A practical view on linear algebra tools

Evgeny Epifanovsky
University of Southern California
University of California, Berkeley
Q-Chem

September 25, 2014

What is Q-Chem?

Established in 1993, first release in 1997.

Software

Q-Chem	3.0 (2006)
	4.0 (2012)
	4.1 (2013)
	4.2 (2014)

Thousands of users

Y. Shao et al., Mol. Phys., in press (2014), DOI:10.1080/00268976.2014.952696
A.I. Krylov and P.M.W. Gill, WIREs Comput. Mol. Sci. 3, 317–326 (2013)

What is Q-Chem?

Established in 1993, first release in 1997.

Software

Q-Chem 3.0 (2006)
4.0 (2012)
4.1 (2013)
4.2 (2014)

Thousands of users

Platform

Supported infrastructure for state-of-the-art quantum chemistry

Free of charge and open source for developers

157 contributors (Q-Chem 4)

Y. Shao et al., Mol. Phys., in press (2014), DOI:10.1080/00268976.2014.952696
A.I. Krylov and P.M.W. Gill, WIREs Comput. Mol. Sci. 3, 317–326 (2013)

What is Q-Chem?

Established in 1993, first release in 1997.

Software

Q-Chem 3.0 (2006)
4.0 (2012)
4.1 (2013)
4.2 (2014)

Thousands of users

Platform

Supported infrastructure for
state-of-the-art quantum chemistry

Free of charge and open source
for developers

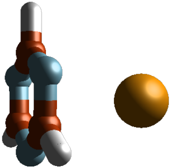
157 contributors (Q-Chem 4)

Q-CHEM

Pleasanton, CA

Y. Shao et al., Mol. Phys., in press (2014), DOI:10.1080/00268976.2014.952696
A.I. Krylov and P.M.W. Gill, WIREs Comput. Mol. Sci. 3, 317–326 (2013)

Electronic structure model



1. Separate electrons and nuclei

   Nuclei become point charges, electrons are a quantum system

2. Choose a discretization scheme

   Introduce atomic orbitals

3. Choose type of wavefunction (or density functional)

   Collapses the dimensionality from 3N to a reasonable number

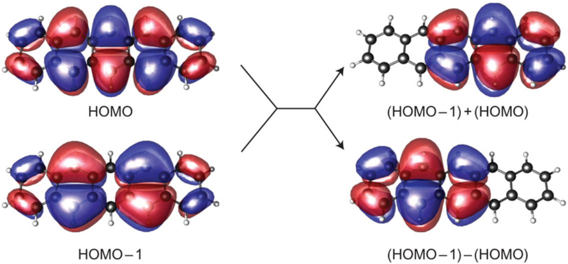
   First choice is mean-field (Hartree–Fock or Kohn–Sham)

4. Solve for parameters of wavefunction

   HF or KS molecular orbitals

Origin of dense and sparse objects

Atomic orbitals	Non-orthogonal	Local	Sparse
Molecular orbitals	Orthonormal	Delocalized	Dense
Localized MOs	Both	Local	Sparse



J and K matrices

Making J and K matrices in HF and DFT

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \qquad K_{\lambda\nu} = \sum_{\mu\sigma} (\mu\nu|\lambda\sigma) P_{\mu\sigma}$$

$$(\mu\nu|\lambda\sigma) \equiv \int \phi_\mu(r_1)\phi_\nu(r_1)\frac{1}{r_{12}}\phi_\lambda(r_2)\phi_\sigma(r_2)\; dr_1 dr_2$$

Nominal scaling of computational cost for J and K is N^4.

Making J and K matrices in HF and DFT

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \qquad K_{\lambda\nu} = \sum_{\mu\sigma} (\mu\nu|\lambda\sigma) P_{\mu\sigma}$$

$$(\mu\nu|\lambda\sigma) \equiv \int \phi_\mu(r_1)\phi_\nu(r_1)\frac{1}{r_{12}}\phi_\lambda(r_2)\phi_\sigma(r_2)\; dr_1 dr_2$$

Nominal scaling of computational cost for J and K is N^4.

