PANACHE: A Modular Approach to Approximate Two-Electron Repulsion Integrals

Benjamin P. Pritchard, T. Daniel Crawford
Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, U.S.A.

Robert M. Parrish, C. David Sherrill
Center for Computational Molecular Science and Technology
School of Chemistry and Biochemistry, Georgia Institute of Technology, Atlanta, Georgia 30332, U.S.A.

Andrew C. Simmonett
Laboratory of Computational Biology, National Heart, Lung, and Blood Institute
National Institutes of Health, Rockville, Maryland 20892, U.S.A.

Justin M Turney
Center for Computational Quantum Chemistry, Department of Chemistry
University of Georgia, Athens, Georgia 30602, U.S.A.

Abstract

The calculation of electron repulsion integrals (ERIs) over nucleus-centered Gaussian-type functions is often among the most computationally intensive tasks of quantum chemical models. The memory required to store these integrals — ca. $\mathcal{O}(N^4)$, where N is the number of basis functions — is often beyond the capacity of the computing hardware, thus necessitating either repeated retrieval from slow, disk-based storage or "on-the-fly" recalculation. A number of approximate ERI methods attempt to overcome this bottleneck by decomposing the full integral matrix into products of tensors of reduced rank [e.g., $\mathcal{O}(N^3)$], which either fit in memory or, at the very least, significantly reduce the quantity of data to be transferred from external storage. However, the robust implementation of these methods requires considerable software infrastructure, such as the ability to compute ERIs with varying basis sets on different centers and out-of-core, integral-direct decomposition algorithms, among others. In this work, two of the most widely-used ERI approximations — density-fitting and Cholesky decomposition — have been implemented in a C++ library called the "Parallel Numerical Approximations in Chemistry Engine" or *PANACHE*. The library is

designed to be modular and portable, and to support interfaces in multiple programming languages.

1 Introduction

Among the most computationally expensive and time-consuming steps in many quantum chemical methods is the generation and manipulation of four-center two-electron repulsion integrals (ERIs),

$$(\mu\nu|\rho\sigma) = \int\int \chi_\mu(\boldsymbol{r}_1)\chi_\nu(\boldsymbol{r}_1)\frac{1}{r_{12}}\chi_\rho(\boldsymbol{r}_2)\chi_\sigma(\boldsymbol{r}_2)d\boldsymbol{r}_1 d\boldsymbol{r}_2, \tag{1}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are the spatial coordinates of electrons one and two, respectively, and the χ_μ are (typically) nucleus-centered contracted Gaussian-type orbitals. The calculation and storage of such ERIs scales poorly [formally as $\mathcal{O}(N^4)$ with respect to the number of basis functions, N], thus requiring either retrieval of the integrals from disk (which is often very slow) or their on-demand recomputation (which can be inefficient, depending on the application). Alternatively, the ERI matrix, which is a four-index tensor, may be decomposed into a product of three-index tensors, *viz.*,

$$(\mu\nu|\rho\sigma) = \sum_Q B^Q_{\mu\nu} B^Q_{\rho\sigma}, \tag{2}$$

where the specific definition of the tensor, B, and the associated auxiliary index, Q, depends on the choice of decomposition method, and the resulting efficiency of the implementation depends strongly on the dimension of the latter. Such decompoaisions obviously reduce the storage requirements as compared to the full-rank ERIs, as the smaller tensors may either be retained completely in memory or retrieved from external disk much more efficiently.

In addition, the expense required in the manipulation and transformation of the integrals [for example, from the atomic orbital (AO) to molecular orbital (MO) basis] is also improved, a fact exploited in a number of reduced-scaling many-body methods, such as...

Furthermore, the mathematical expressions involving the four-index tensors can often be rewritten in terms of their lower-rank counterparts, potentially reducing the scaling of expressions in, for example, second-order perturbation theory or coupled cluster theory calculations.[3] [Need lots more references here.]

Two of the most popular techniques to approximate these tensors are density fitting (also referred to as “resolution of the identity”) and Cholesky decomposition [need original references here], and these have been implemented in a number of program packages. [PSI4, Turbomole, Molcas, Molpro and others]

such as Gaussian,[1] Psi4,[2] MPQC, and others.

