Parallel spin-orbit CI

Jeffrey L. Tilson*, Walter C. Ermler**, and Russell M. Pitzer+

*Argonne National Laboratory, ** Stevens Institute of Technology, +The Ohio State University

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, Field Work Proposal No. 56694, B&R No. KJ-01-01-03-0.
Research Effort

- Actinide containing systems. (U, Pu, etc.)
  - Requires high-accuracy
    - c / mass effects
    - spin-orbit (SO) effects
    - structural and dynamic correlation effects.
  - Software must be modified.
    - Calculations require the fastest/biggest machines.

- Chemistry
  - f-f transition energies (+assignments)
  - Large ionization pots. (large formal charges)
Approach.

• Build models using modern formalism.
  – ARECPs-incorporates important c / mass effects in core.
  – SO operator rigorously included. (ARECP-RECP)
  – In a form useable in standard (spin-orbital) CI code
  – Permits valence correlation.

• Build upon available software.
  – F90 and C languages.
  – Global Arrays (GA) for distributed data.
  – Parallel I/O (ChemIO) for distributed out-of-core work.
  – Start with available “legacy” code (CIDBG.X)
    • COLUMBUS system
Conventional SOCI

- Massive, sparse eigenvector problem.
- Symmetric and Real*
- Conventional, i.e., “Direct” approach
  - Construct whole matrix.
  - Store H on disk
- Solve iteratively.
  - (Davidson’s method)

  *Can be made real for select point groups

Conduct H in double-group basis.
- Precompute configuration list.
- Include all configs satisfying total “J” (neither LS not jj coupling)
- Store coupling information in memory/disk
  - fine-grain access.

Eigenvectors (~40-50)
- Blocks of degeneracies
- all roots under 2-3 eV
**Spin-Orbit CI (SOCl)**

\[
H = \sum_{\mu} h_{\mu} - \sum_{\mu>\nu} \frac{1}{r_{\nu\nu}} + \sum_{l} O_{l} \xi_{l}(r) \left( l \cdot s \right) O_{l}
\]

\[
h_{\mu} = \frac{-1}{2} \nabla_{\mu} + \sum_{\alpha}^{N} \left( \frac{-Z^{\text{eff}}}{r_{\alpha\mu}} + U_{\alpha}^{\text{ARECP}} \right)
\]

H. time-independent hamiltonian operator.
\(\mu,\nu\) index valence electrons. \(\alpha\) indexes nuclei.
\(U^{\text{ARECP}}\) is \(j\) averaged-RECPs.
\(\xi_{l}(r) = 2\Delta U^{\text{RECP}}/(2l+1)\)
\(O_{l}\) formally projects \(\xi(r)\) back into \(l|m_{l}\>\).

Rigorous inclusion of spin-orbit terms. AO integrals only in standard basis required.
Sample hamiltonian

Zero elements colored green.
Growth of spin-orbit CI problem with wavefunction size

Number of double-groups (millions)

Current state-of-the-art 1998

"tough" (RS6000) calculation (1996)

"state-of-the-art" calculation (1996)

"Easy" calculation (1998)
Parallel SOCI

• CIDBG.X algorithm changes.
  – Wavefunction description.
    • Precompute couplings in concurrent blocks
    • Store in distributed memory memory*
  – Hamiltonian construction
    • Static load-balancing scheme.
    • Construct in concurrent blocks
    • Store in distributed “storage”
  – Eigenvectors
    • Borrow NWChem parallel Davidson.

• Global Arrays 2.3 (GA)
  – Constructs and manages distributed-data space.
  – Permits portable imp.
  – Little perceived performance penalty.
  – Simple implementation

*fine grained access. Works very well on the T3E. High latency on the SPs requires the application to chunk data.
Parallel H storage

• Method One - Fastest
  – Store H in aggregate memory using GA
  – Simple distribution changes

• Limited problem sizes.
  – approx. 8 million double-groups. (theoretical)
  – Largest to-date: 3 million

• Method Two - Larger problems.
  – Store H onto disks using ChemIO.
  – Exclusive access model (EAF), no striping.
  – Each node writes “its” part of H to its local-disk.

• I/O times slower than memory access.
SOCl Procedure. (typical)

Typical SOCI Run

- AO Integrals
  - ARGOS.X (parallel)
  - Optimum orbitals
    - SCFPQ.X sequential
      - spherical ave.
    - MCSCF.X sequential
      - flexibility
  - Wavefunction conf.
    - CGDBG.X sequential
    - NWChem Parallel

Generate MOs

- TRAN.X sequential
  - NWChem Parallel
- Generate SOCI Massively Parallel
  - Memory-cached sparse H in memory
    - Disk-cached (parallel I/O)
  - Final Analysis

ARECPs/RECPs are generated in a previous series of steps.
SOCl Performance

Method 1 - Memory cached

Parallel performance of SOCl on the Cray T3E.

Memory-cache version.

$H$ construction time.
Disk I/O SOCI.

- **Initial tests**
  - Replaced H memory store with H disk store.
  - A simple approach
  - Permits restart
  - BUT! Subsequent read performance poor.
    - I/O Access too fine-grained

- **Prefetching algorithm**
  - Increased performance
  - Uses same algorithm.
  - Portable.

- **Method.**
  - Explicit (application level) prefetching
  - H store still performing single column access.
Application prefetching.

• Attempt to increase read performance.
  – by minimizing latency
  – achieving better B.Width utilization.

• Method
  – H columns are contiguous on disk.
  – Hence, Read several H columns at one time.

• Method.
  – Simple additional interface to ChemIO.
  – Specify maximum prefetched rows.
  – Fully in-core if possible.

• Experiments
  – How much should we prefetch?
  – how good can we get?
Hamiltonian I/O step

- Performance good.
  - Write 1 column at a time. (non-zeros only)
  - No “chunking” of writes.
  - H sufficiently complicated that 1-column accesses are good.

- No additional modifications (yet) required.
Eigenvector I/O step

• Problem step.
  – Lots of I/O latencies.
  – H columns sparse (10K)
    • Poor utilization of BW
  – Little work per column read.
  – Many vectors/poor guesses/Lots of iterations.

• Requires prefetching or equivalent

LiBe system
Number of elements is 473476
Number of nodes is 16
Total columns is 11050

NERSC T3E
OSC T3E

25 columns (4% of local cols.)

Data is fully cached in local memory

In-memory version (no Disk I/O at all)
Prefetching summary.

• Hamiltonian construction.
  – Two times slower than in-core version.
  – Still room for improvement.
  – Good for now.

• Matrix-vector products.
  – 5-8 times slower per product.
    • (nominal prefetching)
  – Still less time than H construction.
  – Lots of new things to try.
Future Plans

• Further optimizations and tuning of SOCI.

• Semi-direct approach.
  - Determine distribution of work w.r.t. stored H blocks.
  - Build this into integrated storage and load description.

• Incorporate dynamic load-balancing.

• Explore MPI-2 implementation.

• Interface/Merge ideas into NWChem.

• SOCI analytic gradients
Acknowledgements.

- Acknowledgments
  - DOE/MICS
- Thanks to
  Pacific Northwest National Laboratory
  Argonne National Laboratory/MCS and CHM.
- NERSC and the Ohio Supercomputer Center.
Select WWW sites and References


COLUMBUS: http://www.itc.univie.ac.at/~hans/Columbus/columbus_homep.html.


This list is intended to be representative and not complete.