For J-matrix:

1. Define significant pairs $(\mu\nu|$ and $|\lambda\sigma)$ – $O(N)$
2. Compute integrals – $O(N^2)$ to $O(N)$
3. Contract with density – $O(N^2)$ to $O(N)$

Making J and K matrices in HF and DFT

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \qquad K_{\lambda\nu} = \sum_{\mu\sigma} (\mu\nu|\lambda\sigma) P_{\mu\sigma}$$

$$(\mu\nu|\lambda\sigma) \equiv \int \phi_\mu(r_1)\phi_\nu(r_1)\frac{1}{r_{12}}\phi_\lambda(r_2)\phi_\sigma(r_2)\ dr_1 dr_2$$

Nominal scaling of computational cost for J and K is N^4.

For J-matrix:

1. Define significant pairs $(\mu\nu|$ and $|\lambda\sigma)$ – $O(N)$
2. Compute integrals – $O(N^2)$ to $O(N)$
3. Contract with density – $O(N^2)$ to $O(N)$

For K-matrix repeat for each $(\mu\nu|$:

1. Compute $(\widetilde{\mu\nu}|\lambda\sigma)$ – $O(N)$
2. Contract with density – $O(N^2)$

Contractions in coupled-cluster theory

$$t_{ij}^{\lambda\sigma} = \sum_{ab} t_{ij}^{ab} C_{\lambda a} C_{\sigma b}$$

$$\sum_{\lambda\sigma} \left[(\mu\nu|\lambda\sigma) - (\lambda\nu|\mu\sigma)\right] t_{ij}^{\lambda\sigma} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) t_{ij}^{\lambda\sigma} - \sum_{\lambda\sigma} (\lambda\nu|\mu\sigma) t_{ij}^{\lambda\sigma}$$

Nominal scaling of the steps is $O^2 N^4$.

Including sparsity reduces scaling of J-type and K-type contractions to $O^2 N^2$ and $O^2 N^3$, respectively.

Resolution of the identity approximation

$$\begin{aligned}(\mu\nu|\lambda\sigma) &\approx \sum_{PQ} C_{\mu\nu}^{P}(P|Q)C_{\lambda\sigma}^{Q} \\ &= \sum_{PQ}(\mu\nu|P)(P|Q)^{-1}(Q|\lambda\sigma)\end{aligned}$$

$$(\mu\nu|P) = \sum_{Q}(P|Q)C_{\mu\nu}^{Q}$$

(no approximation is made if auxiliary basis is complete)

$$B_{\mu\nu}^{Q} = \sum_{P}(\mu\nu|P)(P|Q)^{-1/2}$$

$$(\mu\nu|\lambda\sigma) \approx \sum_{Q} B_{\mu\nu}^{Q}B_{\lambda\sigma}^{Q}$$

Make K matrix with RI

$$K_{\lambda\nu} = \sum_{\mu\sigma Q} B^{Q}_{\mu\nu} B^{Q}_{\lambda\sigma} P_{\mu\sigma}$$

- How to factorize the equation (choose intermediates)?
- To minimize computations?
- To stay within given memory constraint?

AO-MO transformation

Integral transformation step in MP2 and RI-MP2

$$(ia|jb) = \sum_{\mu\nu\lambda\sigma} (\mu\nu|\lambda\sigma) C_{\mu i} C_{\nu a} C_{\lambda j} C_{\sigma b}$$

$$(ia|P) = \sum_{\mu\nu} (\mu\nu|P) C_{\mu i} C_{\nu a}$$

- With given memory constrains how to choose batch size and intermediates?

