oo(i|j) = f_oo(i|j)
                + 0.5 * contract(k|a|b, i_ovov(j|k|a|b), t2(i|k|a|b));
    f1_vv(b|c) = f_vv(b|c)
                - 0.5 * contract(k|l|d, i_ovov(k|l|c|d), t2(k|l|b|d));
    ii_oooo(i|j|k|l) = i_oooo(i|j|k|l)
                      + 0.5 * contract(a|b, i_ovov(k|l|a|b), t2(i|j|a|b));
    ii_ovov(i|a|j|b) = i_ovov(i|a|j|b)
                     - 0.5 * contract(k|c, i_ovov(i|k|b|c), t2(k|j|c|a));

    // Compute updated T2
    t2new(i|j|a|b) = i_ovov(i|j|a|b)
                   + asymm(a, b, contract(c, t2(i|j|a|c), f1_vv(b|c)))
                   - asymm(i, j, contract(k, t2(i|k|a|b), f1_oo(j|k)))
                   + 0.5 * contract(k|l, ii_oooo(i|j|k|l), t2(k|l|a|b))
                   + 0.5 * contract(c|d, i_ovov(a|b|c|d), t2(i|j|c|d))
                   - asymm(a, b, asymm(i, j,
                                     contract(k|c, i_ovov(k|b|j|c), t2(i|k|a|c)));
}
```

## Current and future applications

The tensor library powers a large number of post-Hartree-Fock methods in Q-Chem.

- Coupled-cluster methods: CCD, CCSD, RI-CCSD, CCSD(T).
- Equation-of-motion methods: EOM-EE, EA, IP, SF, DIP.
- Algebraic-diagrammatic construction suite of methods (Michael Wormit, Andreas Dreuw).

### Future developments (in partnership with FASTMath and SUPER)

- Tensors with complex entries: Complex-scaled theories for resonance states (Krylov, McCurdy, Haxton).
- Tensor factorizations and reduced-rank approximations (tensor decompositions): Use Cholesky, canonical, Tucker decompositions to compute CC and EOM amplitudes in attempt to reduce the scaling of calculations with system size.
- Parallelization of existing and future tensor algebra algorithms using MPI + threads. Extension to heterogeneous architectures including CPU with accelerators.

## Timings

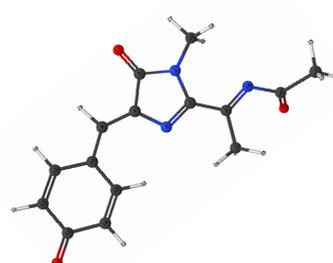
Total time to compute the CCSD energy of the uracil molecule with 6-31G\* basis set (128 basis set functions).



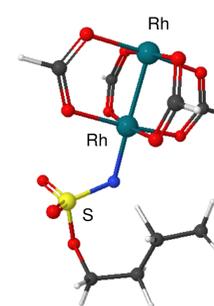
CPU	1	4	8	12	16	20	24
Two 12-core Opterons 6172 (Magny-Cours) 2.1 GHz							
Time (sec.)	7706	2119	1243	1134	861	866	895
Speedup		3.6	6.2	6.8	9.0	8.9	8.6
Two 6-core Xeons X5660 (Gulftown/Westmere) 2.8 GHz							
Time (sec.)	4792	1253	697	557			
Speedup		3.8	6.9	8.6			



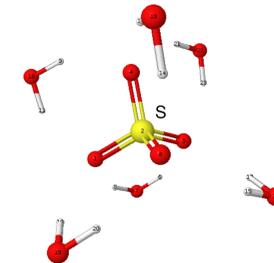
**Nucleobase dimer**  
CCSD/6-31+G\*  
362 BSF  
4 CPUs, 27 hours  
8 CPUs, 17 hours



**Red fluorescent chromophore**  
CCSD/cc-pVTZ  
826 BSF  
8 CPUs, 36 days



**Rhodium complex**  
CCSD(T)/6-31G\*  
(LANL2DZ on Rh, S)  
358 BSF  
8 CPUs, 15 days



**Sulfate-water cluster**  
CCSD(T)/aug-cc-pVDZ  
365 BSF  
8 CPUs, 4 days