Despite the overall gain in efficiency, implementing density fitting or Cholesky decomposition can be tedious and time consuming, with the feeling of 'reinventing the wheel'. In addition, retrofitting existing code can be difficult as the code dealing with integrals is often among the oldest and most heavily-optimized, and therefore least understood, parts of computational chemistry software.

The motivation for *PANACHE* is to create a standalone library that allows for the implementation of these approximate integral techniques in new or existing code with minimal effort, freeing the programmer to concentrate on other, more interesting parts of their code. It also aims to demonstrate the power and flexibility of modular, object-oriented design principles in the computational physical sciences.

PANACHE is based on code originally extracted from the Psi4[2] package, and heavily modified for efficiency and to provide it with an interface usable from other software packages.

1.1 Theory

What follows is a brief outline of density fitting and Cholesky decomposition. For a more detailed derivation and analysis, we refer the reader to the relevant literature.

In density fitting,[3–6] a four-index ERI tensor can be approximated using an auxiliary basis. Starting with the statement of the ERI in terms of the repulsion between two electron densities

$$(\mu\nu|\lambda\sigma) = \int\int \rho_{\mu\nu}(\boldsymbol{r}_1)\frac{1}{r_{12}}\rho_{\lambda\sigma}(\boldsymbol{r}_2)d\boldsymbol{r}_1 d\boldsymbol{r}_2 \tag{3}$$

where μ,ν,λ, and σ are genera atomic orbital indices. The density $\rho_{\mu\nu}$ can be approximated using the auxiliary basis χ_Q and fitting coefficients $d_Q^{\mu\nu}$

$$\tilde{\rho}_{\mu\nu}(\boldsymbol{r}) = \sum_Q^N d_Q^{\mu\nu}\chi_Q(\boldsymbol{r}) \tag{4}$$

$$= \sum_Q^N d_Q^{\mu\nu}(Q| \tag{5}$$

where N represents the number of basis functions in the fitting basis. Commonly, these fitting coefficients are taken to be in the form[4,5]

$$d_Q^{\mu\nu} = \sum_P (\mu\nu|P)[\boldsymbol{J}^{-1}]_{PQ} \tag{6}$$

$$J_{PQ} = \int\int \chi_P(\boldsymbol{r}_1)\frac{1}{r_{12}}\chi_Q(\boldsymbol{r_2})d\boldsymbol{r}_q d\boldsymbol{r}_2 \tag{7}$$

That is, the metric $\boldsymbol{J}$ can be obtained from two-center electron repulsion integrals of the auxiliary basis.

Therefore, the full four-index ERI tensor can be written as a contraction involving the metric (matrix) $\boldsymbol{J}$ and a three-index tensor

$$(\mu\nu|\lambda\sigma) = \sum_P^N d_Q^{\mu\nu}(Q|\lambda\sigma) \tag{8}$$

$$= (\mu\nu|P)[\boldsymbol{J}^{-1}](Q|\lambda\sigma) \tag{9}$$

where $(\mu\nu|P)$ and $(Q|\lambda\sigma)$ are formed from three-center electron repulsion integrals involving the primary and auxiliary basis sets.

One particularly useful form of Eq. 9 involves replacing $\boldsymbol{J}^{-1}$ with the product $\boldsymbol{J}^{-\frac{1}{2}}\boldsymbol{J}^{-\frac{1}{2}}$

and simplifying

$$(\mu\nu|\lambda\sigma) = (\mu\nu|P)[\boldsymbol{J}^{-\frac{1}{2}}]_{PQ}[\boldsymbol{J}^{-\frac{1}{2}}]_{QR}(R|\lambda\sigma) \tag{10}$$

$$= \sum_{Q} B_{ab}^{Q} B_{cd}^{Q} \tag{11}$$

where, keeping in mind the permutational symmetry of the three-index integrals and the fact that P, Q, and R are only summation indices

$$B_{ab}^{Q} = \sum_{P}^{N} (\mu\nu|Q)[\boldsymbol{J}^{-\frac{1}{2}}]_{PQ} \tag{12}$$

The advantage of such a form is the elimination of the need for storing the metric separately. In addition, this allows for on-the-fly contraction of $(\mu\nu|Q)$ with $\boldsymbol{J}^{-\frac{1}{2}}$ as it is generated.

