Fast direct large-scale MCSCF code for Segmented and General Contraction Basis Sets

Alexander A. Granovsky

Laboratory of Chemical Cybernetics, M.V. Lomonosov Moscow State University, Moscow, Russia

September 14, 2005

Large-scale MCSCF

- Main steps of MCSCF iteration ("unfolded two step" type)
  - Integral transformation
  - CI problem
  - DM1 & DM2 calculation
  - Orbital improvement
    - Multiple different strategies based on linear, quasi-linear, or quadratic minimization methods
- Large basis sets, medium size active spaces
  - Performance limited by integral transformation
- Large active spaces, small basis set
  - Performance limited by CI matrix diagonalization

Memory requirements

- Integral transformation
  - $C \cdot N^3$
  - $\underline{C \cdot N^2}$
- CI matrix diagonalization:
  - $C \cdot N_{det}$
- Orbitals improvement
  - up to $C \cdot (N^2 + N_{det})^2$ (“folded one- & two-step”)
  - $C \cdot N^4$
  - $C \cdot N^3$
  - $\underline{C \cdot N^2}$ (example: quasi-Newton type methods)

Classification of transformed 2-e integrals

- Orbital types:
  - o - doubly occupied (core)
  - a - active space (valence)
  - v - virtual
  - p, q, r, s - arbitrary
- (pq|rs) types:
  - (aa|aa) & Fock matrix - required always (CI step)
  - (aa|rs) - required for calculation of the diagonal part of orbital Hessian and quasi-Newton orbital improvement methods
  - (o+a,q|rs) - required for full orbital Hessian and true Newton-type orbital improvement step (integrals with three virtual indices are not needed)

Method selection for large-scale MCSCF

- Memory requirements: $C \cdot N^2$ =>
  - Dedicated low-memory demands integral transformation code
- Quasi-Newton orbital improvement step
  - Fast
  - Modest memory demands
  - Requires only small subset of transformed integrals =>
    - simpler and more efficient integral transformation

Main problem

- Special efficient integral transformation code for (aa|rs)-type integrals with:
  - Quadratic memory demands
  - Ability to handle both SC and GC basis sets efficiently
  - High parallel mode scalability

Integral transformation basics

- $(pq|rs) = \Sigma_\mu \Sigma_\nu \Sigma_\lambda \Sigma_\sigma C_{p\mu} C_{q\nu} C_{r\lambda} C_{s\sigma} (\mu\nu|\lambda\sigma)$
- Usually considered as a sequence of four quarter-transformations:
  - $(p\nu|\lambda\sigma) = \Sigma_\mu C_{p\mu}(\mu\nu|\lambda\sigma)$
  - $(pq|\lambda\sigma) = \Sigma_\nu C_{q\nu}(p\nu|\lambda\sigma)$, etc…
- Alternative approach:
  - $(pq|\lambda\sigma) = \Sigma_\mu \Sigma_\nu C_{q\nu} C_{p\mu} (\mu\nu|\lambda\sigma)$
    - $D^{(pq)}_{\mu\nu} = C_{q\nu} C_{p\mu}$
    - $J^{(pq)}_{\lambda\sigma} = (pq|\lambda\sigma) = \Sigma_{\mu\nu} D^{(pq)}_{\mu\nu} (\mu\nu|\lambda\sigma)$
- Reminiscence: Fock Matrix
  - $F_2(D) = J(D) - K(D)$
    - $J_{\lambda\sigma} = \Sigma_{\mu\nu} (\mu\nu|\lambda\sigma) D_{\mu\nu}$

Approach comparison

- Standard approach (four sequential quarter-transformations):
  - Asymptotically $n_a N^2$ operations
  - Straightforward to utilize the eightfold permutation symmetry of ERIs
  - N^3 memory demands
  - Limited parallel scalability
- Alternative approach:
  - Asymptotically $n_a^2 N^2$ operations
  - Straightforward to utilize the eightfold permutation symmetry of ERIs
  - N^2 memory demands
  - High degree of scalability
  - Implementation based on our direct Fock matrix construction code

Alternative approach: pros and cons

- Pros
  - For small active spaces, n_a is small => additional overhead due to worse asymptotic can be neglected as dominant part of the calculations is evaluation of ERIs in AO basis
  - Modest memory requirements
  - Allows direct generalization to GC case based on our approach to Fock matrix construction for GC-type basis sets
  - High level of intrinsic parallelism
- Cons
  - For larger active spaces, n_a^2 is significantly larger than n_a => additional overhead due to different asymptotic is considerable
  - For GC-type basis sets, additional overhead is even more serious if using our strategy of Fock-like matrix builds.

