Big Calculations in Little Boxes: the BIGSTICK shell model code

Collaborators:

W. Erich Ormand, Lawrence Livermore

Plamen G. Krastev, SDSU/Harvard

Hai Ah Nam, SDSU/ Oak Ridge

Collaborator-in-training:

Micah Schuster, SDSU

Institute for Nuclear Theory – Sept 2012

The Basic Problem

The basic *science question* is to model detailed quantum structure of many-body systems, such the structure of an atomic nucleus.

The algorithms described today are best applied to many body systems with
(a) two "species" (protons and neutrons, or +1/2 and -1/2 electrons)
(b) single-particle basis states with good rotational symmetry (j, m)

To answer this, we solve *Schrödinger's equation*:

$$\hat{\mathbf{H}}|\Psi\rangle = E|\Psi\rangle$$

* **H** is generally a very large matrix – dimensions up to 10^{10} have been tackled.
* **H** is generally very sparse.
* We usually only want a few low-lying states

The Key Ideas

Basic problem: find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Rightarrow$ Lanczos algorithm
fundamental operation is *matrix-vector multiply*

Despite sparsity, nonzero matrix elements can require TB of storage

Only a fraction of matrix elements are unique; **most are reused.**
Reuse of matrix elements understood through *spectator* particles.

Reuse can be **exploited using exact factorization**
enforced through *additive/multiplicative quantum numbers*

The Basic Problem

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Rightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

We want to solve *Schrödinger's equation*:

$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \sum_{i<j} V(\vec{r}_i - \vec{r}_j) \right) \Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3 \ldots) = E\Psi$$

or

$$\hat{\mathbf{H}}|\Psi\rangle = E|\Psi\rangle$$

The Basic Problem

Find extremal eigenvalues of very large, very sparse Hermitian matrix

→ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

This differential equation is too difficult to solve directly

$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \xcancel{\sum_{i<j} V(\vec{r}_i - \vec{r}_j)} \right) \Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3 \ldots) = E\Psi$$

so we use the matrix formalism

$$\hat{\mathbf{H}} |\Psi\rangle = E |\Psi\rangle$$

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Longrightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

$$|\Psi\rangle = \sum_\alpha c_\alpha |\alpha\rangle \qquad H_{\alpha\beta} = \langle\alpha|\hat{\mathbf{H}}|\beta\rangle$$

$$\sum_\beta H_{\alpha\beta} c_\beta = E c_\alpha \quad \text{if} \quad \langle\alpha|\beta\rangle = \delta_{\alpha\beta}$$

so we use the matrix formalism

$$\hat{\mathbf{H}}|\Psi\rangle = E|\Psi\rangle$$

Nuclear Hamiltonian: $$\hat{H} = \sum_i -\frac{\hbar^2}{2M}\nabla_i^2 + \sum_{i<j} V(r_i, r_j)$$

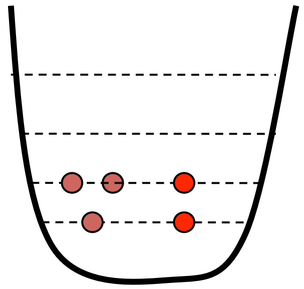
Solve by diagonalizing **H** in a basis of many-body states.

$$\sum_B H_{AB} \mathrm{v}_B = E_A \mathrm{v}_A \qquad H_{AB} = \langle A | \hat{H} | B \rangle$$

What do we use for the many-body basis states { | A> ? ?

Nuclear Hamiltonian: $\hat{H} = \sum_i -\frac{\hbar^2}{2M}\nabla_i^2 + \sum_{i<j} V(r_i, r_j)$

Solve by diagonalizing **H** in a basis of many-body states.
The many-body states are *Slater determinants*, or
anti-symmeterized products of single-particle wfns.



The single-particle states are defined by a single-particle potential *U(r)* (such as harmonic oscillator or Hartree-Fock)

Maria Mayer

At this point one generally goes to occupation representation:

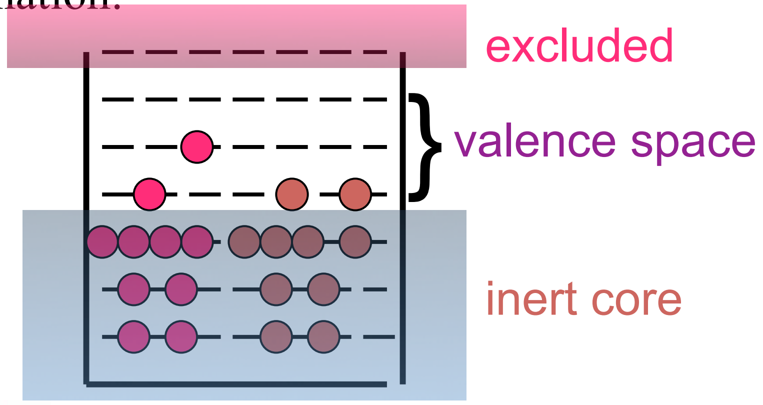
$$\hat{H} = \sum_i \boxed{\varepsilon_i} \hat{a}_i^+ \hat{a}_i + \frac{1}{4}\sum_{ij\,kl} \boxed{V_{ij\,kl}} \hat{a}_i^+ \hat{a}_j^+ \hat{a}_l \hat{a}_k$$