Linear algebra in many dimensions

Coupled-cluster doubles (CCD) equations

$$
\begin{aligned}
D_{ij}^{ab} &= \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b \\
T_{ij}^{ab} D_{ij}^{ab} &= \langle ij||ab\rangle + \mathcal{P}_-(ab)\left(\sum_c f_{bc} t_{ij}^{ac} - \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{kl}^{bd} t_{ij}^{ac}\right) \\
&- \mathcal{P}_-(ij)\left(\sum_k f_{jk} t_{ik}^{ab} + \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{jl}^{cd} t_{ik}^{ab}\right) \\
&+ \frac{1}{2}\sum_{kl}\langle ij||kl\rangle t_{kl}^{ab} + \frac{1}{4}\sum_{klcd}\langle kl||cd\rangle t_{ij}^{cd} t_{kl}^{ab} + \frac{1}{2}\sum_{cd}\langle ab||cd\rangle t_{ij}^{cd} \\
&- \mathcal{P}_-(ij)\mathcal{P}_-(ab)\left(\sum_{kc}\langle kb||jc\rangle t_{ik}^{ac} - \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{lj}^{db} t_{ik}^{ac}\right)
\end{aligned}
$$

$$
\mathcal{P}_-(ij)A_{ij} = A_{ij} - A_{ji}
$$

Tensor expressions for CCD

```
void ccd_t2_update(...) {

    letter i, j, k, l, a, b, c, d;
    btensor<2> f1_oo(oo), f1_vv(vv);
    btensor<4> ii_oooo(oooo), ii_ovov(ovov);

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

    //  Compute updated T2
    t2new(i|j|a|b) =
          i_oovv(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_vvvv(a|b|c|d), t2(i|j|c|d))
        - asymm(a, b, asymm(i, j,
            contract(k|c, ii_ovov(k|b|j|c), t2(i|k|a|c))));
}
```

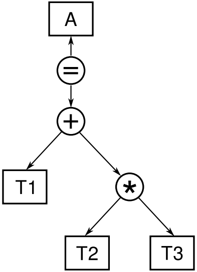
Evaluation of tensor expressions

1. Convert expression to abstract syntax tree (AST)
2. Optimize and transform AST with given constraints

$$A_{ij} = T^1_{ij} + \sum_k T^2_{ik} T^3_{kj}$$

```
A(i|j) =
         T1(i|j) +
         contract(k, T2(i|k), T3(k|j));
```

→



Evaluation of tensor expressions

1. Convert expression to abstract syntax tree (AST)
2. Optimize and transform AST with given constraints
3. Evaluate expression following optimized AST

Back-end:

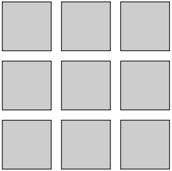
- Shared memory threaded model (single node)[3]
- Distributed memory parallel model (via CTF)
- Replicated memory parallel model

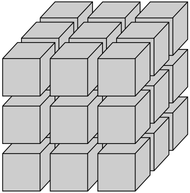
[3] E.Epifanovsky et al., J. Comput. Chem. 34, 2293–2309 (2013)

Block tensors in libtensor

Three components:

- Block tensor space: dimensions + tiling pattern.
- Symmetry relations between blocks.
- Non-zero canonical data blocks.





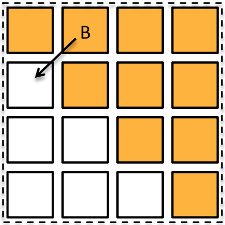
Block tensors in libtensor

Three components:

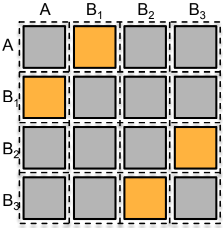
- Block tensor space: dimensions + tiling pattern.
- Symmetry relations between blocks.
- Non-zero canonical data blocks.