In practice, the auxiliary basis is much larger than the primary basis set, however in most cases the storage required for the three-index tensor is much less than its four-index counterpart. The B_{pq}^{Q} tensor (labeled $\boldsymbol{Q_{so}}$ within *PANACHE* the code) can also be transformed via the MO coefficient matrix (or blocks thereof) to form $\boldsymbol{Q_{mo}}$, $\boldsymbol{Q_{oo}}$, $\boldsymbol{Q_{ov}}$, and $\boldsymbol{Q_{vv}}$ for MO, occupied-occupied, occupied-virtual, and virtual-virtual blocks, respectively. This transformation now scales $O(N_{\text{aux}}N^2)$, rather than a $O(N^4)$ transformation on the four-index ERI tensor.

In Cholesky decomposition,[7,8] the two-electron tensor (written as a matrix with rows and columns corresponding to composite indices $a = (\mu\nu)$ and $b = (\lambda\sigma)$, respectively) can be written as a product of a lower triangular matrix $\boldsymbol{L}$ and its transpose

$$(a|b) = \boldsymbol{V} = \boldsymbol{L} \cdot \boldsymbol{L}^T \tag{13}$$

where elements of $\boldsymbol{L}$ are determined by

$$L_{ii} = \left(V_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{\frac{1}{2}} \tag{14}$$

$$L_{ij} = \frac{1}{L_{jj}} \left(V_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right), \qquad (j+1) \leq i \leq M \tag{15}$$

where M represents the index of the last for of $\boldsymbol{V}$. The generated $\boldsymbol{L}$ matrix is analogous to a three-index tensor, similar to B from density fitting, with M rows and with columns corresponding to the AOs of the basis set. If the indicies of $\boldsymbol{L}$ are ordered where

$$L_{ii}^2 \geq V_{nn} - \sum_{k=1}^{i-1} L_{nk}^2, \qquad (j+1) \leq n \leq M \tag{16}$$

The iterations may be stopped if $L_{ii}^2 \leq \delta$ where *delta* represents the desired accuracy. In this way, the number of rows of $\boldsymbol{L}$ are less than M.

Since $\boldsymbol{L}$ is analogous to the three-index tensor from density fitting, it may be manipulated in an identical manner, including transformations to the MO basis.

2 Description and Features

PANACHE is written in C++11, with interfaces for use from C and Fortran code. It is made to be portable – it utilizes the CMake[9] build system and is relatively self-contained with few external dependencies. The library enforces strict separation of interface and implementation, thereby making it modular and extensible. For example, it is relatively easy to add new storage classes or different integral backends. Both density fitting and Cholesky decomposition share nearly identical interfaces, allowing easy adoption of either or both methods in new or existing code.