single-particle energies

two-body matrix elements

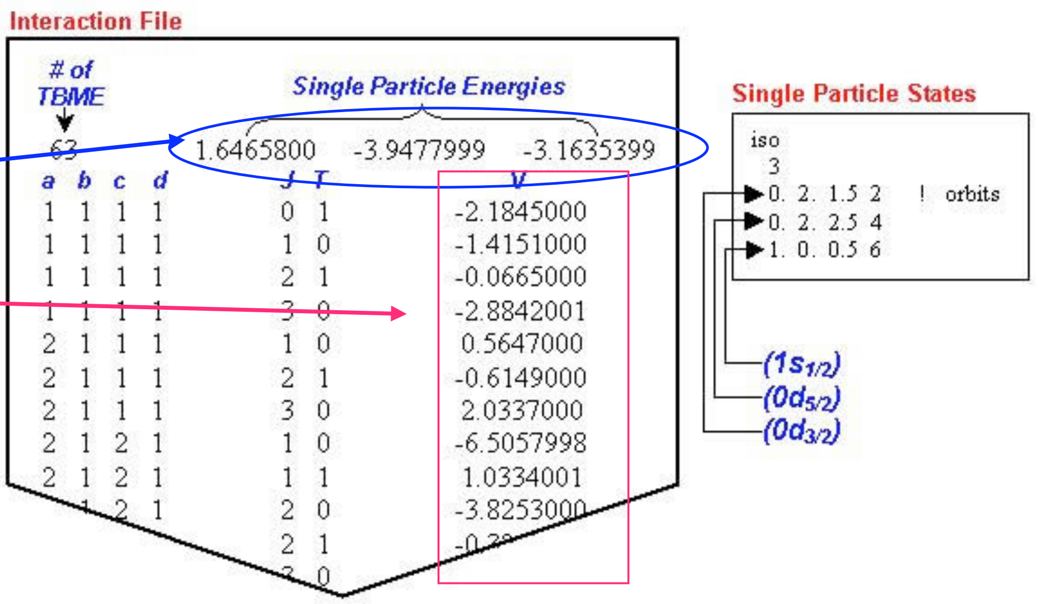
When running a fermion shell model code (e.g. MFD, BIGSTICK), one enters the following information:

(1) The single-particle valence space (such as *sd* or *pf*); assumes inert core



(2) The many-body model space (number of protons and neutrons, truncations, etc.)

(3) The interaction:
single-particle energies
and
two-body matrix elements
$V_{JT}(ab,cd)$


Interaction File
# of TBME
Single Particle Energies
Single Particle States

63 1.6465800 -3.9477999 -3.1635399

a	b	c	d	J	T	V
1	1	1	1	0	1	-2.1845000
1	1	1	1	1	0	-1.4151000
1	1	1	1	2	1	-0.0665000
1	1	1	1	3	0	-2.8842001
2	1	1	1	1	0	0.5647000
2	1	1	1	2	1	-0.6149000
2	1	1	1	3	0	2.0337000
2	1	2	1	1	0	-6.5057998
2	1	2	1	1	1	1.0334001
[illegible]	1	2	1	2	0	-3.8253000
[illegible]	[illegible]	[illegible]	[illegible]	2	1	-0.[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	[illegible]

```
iso
 3
0. 2. 1.5 2    !  orbits
0. 2. 2.5 4
1. 0. 0.5 6
```