Optimal strategy

- Small active spaces:
  - use alternative approach for both SC and GC-type basis sets
- Larger active spaces:
  - use something else (but not the standard approach in its straightforward implementation)

Standard way modification

- Why standard way requires so much memory?
  - Because it utilizes eightfold permutation symmetry of ERIs:
    - $C_{p\mu}(\mu\nu|\lambda\sigma) \rightarrow (p\nu|\lambda\sigma)$
    - $C_{p\nu}(\nu\mu|\lambda\sigma) \rightarrow (p\mu|\lambda\sigma)$
    - $C_{p\lambda}(\lambda\sigma|\nu\mu) \rightarrow (p\sigma|\nu\mu)$
    - $C_{p\sigma}(\sigma\lambda|\nu\mu) \rightarrow (p\lambda|\nu\mu)$
- Solution:
  - use only fourfold permutation symmetry
    - $C_{p\mu}(\mu\nu|\lambda\sigma) \rightarrow (p\nu|\lambda\sigma)$
    - $C_{p\nu}(\nu\mu|\lambda\sigma) \rightarrow (p\mu|\lambda\sigma)$
  - Compute $(p\nu|\lambda\sigma)$ for all $\mu\nu$ and fixed $\lambda\sigma$, then perform second half-transformation (matrix multiplication) $(pq|\lambda\sigma) = \Sigma_\nu C_{q\nu}(p\nu|\lambda\sigma)$ ($\lambda\sigma$ fixed) and store

Modified vs. standard way

- Larger overhead due to ERI reevaluation
  - Not significant for large active spaces
- Requires much less memory (the same amount as the alternative approach)
- Has the same parallel scaling properties as the alternative approach
- Has the same good $n_a N^2$ operations count asymptotic as the standard way
- Allows efficient generalization for GC-type basis sets based on our approach to Fock matrix construction

Generalization for GC basis sets

- $(pq|rs) = \Sigma_\mu \Sigma_\nu \Sigma_\lambda \Sigma_\sigma C_{p\mu} C_{q\nu} C_{r\lambda} C_{s\sigma} (\mu\nu|\lambda\sigma)$
- $(\mu\nu|\lambda\sigma) = \Sigma_M \Sigma_N \Sigma_L \Sigma_S \hat{C}_{\mu M} C_{\nu N} C_{\lambda L} C_{\sigma S} (MN|LS)$
  - $(pq|rs) = \Sigma_\mu \Sigma_\nu \Sigma_\lambda \Sigma_\sigma C_{p\mu} C_{q\nu} C_{r\lambda} C_{s\sigma} \Sigma_M \Sigma_N \Sigma_L \Sigma_S C_{\mu M} C_{\nu N} C_{\lambda L} C_{\sigma S} (MN|LS)$
  - $(pq|rs) = \Sigma_M \Sigma_N \Sigma_L \Sigma_S (\Sigma_\mu C_{\mu M} C_{p\mu}) (\Sigma_\nu C_{\nu N} C_{q\nu}) (\Sigma_\lambda C_{\lambda L} C_{r\lambda}) (\Sigma_\sigma C_{\sigma S} C_{s\sigma}) (MN|LS)$
  - => new transformation matrix is simply $\mathbf{C}^* C$
- It is not efficient for standard way as N would be replaced by much larger N_{prim}, dramatically increasing memory demands and computational costs
- It is much more efficient for modified way and MCSCF due to
  - different memory asymptotic
  - small values of n_a required for MCSCF integral transformation

Thank you for your attention!