High Performance Shell Model Calculations

Mihai Horoi,
Department of Physics, Central Michigan University, Mount Pleasant, Michigan 48859, USA

Support from NSF grant PHY-0244453 is acknowledged
p - $10^2$ - 1960’s
sd - $10^5$ - 1980’s
pf - $10^9$ - 1990’s
pf$_{5/2}g_{9/2}$ - $10^{10}$ - 2005

Example: $^{76}\text{Sr}$

$\beta_{\text{g.s.}} = 0.4$
PRL 92, 232501

pf$_{5/2}g_{9/2}$ dimension

11,090,052,440

One Major Shell

Nuclear drip lines

JINA/MSU
February 15, 2006
Nuclear Shell Model

$0 \hbar \omega$

$1 \hbar \omega$

$(0 + 2) \hbar \omega$

$(1 + 3) \hbar \omega$

$\vec{m} = [m_1, m_2, m_3, ...] : p$

$H = \sum_k \epsilon_k a_k^+ a_k + \frac{1}{2} \sum_{klmn} V_{klmn} a_k^+ a_l^+ a_n a_m$

$| i(MT_z) > = |00101010010001011100100000 >$

$10p \& 10n \ in \ fp \ \begin{pmatrix} 20 \\ 10 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} = 34 \times 10^9$

$d = 2 (2j + 1)$

$| \alpha > = \sum_i C_i^\alpha | i >$

$\sum_j < i | H | j > C_j^\alpha = E_\alpha C_i^\alpha$
Why Huge Shell Model Dimension?

Table 1. Properties of $^{48}\text{Cr}$ in different valence spaces

<table>
<thead>
<tr>
<th>$^{48}\text{Cr}$</th>
<th>$(f_{7/2})^8$</th>
<th>$(f_{7/2}p_{3/2})^8$</th>
<th>$(fp)^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q($2^+$) (e.fm$^2$)</td>
<td>0.0</td>
<td>-23.3</td>
<td>-23.8</td>
</tr>
<tr>
<td>E($2^+$) (MeV)</td>
<td>0.63</td>
<td>0.44</td>
<td>0.80</td>
</tr>
<tr>
<td>E($4^+$/E($2^+$)</td>
<td>1.94</td>
<td>2.52</td>
<td>2.26</td>
</tr>
<tr>
<td>BE2($2^+ \rightarrow 0^+$) (e$^2$.fm$^4$)</td>
<td>77</td>
<td>150</td>
<td>216</td>
</tr>
<tr>
<td>B(GT)</td>
<td>0.90</td>
<td>0.95</td>
<td>3.88</td>
</tr>
</tbody>
</table>

A. Poves and F. Nowacki, unpublished

Can we describe shape transition and shape coexistence using the shell model?
Shell Model Codes

1. OXBASH: J or JT projection
2. Oslo Shell Model: m-scheme
   - m-scheme dimensions $\sim 10^8$
3. Central Michigan Shell Model (CMichSM): m-scheme
4. ANTOINE (Strasbourg): m-scheme
5. NATHAN: J projection
6. MFD (Iowa): m-scheme
7. RedStick (Livermore): m-scheme
8. MSHELL (Tokyo): m-scheme

Lanczos alg.: $(H, x_0) \rightarrow Hx_0, H(Hx_0), H(H(Hx_0)), \ldots$ (Krylov space)

JINA/MSU
February 15, 2006
Salient Features of m-scheme

- Number of matrix elements per row is quasi constant: e.g. in $fp$ model space is ~ 500 in average
  - Amount of work is $O(N)$ and not $(N^2)$ as is in projected methods
  - Matrix file size increases linearly with the m-scheme dimension
  - Lanczos vectors can be stored in SINGLE PRECISION

- Lanczos vectors **must** be stored to:
  - Maintain orthogonality
  - Recover the eigenvectors
- Hamiltonian matrix could be stored totally or partially ($pn$)

JINA/MSU
February 15, 2006
Effective JT Projection in CMichSM

• Start with basis $|M=J \ T_z=T>$

• Diagonalize the modified Hamiltonian

$$H' = H + \eta J^2 + \gamma T^2$$

• Calculate modified energies

$$E(JT) = E'(JT) - \eta J(J + 1) - \gamma T(T + 1)$$

• Similar to center-of-mass “purification”

$$H'' = H + \beta(H_{CM} - 3/2\hbar \omega)$$
Hardware Needs

• Memory
  – at least 2 Lanczos vectors must be kept in memory: $\text{dim} = 10^9 \Rightarrow 8$ GB RAM

• I/O
  – at least # iterations Lanczos vectors must be kept on external storage and read: #it=100 $\Rightarrow$ 400 GB have to be stored and read
    – Normal hard drives read at about 40 MB/s $\Rightarrow 10^4$ s
    – Solution: use high performance I/O striped systems: RAID0 + Fiber/Ultra320 SCSI/SAS

• Processing power (parallel computers)
  – Many processors, shorter times with efficient algorithms
Parallelization

- **Amdahl’s Law:**
  - $S$ - speedup
  - $\gamma$ - the parallel fraction of the code


\[
S(N) = \frac{T_1}{T_N} = \frac{T_1}{\gamma T_1/N + (1-\gamma)T_1 + T_{comm}}
\]

- $\eta$ - imbalance factor
Distributed Memory : MPI

• Simplest way is keeping a copies of the Lanczos vector on each processor

• Use Collective Communications:
  – MPI_Reduce
  – MPI_AllGatherV
  – MPI_ScatterV

• Caveat: waste of memory ➔ use shared memory machine, e.g.: green
Shared Memory: OpenMP

- Efficient use of memory
- One has to update Lanczos vector atomically: $v_2 = H v_1$

```plaintext
do i=1,no_of_rows
read (k,ham,idx)
ap=v1(i)
ap=0.0e0
  do j=1,k
    indk=idx(j)
ap=ap+v1(indk)*ham(j)
!$omp atomic
  v2(indk)=v2(indk)+api*ham(j)
  enddo  ! on k
  v2(i)=ap
  enddo  ! on i
```