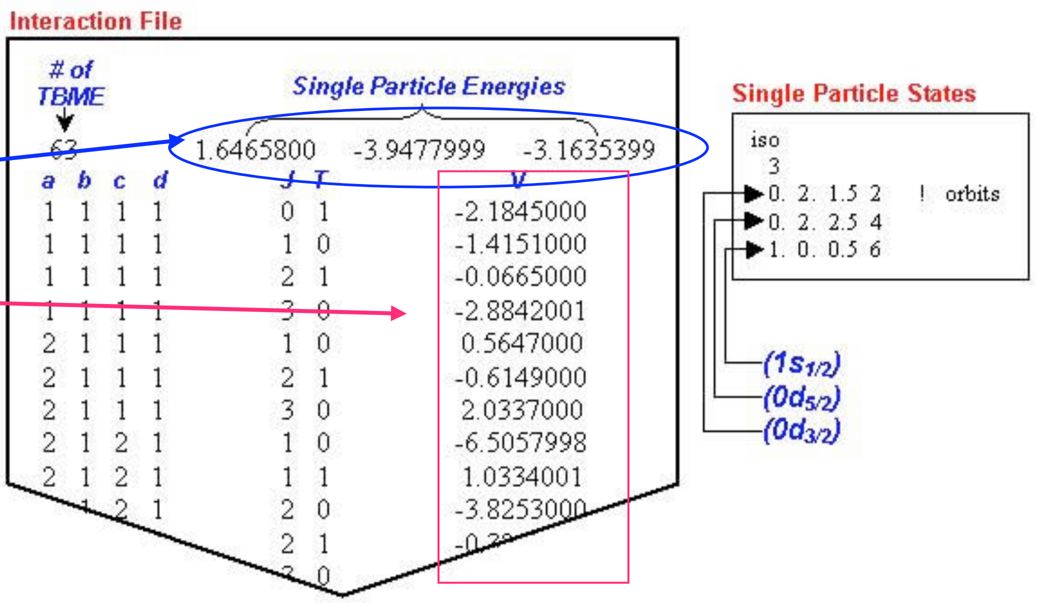
$(1s_{1/2})$
$(0d_{5/2})$
$(0d_{3/2})$

The two-body matrix elements *in principle but not in practice* depend on the single-particle wfns:

$$\langle ab;JT|\hat{H}|cd;JT\rangle = \int d^3r \int d^3r' \varphi_a{}^*(r)\varphi_b{}^*(r') V(r,r')\left(\varphi_c(r)\varphi_d(r') - \varphi_c(r')\varphi_d(r)\right)$$

But only the final number is read in!

(3) The interaction:
single-particle energies
and
two-body matrix elements
$V_{JT}(ab,cd)$



```
 63     1.6465800   -3.9477999   -3.1635399
 1  1  1  1        0  1      -2.1845000
 1  1  1  1        1  0      -1.4151000
 1  1  1  1        2  1      -0.0665000
 1  1  1  1        3  0      -2.8842001
 2  1  1  1        1  0       0.5647000
 2  1  1  1        2  1      -0.6149000
 2  1  1  1        3  0       2.0337000
 2  1  2  1        1  0      -6.5057998
 2  1  2  1        1  1       1.0334001
    1  2  1        2  0      -3.8253000
                   2  1      -0.3
                   3  0
```

```
iso
 3
0. 2. 1.5 2    !  orbits
0. 2. 2.5 4
1. 0. 0.5 6
```

Summary:

The Schrödinger eqn has become a matrix eigenvalue equation

$$\sum_B H_{AB} v_B = E_A v_A$$

One chooses a basis of approx10^4 - 10^{10} states

Key point: Once a basis is chosen, the two-body interaction is reduced to integrals between single-particle states and is stored as a list of real numbers (the *two-body matrix elements*)

A *shell model program* then computes the *many-body matrix elements* from the *two-body matrix elements* and solves for eigenvalues/vectors.

The Basic Problem

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Rightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

$$H_{\alpha\beta} = \langle \alpha | \hat{\mathbf{H}} | \beta \rangle$$

* **H** is generally a very large matrix – dimensions up to 10^{10} have been tackled.

* **H** is generally very sparse.

* We usually only want a few low-lying states

$\Rightarrow$ Lanczos algorithm!

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Longrightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

Standard algorithm to obtain *all* eigenvalues of a real, symmetric matrix **A**: <u>Householder</u>

Find orthogonal matrix **U** such that $\mathbf{U}^T \mathbf{A} \mathbf{U} = \mathbf{B}$, a tridiagonal matrix

The Lanczos algorithm is similar, in that it also uses an orthogonal matrix to take **A** to a tridiagonal matrix **B**.....

Lanczos algorithm!

The Basic Problem

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\Rightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

$$\mathbf{A}\vec{v}_1 = \alpha_1\vec{v}_1 + \beta_1\vec{v}_2$$

$$\mathbf{A}\vec{v}_2 = \beta_1\vec{v}_1 + \alpha_2\vec{v}_2 + \beta_2\vec{v}_3$$

$$\mathbf{A}\vec{v}_3 = \beta_2\vec{v}_2 + \alpha_3\vec{v}_3 + \beta_3\vec{v}_4$$

$$\mathbf{A}\vec{v}_4 = \beta_3\vec{v}_3 + \alpha_4\vec{v}_4 + \beta_4\vec{v}_5$$

Lanczos algorithm!

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

$\longrightarrow$ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

$$
\begin{aligned}
\mathbf{A}\vec{v}_1 &= \alpha_1\vec{v}_1 + \beta_1\vec{v}_2\\
\mathbf{A}\vec{v}_2 &= \beta_1\vec{v}_1 + \alpha_2\vec{v}_2 + \beta_2\vec{v}_3\\
\mathbf{A}\vec{v}_3 &= \beta_2\vec{v}_2 + \alpha_3\vec{v}_3 + \beta_3\vec{v}_4\\
\mathbf{A}\vec{v}_4 &= \beta_3\vec{v}_3 + \alpha_4\vec{v}_4 + \beta_4\vec{v}_5
\end{aligned}
$$

matrix-vector multiply

Lanczos algorithm!

