

Efficient Transformation and Permutation of Electron Repulsion Integral Quartets

Bachelor Thesis

At a glance

- Implement a data reduction technique useful in quantum chemistry applications
- Use C++ templates and constexpr functions for efficient code generation
- Employ AVX2 and AVX512 for high-performance vectorization
- No prior knowledge of quantum chemistry is required!

Electron Repulsion Integrals (ERIs) are an essential quantity for ab initio molecular dynamics packages such as CP2K. To process them more efficiently, they are grouped into quartet classes $\{abcd\}$, which are laid out as 4D arrays in memory. The size of each array dimension depends on the orbital shells used for a, b, c, d : For example, the s, p, d, f orbital shells in the cartesian basis contain 1, 3, 6, 10 orbitals, respectively, so an $\{dd|fp\}$ quartet contains $6 \times 6 \times 10 \times 3 = 1080$ ERIs. Multiple libraries already exist for computing quartets in the cartesian basis, most prominently Libint. CP2K, however, expects ERIs in the so-called spherical basis, in which the s, p, d, f shells only contain 1, 3, 5, 7 orbitals, respectively. Changing the basis requires a linear transformation applied to each dimension of the cartesian 4D quartet array. Naively, this can be executed in 4 steps:

$$\begin{array}{l}
 q_1(:, b, c, d) \leftarrow T_a q_c(:, b, c, d) \\
 q_2(a, :, c, d) \leftarrow T_b q_1(a, :, c, d) \\
 q_3(a, b, :, d) \leftarrow T_c q_2(a, b, :, d) \\
 q_s(a, b, c, :) \leftarrow T_d q_3(a, b, c, :)
 \end{array}
 \quad
 \begin{array}{c}
 \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \\
 \begin{pmatrix} a_{-2,2} \\ a_{-1,2} \\ a_{0,2} \\ a_{1,2} \\ a_{2,2} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & t_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_{2,5} & 0 \\ t_{3,1} & 0 & 0 & t_{3,4} & 0 & t_{3,6} \\ 0 & 0 & t_{4,3} & 0 & 0 & 0 \\ t_{5,1} & 0 & 0 & t_{5,4} & 0 & 0 \end{pmatrix}}_{T_a} \begin{pmatrix} a_{2,0,0} \\ a_{1,1,0} \\ a_{1,0,1} \\ a_{0,2,0} \\ a_{0,1,1} \\ a_{0,0,2} \end{pmatrix}
 \end{array}$$

Here, T_a, \dots, T_d are sparse transformation matrices for transforming one shell in the cartesian basis to the respective spherical basis; q_c and q_s are the quartets in the cartesian/spherical basis; and q_1, q_2 , and q_3 are the quartets with partially transformed bases. Continuing the previous example, q_s for $\{dd|fp\}$ contains 525 ERIs, whereas the partially transformed q_1 contains 900 ERIs.

In this thesis, the operation described above must be implemented efficiently and elegantly, which poses the following challenges: Firstly, the number of orbitals in a shell are known at compile time, and therefore a specialized function can be generated for each quartet class using template meta-programming. Secondly, the vectorization is non-trivial due to irregular memory access patterns in the sparse vector matrix multiplications. Lastly, libraries such as Libint usually only compute permutationally unique quartets (canonical quartets), where it is left to the user to apply the correct permutation. This step could also be merged into the transformation process.

The overarching goal of this thesis is to build a small library that efficiently supports the transformations on modern x86 processors, and that exposes them via an easy-to-use but high-performance API. This library, given that the results are satisfactory, will then be integrated into CP2K.

Contact:

Johannes Menzel, Office: X1.125, Phone: 05251 60-1741, Mail: johannes.menzel@uni-paderborn.de

For information on theses offered by the HPC group or PC2 please visit:
<https://en.cs.uni-paderborn.de/hpc/teaching/open-theses>