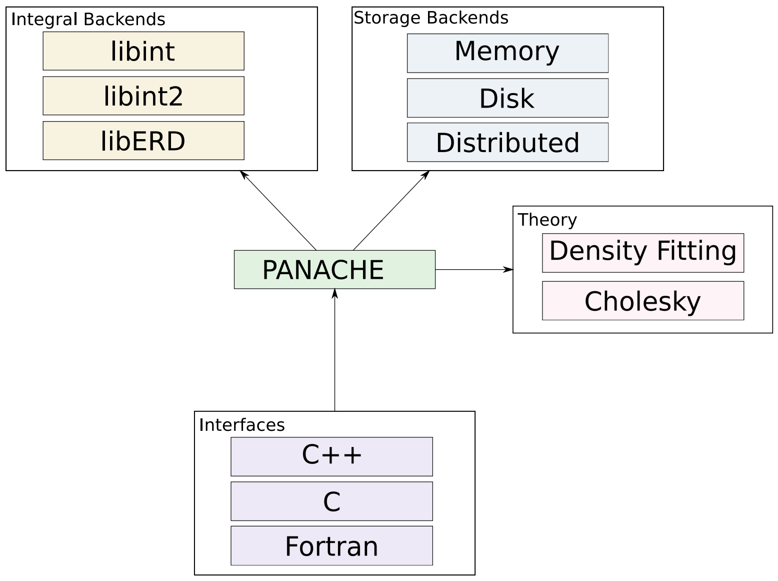

Figure 1: Overview of the structure of the *PANACHE* library

2.1 Library Structure

The central class of the *PANACHE* library is the abstract *ThreeIndexTensor* class (see Figure 2), whose main purposes are to provide the common interface to density fitting and Cholesky decomposition, and to implement common operations (such as handling of MO coefficient matrices and related transformations). In general, this class is responsible for the interface for almost all transformations and other tensor operations, as well as providing a method for transferring data back to the calling program. Classes derived from *Three-IndexTensor* (DFTensor and CHTensor for density fitting and Cholesky decomposition, respectively) are in general only responsible for generation of the three-index, AO-basis $\boldsymbol{Q_{so}}$ tensor, where $\boldsymbol{Q_{so}}$ is either the B tensor from density fitting or L from Cholesky decomposition (see Section 1.1)

After the generation, $\boldsymbol{Q_{so}}$ can be treated the same, independent of whether it is from density fitting or Cholesky decomposition. This includes reordering, transformation (to MO basis, etc), storage, and retrieval by the calling program. In that sense, it is obvious that the presented class hierarchy is a natural fit for these theoretical methods.

Actual storage of three-index tensors is implemented by the *StoredQTensor* and its

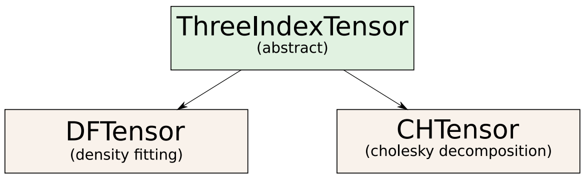

Figure 2: The *ThreeIndexTensor* class and its derived classes.

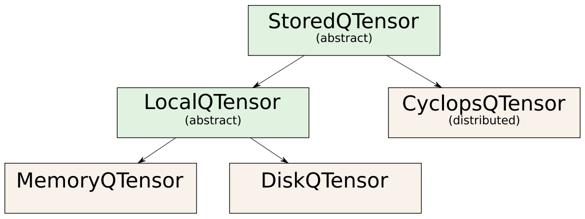

Figure 3: The *StoredQTensor* class and its derived classes

derived classes (Figure 3). Objects of classes derived from the *StoredQTensor* base class are stored in the *ThreeIndexTensor* class.

Any calculated three-index quantities (from either density fitting or Cholesky decomposition, and in the AO or MO basis) can be stored in memory or on disk, with the determination made at runtime. The tensors may also be kept after the calculation is finished for later reuse or analysis, and transparently moved from disk to memory. This is all handled in the *LocalQTensor* class and its derived classes. For distributed calculation and storage, the Cyclops Tensor Framework[10] (CTF) can be used via the *CyclopsQTensor* class. It should be noted, however, that because of the differences in distributed vs single-node storage, aspects of the generation of $\boldsymbol{Q_{so}}$ as well as details concerning the MO transformations are actually contained in the storage classes.

Calculation of two- and three-center ERIs are handled by popular integral packages; currently, *PANACHE* supports libint[11–13] (both v1 and v2) and libERD[14] (distributed with *PANACHE*), although adding others is possible. No further linking should be required for code already utilizing libint. Extending *PANACHE* to use other integral code is also possible.

Parallelization is achieved primarily via OpenMP (with appropriate compiler support), although the library is certainly usable when compiled without it. As mentioned previously, distributed computation of both density fitting and Cholesky decomposition is experimentally supported via CTF.

2.2 Interface

The interface to *PANACHE* includes passing in the required information (such as the basis set and density fitting basis set, or the Cholesky decomposition cutoff), transformation to specified the MO basis (or subsets such as occupied-occupied), specification of storage class (disk, memory), and retrieval of the required three-index tensors for use by the calling code.