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

I need to quickly cover:

- How the basis states are represented
- How the Hamiltonian operator is represented
- Why most matrix elements are zero
- Typical dimensions and sparsity

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the basis states are represented

This differential equation is too difficult to solve directly

$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \sum_{i<j} V(\vec{r}_i - \vec{r}_j)\right)\Psi(\vec{r}_1,\vec{r}_2,\vec{r}_3\ldots) = E\Psi$$

Can only really solve 1D differential equation

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr^2} + U(r)\right)\phi_i(r) = \varepsilon_i\phi_i(r)$$

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the basis states are represented

Can only really solve 1D differential equation

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}+U(r)\right)\phi_i(r)=\varepsilon_i\phi_i(r) \quad \Longrightarrow \quad \left\{\phi_i(\vec{r})\right\}$$

Single-particle wave functions labeled by, *e.g.*, *n, j, l, m*

Atomic case: 1s, 2s, 2p, 3s, 3p, 3d *etc*

Nuclear: $0s_{1/2}$, $0p_{3/2}$, $0p_{1/2}$, $0d_{5/2}$, $1s_{1/2}$, $0d_{3/2}$, *etc*

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the basis states are represented

Can only really solve 1D differential equation

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}+U(r)\right)\phi_i(r)=\varepsilon_i\phi_i(r) \Rightarrow \left\{\phi_i(\vec{r})\right\}$$

Product wavefunction ("Slater Determinant")

$$\Psi(\vec{r}_1,\vec{r}_2,\vec{r}_3\ldots)=\phi_{n_1}(\vec{r}_1)\phi_{n_2}(\vec{r}_2)\phi_{n_3}(\vec{r}_3)\ldots\phi_{n_N}(\vec{r}_N)$$

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the basis states are represented

Product wavefunction ("Slater Determinant")

$$\Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3 \ldots) = \phi_{\boxed{n_1}}(\vec{r}_1)\phi_{\boxed{n_2}}(\vec{r}_2)\phi_{\boxed{n_3}}(\vec{r}_3)\ldots\phi_{\boxed{n_N}}(\vec{r}_N)$$

Each many-body state can be *uniquely* determined by a list of "occupied" single-particle states = "occupation representation"

$$\left|\alpha\right\rangle = \hat{a}^{+}_{n_1}\hat{a}^{+}_{n_2}\hat{a}^{+}_{n_3}\ldots\hat{a}^{+}_{n_N}\left|0\right\rangle$$

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

"occupation representation" $$\left|\alpha\right\rangle = \hat{a}^{+}_{n_1}\hat{a}^{+}_{n_2}\hat{a}^{+}_{n_3}\ldots\hat{a}^{+}_{n_N}\left|0\right\rangle$$

"creation operator"

$$\hat{H} = \sum_{ij} T_{ij}\hat{a}^{+}_{i}\hat{a}_{j} + \frac{1}{4}\sum_{ijkl} V_{ijkl}\hat{a}^{+}_{i}\hat{a}^{+}_{j}\hat{a}_{l}\hat{a}_{k}$$

motion of a single particle
("one-body operator")

interaction of two particles
("two-body operator")

$$V_{ijkl} = \iint \phi_i(\vec{r})\phi_j(\vec{r}\,')V(\vec{r},\vec{r}\,')\phi_k(\vec{r})\phi_l(\vec{r}\,')d^3r d^3r'$$

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

"occupation representation"

$$|\alpha\rangle = \hat{a}^{+}_{n_1}\hat{a}^{+}_{n_2}\hat{a}^{+}_{n_3}\ldots\hat{a}^{+}_{n_N}|0\rangle$$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

some technical details:
the "M-scheme"

$$|\alpha\rangle = \hat{a}^+_{n_1} \hat{a}^+_{n_2} \hat{a}^+_{n_3} \ldots \hat{a}^+_{n_N} |0\rangle$$

label	N	l	m_l
1	1	0 (S)	0
2	2	0 (S)	0
3	2	1 (P)	1
4	2	1 (P)	0
5	2	1 (P)	-1

For any Slater determinant, the total M = sum of the m_l's

Because J_z commutes with H, we can use a basis with M fixed = "M-scheme"

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

some technical details:
the "M-scheme"

$$|\alpha\rangle = \hat{a}^+_{n_1} \hat{a}^+_{n_2} \hat{a}^+_{n_3} \ldots \hat{a}^+_{n_N} |0\rangle$$

label	N	l	m_l
1	1	0 (S)	0
2	2	0 (S)	0
3	2	1 (P)	1
4	2	1 (P)	0
5	2	1 (P)	-1

So for 2-particle system, positive parity, M = 0:
1 and 2 $(1s_{m=0})(2s_{m=0})$
3 and 5 $(2p_{m=1})(2p_{m=-1})$

negative parity, M = 0
1 and 4 $(1s_{m=0})(2p_{m=0})$
2 and 4 $(2s_{m=0})(2p_{m=0})$