Symmetry:

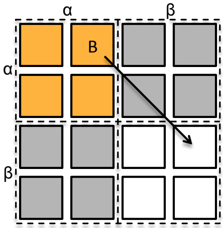
$$S : SB_i \mapsto (B_j, U_{ij})$$



Permutational



Point group



Spin

Perturbation theory correction

Perturbation theory

$$\sum_{ijab} \frac{\left[(ia|jb) - (ib|ja)\right]^2}{\Delta_{iajb}} \qquad \sum_{ijkabc} \frac{t_{ijk}^{abc}\tilde{t}_{ijk}^{abc}}{\Delta_{iajbkc}}$$

$$t_{ijk}^{abc} = \mathcal{P}(ijk)\mathcal{P}(abc)\left(\sum_d t_{ij}^{cd}\langle kd||ab\rangle + \sum_l t_{lk}^{ab}\langle ij||lc\rangle\right)$$

$$\tilde{t}_{ijk}^{abc} = t_{ijk}^{abc} + \mathcal{P}(ijk)\mathcal{P}(abc)\left(t_i^c\langle kj||ab\rangle + f_i^c t_{kj}^{ab}\right)$$

$$\mathcal{P}(ijk)a_{ijk} = a_{ijk} - a_{jik} - a_{ikj} - a_{kji} + a_{jki} + a_{kij}$$

- How to partition the numerator to minimize computational cost and satisfy memory constraints?

Summary

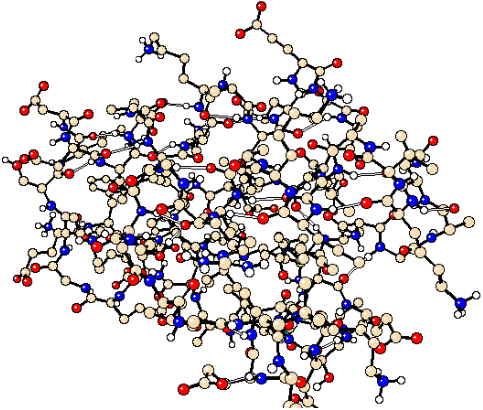
- Most of the problems are sparse multi-dimensional linear algebra problems
- For many of those cases there exist a mapping to a dense two-dimensional problem
- Almost all new problems contain sparse many-tensor contractions, for which general optimal algorithms have not been developed

Open problems

- Given a contraction of multiple sparse tensors, what is the best way to factorize it into pairwise contractions?
- How to optimally compute a tensor expression satisfying memory constraints?

Scalability and software requirements

Scaling to large problems



How well are existing electronic structure methods equipped to benefit from large-scale HPC systems?

Do they really need to be massively parallel?

Technical requirements

Linear algebra tools are just one component in a large ecosystem:

- Routines should have no side-effects
- Routines should be thread-safe and otherwise parallel-friendly
- User should be able to designate resources for each operation. How to pass this information?

Technical requirements

Linear algebra tools are just one component in a large ecosystem:

- Routines should have no side-effects
- Routines should be thread-safe and otherwise parallel-friendly
- User should be able to designate resources for each operation. How to pass this information?

Good example: original BLAS

Technical requirements

Linear algebra tools are just one component in a large ecosystem:

- Routines should have no side-effects
- Routines should be thread-safe and otherwise parallel-friendly
- User should be able to designate resources for each operation. How to pass this information?

Good example: original BLAS

Bad example: modern BLAS-OpenMP

Acknowledgments

- **Dr. Michael Wormit** (Heidelberg)
- Edgar Solomonik (UCB → ETH Zurich)
- Dr. Arik Landau, Dr. Tomasz Kus, Kirill Khistyaev, Dr. Prashant Manohar (USC)
- Xintian Feng, Dr. Dmitry Zuev (USC)
- Prof. Anna I. Krylov (USC),
  Prof. Martin Head-Gordon (UC Berkeley)

- DOE SciDAC collaboration `http://mee-scidac.org/`
- NSF SICM^2 collaboration `http://www.s2i2.org/`

Q-CHEM