Because of the diversity of existing computational chemistry software, the interface to *PANACHE* was made as flexible and as extensible as possible. The core interface to the library is its C++ interface, however C and Fortran bindings are included. The Fortran interface is written in Fortran 2003, allowing for transparent conversion between C and Fortran data types and name-mangling conventions (that is, uppercase function names and trailing underscores). It should be noted that although the Fortran interface is written Fortran 2003, software written in older Fortran standards will still be able to function with *PANACHE*. In addition, due to the relatively straightforward C interface, it should be possible to use it with any language that supports C bindings, such as Python.

One major issue with interfacing with existing codes is the difference in ordering with a particular Gaussian (spherical or cartesian) shell. To help with this *PANACHE* includes the ability to reorder its integrals. When possible, this is done efficiently by only reordering the MO coefficients, rather than any generated three-index tensors. This reordering capability currently supports PSI4 (default), GAMESS, and Dalton ordering, but is easily extended if needed. Internally, *PANACHE* uses the same ordering as that used in Psi4.

Lastly, for existing software where implementing parsing and storage of the auxiliary basis set needed for density fitting may be difficult, the library supports constructing the auxiliary basis from a given Gaussian-formatted basis set file and information describing the molecule.

To test the library, implementations of calculation of MP2 correlation energy[3] were successfully developed for the PSI4,[2] GAMESS,[15,16] and DALTON[17,18] packages. THe MP2 correlation energy was successfully calculated using both both density fitting and Cholesky decomposition utilizing the geometry, basis set, and MO coefficients passed from the calling code.

Details on how the interfaces are used can be found in the *PANACHE* documentation.

2.3 Conclusion and Future work

The *PANACHE* library provides a modular, efficient approach to calculation of approximate electron repulsion integrals via density fitting and Cholesky decomposition. It is designed to be flexible and able to be incorporated into existing computational chemistry software via C++, C, and Fortran bindings.

Parallelization is achieved via OpenMP; support for distributed computational is currently experimental and is planned to be expanded in the future to other tensor libraries.

3 Availability and Documentation

***** Correct? ***** The *PANACHE* library is available at GitHub at ***** address ***** and is licensed under the GNU General Public License (GPL).

Extensive bulding and usage documentation, as well as code documentation, is available at http://www.psicode.org/panache.php.

4 Acknowledgements

This research was funded by the U.S. National Science Foundation (Grant No. ACI-1147794)

References

[1] Frisch, M. J. et al. Gaussian 09, Revision A.02. Gaussian, Inc., Wallingford CT, 2009.

[2] Turney, J. M. et al. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2012**, *2*, 556–565.

[3] Werner, H.-J.; Manby, F. R.; Knowles, P. J. *J. Chem. Phys.* **2003**, *118*, 8149–8160.

[4] Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *Int. J. Quantum Chem.*

[5] Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *J. Chem. Phys.* **1979**, *71*, 4993–4999.

[6] Whitten, J. L. *The Journal of Chemical Physics* **1973**, *58*, 4496–4501.

[7] Røeggen, I.; Johansen, T. *J. Chem. Phys.* **2008**, *128*, 194107.

[8] Beebe, N. H. F.; Linderberg, J. *Int. J. Quantum Chem.*

[9] http://www.cmake.org.

[10] Solomonik, E.; Matthews, D.; Hammond, J.; Demmel, J. *Cyclops Tensor Framework: reducing communication and eliminating load imbalance in massively parallel contractions*; 2012.

[11] Valeev, E. F.; Fermann, J. T. http://www.files.chem.vt.edu/chem-dept/valeev/software.html.

[12] Valeev, E. F.; Fermann, J. T. A library for the evaluation of molecular integrals of many-body operators over Gaussian functions. http://libint.valeyev.net/, 2013.

[13] Valeev, E. F. A library for the evaluation of molecular integrals of many-body operators over Gaussian functions. http://libint.valeyev.net/, 2014.

[14] Flocke, N.; Lotrich, V. *J. Comp. Chem.* **2008**, *29*, 2722–2736.

[15] Gordon, M. S.; Schmidt, M. W. *Theory and Applications of Computational Chemistry, the first forty years*.

[16] Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comp. Chem.* **1993**, *14*, 1347–1363.

[17] Dalton, a molecular electronic structure program, Release Dalton2015.X (2015), see http://daltonprogram.org.

[18] Aidas, K. et al. *WIREs Comput. Mol. Sci.* **2015**, *4*, 269–284.