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

some technical details:
the “M-scheme”

$$|\alpha\rangle = \hat{a}^{+}_{n_1}\hat{a}^{+}_{n_2}\hat{a}^{+}_{n_3}\ldots\hat{a}^{+}_{n_N}|0\rangle$$

label	N	l	m_l
1	1	0 (S)	0
2	2	0 (S)	0
3	2	1 (P)	1
4	2	1 (P)	0
5	2	1 (P)	-1

If I have two species (spin up/down) then *combined* M must be fixed:

e.g. 4 electrons, M = 0, + parity
spin up: states 1 + 2 $(1S_{m=0})(2S_{m=0})$
spin down: 3 +5 $(2P_{m=1})(2P_{m=-1})$
or
spin up: states 1 + 3 $(1S_{m=0})(2P_{m=1})$
spin down: states 2, 5 $(2S_{m=0})(2P_{m=-1})$

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

"occupation representation" $\left|\alpha\right\rangle = \hat{a}^+_{n_1}\hat{a}^+_{n_2}\hat{a}^+_{n_3}\ldots\hat{a}^+_{n_N}\left|0\right\rangle$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

$$\hat{H} = \sum_{ij} T_{ij}\hat{a}^+_i\hat{a}_j + \frac{1}{4}\sum_{ijkl} V_{ijkl}\hat{a}^+_i\hat{a}^+_j\hat{a}_l\hat{a}_k$$

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

"occupation representation"

$$|\alpha\rangle = \hat{a}^+_{n_1}\hat{a}^+_{n_2}\hat{a}^+_{n_3}\ldots\hat{a}^+_{n_N}|0\rangle$$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

$$\hat{a}^+_3\hat{a}^+_6\hat{a}_4\hat{a}_5|\alpha=1\rangle = |\alpha=2\rangle$$

A SPARSE MATRIX, BUT....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

"occupation representation" $|\alpha\rangle = \hat{a}^+_{n_1}\hat{a}^+_{n_2}\hat{a}^+_{n_3}\ldots\hat{a}^+_{n_N}|0\rangle$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

$$\hat{a}^+_2\hat{a}^+_4\hat{a}_1\hat{a}_7|\alpha = 2\rangle = |\alpha = 3\rangle$$

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- Why most matrix elements are zero

"occupation representation"

$$|\alpha\rangle = \hat{a}^{+}_{n_1}\hat{a}^{+}_{n_2}\hat{a}^{+}_{n_3}\ldots\hat{a}^{+}_{n_N}|0\rangle$$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

$$\hat{a}^{+}_2\hat{a}^{+}_4\hat{a}^{+}_6\hat{a}_1\hat{a}_5\hat{a}_7|\alpha = 1\rangle = |\alpha = 3\rangle$$

need 3 particles to interact simultaneously!

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- Typical dimensions and sparsity

Nuclide	valence space	valence Z	valence N	basis dim	sparsity (%)
^{20}Ne	“sd”	2	2	640	10
^{25}Mg	“sd”	4	5	44,133	0.5
^{49}Cr	“pf”	4	5	6M	0.01
^{56}Fe	“pf”	6	10	500M	$2\text{x}10^{-4}$

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- Typical dimensions and sparsity

Nuclide	valence space	valence Z	valence N	basis dim	sparsity (%)	
^{20}Ne	"sd"	2	2	640	10	
^{25}Mg	"sd"	4	5	44,133	0.5	
^{49}Cr	"pf"	4	5	6M	0.01	
^{56}Fe	"pf"	6	10	500M	$2x10^{-4}$	
^{12}C	N_{max}=8	6	6	600M	$4x10^{-4}$	2-body force
^{12}C	N_{max}=8	6	6	600M	$2x10^{-2}$	3-body force

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

Nuclide	Space	Basis dim	matrix store
^{56}Fe	*pf*	501 M	4.2 Tb
^{7}Li	$N_{max}=12$	252 M	3.6 Tb
^{7}Li	$N_{max}=14$	1200 M	23 Tb
^{12}C	$N_{max}=6$	32M	0.2 Tb
^{12}C	$N_{max}=8$	590M	5 Tb
^{12}C	$N_{max}=10$	7800M	111 Tb
^{16}O	$N_{max}=6$	26 M	0.14 Tb
^{16}O	$N_{max}=8$	990 M	9.7 Tb

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

Nuclide	Space	Basis dim	matrix store (2-body)	matrix store (3-body)
^{4}He	N_{max}=16	6 M	0.2 Gb	12 Tb
^{4}He	N_{max}=20	39 M	3 Tb	270 Tb
^{7}Li	N_{max}=10	43 M	0.4 Tb	176 Tb
^{12}C	N_{max}=6	32M	0.2 Tb	6.2 Tb
^{12}C	N_{max}=8	590M	5 Tb	200 Tb

Recycled Matrix Elements

Only a fraction of matrix elements are unique; **most are reused.**
Reuse of matrix elements understood through *spectator* particles.

n_i	1	2	3	4	5	6	7	8
α=1	1	1	1	0	0	0	0	1
α=2	1	1	0	1	1	0	0	0
α=3	0	1	1	0	0	1	0	1
α=4	0	1	0	1	1	1	0	0
α=5	0	0	1	0	0	1	1	1
α=6	0	0	0	1	1	1	1	0

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha=1\rangle = |\alpha=2\rangle$$

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha=3\rangle = |\alpha=4\rangle$$

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha=5\rangle = |\alpha=6\rangle$$

All of these have the same matrix element: V_{4538}

Recycled Matrix Elements

Only a fraction of matrix elements are unique; **most are reused.**
Reuse of matrix elements understood through *spectator* particles.

