Progress in MFDn: Mathematics and Computer Science

Philip Sternberg
in collaboration with
Esmond Ng
Chao Yang
James Vary
Pieter Maris
Masha Sosonkina
Hung Le
Progress Since Last Year

- Parallel speedup of basis space generation
- Hierarchical scheme for computing sparsity of Hamiltonian
- Use OpenMP to take advantage of multicore architectures
- Hybrid storage scheme for Hamiltonian
- Integration of MFDn with optimization techniques
- Improved inner loop efficiency
Parallel Basis Generation

To balance load, basis states are assigned to processors cyclically mod $n_{diag}$

Testing a basis state for $m_j$, parity, and oscillator quanta conditions takes much more time than getting the next odometer-ordered state

New algorithm divides number of validity tests by $n_{diag}$

Time to generate basis states comparable to reading from disk
Finding Nonzeros

- nonzeros
- potentially nonzero blocks
- zero blocks

Tradeoff between many fine blocks and few coarse blocks
Multilevel Blocking (New Algorithm)

Any number of levels can be used
Performance Results

Time to compute sparsity:

<table>
<thead>
<tr>
<th></th>
<th>one level</th>
<th>multiple levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^6$He (small example)</td>
<td>180 seconds</td>
<td>90 seconds</td>
</tr>
<tr>
<td>$^{12}$C, Nmax = 8, 2-body</td>
<td>$\sim$ 1 hour</td>
<td>$\sim$ 13 minutes</td>
</tr>
<tr>
<td>$^{16}$O, Nmax = 8, 2-body</td>
<td>$\sim$ 2 hours</td>
<td>$\sim$ 20 minutes</td>
</tr>
<tr>
<td>$^{14}$F, Nmax = 8, 2-body</td>
<td>$\sim$ 8 hours ?</td>
<td>$\sim$ 35 minutes</td>
</tr>
</tbody>
</table>

Time to compute sparsity grows as $O(\text{nnz})$ ($\sim O(n^{1.5})$), rather than $O(n^2)$. Thus, as problems get sparser, the advantage of multiple levels increases.
Performance Improvement in Separate Parts of MFDn

\[ ^{13}\text{C} \text{ Chiral NN+NNN} -- N_{\text{max}} = 6 \text{ Basis Space} \]

- MFDn - V10-B05
- MFDn - V12-B00
- MFDn - V12-B01

Optimal Time (secs) on Franklin 4950 pe's

- Setup Basis
- Evaluate H
- 500 Lanczos
- Suite Observ
- I/O
- TOTAL

Graph showing performance improvement in separate parts of MFDn.
Using Threads for Multicore

Many current supercomputers are multicore (XT4, BG/P)

Cores/Node are increasing, so to achieve best performance, all codes will need to take advantage of multiple cores, using hybrid of distributed and shared memory model

Using OpenMP, have observed following initial speed-ups on quad-core system (Jaguar)

- \( \sim 3 \times \) speedup to evaluate matrix elements
- \( \sim 2 \times \) speedup to perform matrix-vector multiply
- \( \sim 3 \times \) speedup to evaluate suite of observables

Further work needed to integrate into production code
Hybrid Storage of Hamiltonian

We can store **dense blocks** as contiguous values (filling in with some zeros), with integers indicating location and size.

\[
A = \begin{bmatrix}
1.0 & 2.0 & 3.0 \\
4.0 & 5.0 & 6.0 \\
7.0 & 8.0 & \\
9.0 & 1.0 & 0.0 \\
& & 2.0
\end{bmatrix}
\]

Many nonzeros are in dense blocks, using much less memory to store the Hamiltonian and taking less time to perform matvec.
Many Dense Blocks
Dense Blocks Broken Up by Cyclic Assignment
Future work

- Integrate load-balanced reorthogonalization with PARPACK
  - Current PARPACK-enabled MFDn does not use all processors to reorthogonalize
  - Will modify PARPACK to use MFDn’s distribution scheme for Lanczos vectors so PARPACK will exhibit same scaling properties as current production code
- Investigate alternative eigensolvers (LOBPCG)
- Alternative partitioning strategy to keep dense blocks together using same hierarchical scheme as for computing sparsity
- Write highly tuned variant of MFDn for cases with dim \(10^7 - 10^8\) as will be used for external fields
- Engineer modifications to code as production scale problem size increases
Papers