# of nonzero matrix elements vs. # unique matrix elements

Nuclide	valence space	valence Z	valence N	# nonzero	# unique
^{28}Si	"sd"	6	6	26×10^6	3600
^{52}Fe	"pf"	6	6	90×10^9	21,500

Nuclide	ab initio space	basis dim	# nonzero m.e.s	# unique	avg redundancy
^{4}He	N_{max}=16	6M	2×10^{10}	10^9	18
^{12}C	N_{max}=8	600M	6×10^{11}	5×10^7	10,000

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

We work in an M-scheme basis:

Because $\mathbf{J}^2$ and $\mathbf{J}_z$ both commute with $\mathbf{H}$, one does not need *all* basis states, but can use many-body basis restricted to the same M.

This is easy because M is an additive quantum number so it is possible for a single Slater determinant to be a state of good M.

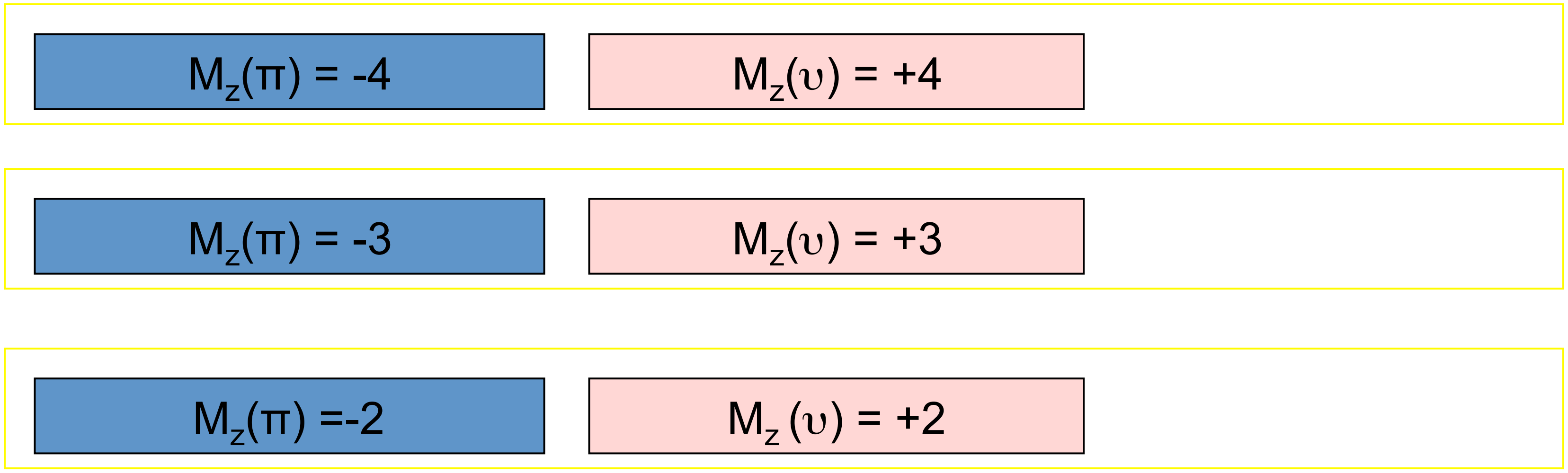
(It's possible to work in a J-basis, e.g. OXBASH or NuShell, but each basis state is generally a complicated sum of Slater determinants).

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Because the M values are discrete integers or half-integers (-3, -2, -1, 0, 1, 2, … or -3/2, -1/2, +1/2, +3/2….) we can organize the basis states in discrete *sectors*

Example: 2 protons, 4 neutrons, total M = 0



FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

In fact, we can see an example of factorization here because all proton Slater determinants in one M-sector *must* combine with all the conjugate neutron Slater determinants

Example: 2 protons, 4 neutrons, total M = 0

$M_z(\pi)$ = -4: 2 SDs	$M_z(\upsilon)$ = +4: 24 SDs	48 combined
$M_z(\pi)$ = -3: 4 SDs	$M_z(\upsilon)$ = +3: 39 SDs	156 combined
$M_z(\pi)$ = -2: 9 SDs	$M_z(\upsilon)$ = +2: 60 SDs	540 combined

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

In fact, we can see an example of factorization here because all proton Slater determinants in one M-sector *must* combine with all the conjugate neutron Slater determinants

$M_z(\pi)$ = -4: 2 SDs

$M_z(\upsilon)$ = +4: 24 SDs

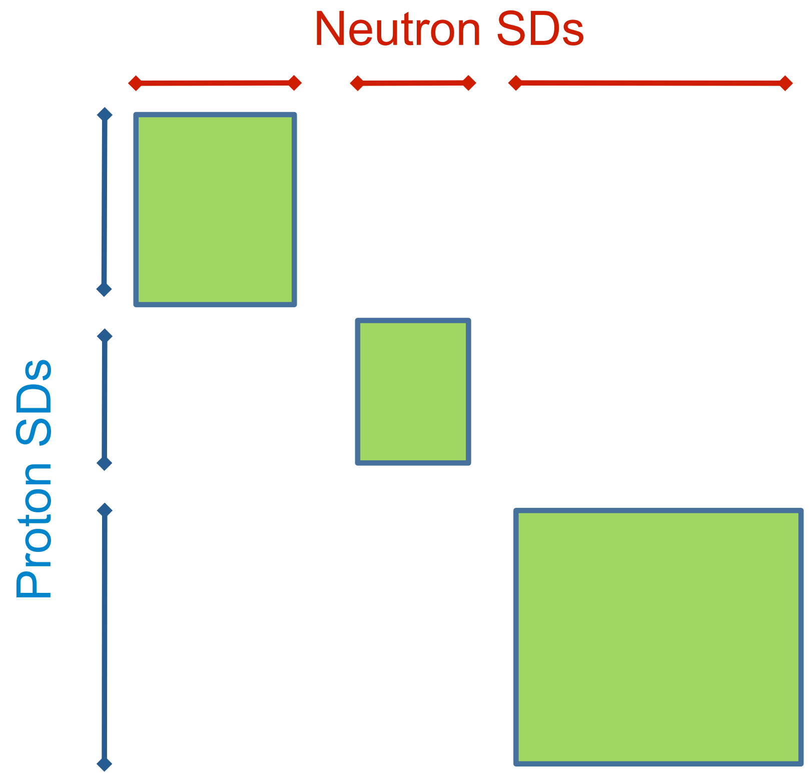
48 combined

$$\begin{matrix} |\pi_1\rangle \\ |\pi_2\rangle \end{matrix} \times \begin{matrix} |\nu_1\rangle \\ |\nu_2\rangle \\ |\nu_3\rangle \\ |\nu_4\rangle \\ \vdots \\ |\nu_{24}\rangle \end{matrix} = \begin{matrix} |\pi_1\rangle|\nu_1\rangle \\ |\pi_2\rangle|\nu_1\rangle \\ |\pi_1\rangle|\nu_2\rangle \\ |\pi_2\rangle|\nu_2\rangle \\ \vdots \\ |\pi_1\rangle|\nu_{24}\rangle \\ |\pi_2\rangle|\nu_{24}\rangle \end{matrix}$$

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$$|\alpha\rangle = |\alpha_p\rangle \times |\alpha_n\rangle$$



Example N = Z nuclei

Nuclide	Basis dim	# pSDs (=#nSDs)
^{20}Ne	640	66
^{24}Mg	28,503	495
^{28}Si	93,710	924
^{48}Cr	1,963,461	4895
^{52}Fe	109,954,620	38,760
^{56}Ni	1,087,455,228	125,970

FACTORIZATION

Reuse can be **exploited using exact factorization**
enforced through *additive/multiplicative quantum numbers*

Factorization allows us to keep track of all basis states
without writing out every one explicitly
-- we only need to write down the proton/neutron components

The same trick can be applied to matrix-vector multiply

$$\hat{H} = \hat{H}_{pp} + \hat{H}_{nn} + \hat{H}_{pn}$$

Move 2 protons;
neutrons are spectators

Move 2 neutrons;
protons are spectators

Move 1 proton + 1 neutron;
rest are spectators

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$$\hat{H}_{pp}$$

Move 2 protons; *neutrons are spectators*

Example: 2 protons, 4 neutrons, total M = 0

$M_z(\pi)$ = -4: 2 SDs

$M_z(\upsilon)$ = +4: 24 SDs

48 combined

There are potentially 48 × 48 matrix elements
But for H_{pp} at most 4 × 24 are nonzero
and we only have to look up 4 matrix elements

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$M_z(\pi) = -4$: 2 SDs

$M_z(\upsilon) = +4$: 24 SDs

48 combined

$$|\pi_1\rangle \quad |\pi_2\rangle$$

$$H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$$

$$|\nu_1\rangle \quad |\nu_2\rangle \quad |\nu_3\rangle \quad |\nu_4\rangle \quad \vdots \quad |\nu_{24}\rangle$$

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$M_z(\pi) = -4$: 2 SDs

$M_z(\upsilon) = +4$: 24 SDs

48 combined

$|\pi_1\rangle$

$|\pi_2\rangle$

$$H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$$

$|\nu_1\rangle$, $|\nu_2\rangle$, $|\nu_3\rangle$, $|\nu_4\rangle$, $\vdots$, $|\nu_{24}\rangle$

$$H_{pp}|\pi_1\rangle|\nu_1\rangle = H_{11}|\pi_1\rangle|\nu_1\rangle + H_{12}|\pi_2\rangle|\nu_1\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_1\rangle = H_{12}|\pi_1\rangle|\nu_1\rangle + H_{22}|\pi_2\rangle|\nu_1\rangle$$
$$H_{pp}|\pi_1\rangle|\nu_2\rangle = H_{11}|\pi_1\rangle|\nu_2\rangle + H_{12}|\pi_2\rangle|\nu_2\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_2\rangle = H_{12}|\pi_1\rangle|\nu_2\rangle + H_{22}|\pi_2\rangle|\nu_2\rangle$$
$$\vdots$$
$$H_{pp}|\pi_1\rangle|\nu_{24}\rangle = H_{11}|\pi_1\rangle|\nu_{24}\rangle + H_{12}|\pi_2\rangle|\nu_{24}\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_{24}\rangle = H_{12}|\pi_1\rangle|\nu_{24}\rangle + H_{22}|\pi_2\rangle|\nu_{24}\rangle$$

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Comparison of nonzero matrix storage with factorization

Nuclide	Space	Basis dim	matrix store	factorization
^{56}Fe	*pf*	501 M	290 Gb	0.72 Gb
^{7}Li	N_{max}=12	252 M	3600 Gb	96 Gb
^{7}Li	N_{max}=14	1200 M	23 Tb	624 Gb
^{12}C	N_{max}=6	32M	196 Gb	3.3 Gb
^{12}C	N_{max}=8	590M	5000 Gb	65 Gb
^{12}C	N_{max}=10	7800M	111 Tb	1.4 Tb
^{16}O	N_{max}=6	26 M	142 Gb	3.0 Gb
^{16}O	N_{max}=8	990 M	9700 Gb	130 Gb

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Comparison of nonzero matrix storage with factorization

Nuclide	Space	Basis dim	matrix store	factorization
^{7}Li	N_{max}=12	252 M	3600 Gb	96 Gb
^{7}Li	N_{max}=14	1200 M	23 Tb	624 Gb
^{12}C	N_{max}=6	32M	196 Gb	3.3 Gb
^{12}C	N_{max}=8	590M	5000 Gb	65 Gb
^{12}C	N_{max}=10	7800M	111 Tb	1.4 Tb
^{16}O	N_{max}=6	26 M	142 Gb	3.0 Gb
^{16}O	N_{max}=8	990 M	9700 Gb	130 Gb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{4}He

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=14	2M	46 Gb	1.2 Gb	2 Tb	16 Gb
N_{max}=16	6M	200 Gb	4 Gb	12 Tb	60 Gb
N_{max}=18	16M	820 Gb	11 Gb	60 Tb	190 Gb
N_{max}=20	39M	3 Tb	29 Gb	270 Tb	600 Gb
N_{max}=22	86M	9 Tb	70 Gb	1.1 Pb	1.4 Tb

Factorization

Comparison of nonzero matrix storage with factorization

^{4}He

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=8	29 M	1.4 Tb	0.6 Gb	120 Tb	11 Gb
N_{shell}=9	93 M	8 Tb	1.7 Gb	870 Tb	40 Gb
N_{shell}=10	270 M	36 Tb	5 Gb	5 Pb	120 Gb
N_{shell}=11	700 M	150 Tb	12 Gb	28 Pb	350 Gb
N_{shell}=12	1.7 G	500 Tb	27 Gb	130 Pb	900 Gb
N_{shell}=13	4 G	1.7 Pb	60 Gb	500 Pb	2 Tb

Factorization

Comparison of nonzero matrix storage with factorization

^{7}Li

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{max}=8$	6 M	36 Gb	1.5 Gb	1 Tb	26 Gb
$N_{max}=10$	43 M	430 Gb	10 Gb	170 Tb	250 Gb
$N_{max}=12$	250 M	4 Tb	60 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{shell}=3$	0.4 M	0.8 Gb	6 Mb	10 Gb	44 Mb
$N_{shell}=4$	45 M	330 Gb	0.3 Gb	9 Tb	4 Gb
$N_{shell}=5$	2 G	38 Tb	16 Gb	2 Pb	140 Gb
$N_{shell}=6$	50 G	2 Pb	87 Gb	170 Pb	3 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{9}Be

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=6	5 M	22 Gb	1 Gb	0.6 Tb	12 Gb
N_{max}=8	63 M	460 Gb	9 Gb	17 Tb	200 Gb
N_{max}=10	570 M	7 Tb	70 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=3	4 M	15 Gb	30 Mb	240 Gb	240 Mb
N_{shell}=4	3 G	30 Tb	3 Gb	1 Pb	50 Gb
N_{shell}=5	400 G	12 Pb	130 Gb	800 Pb	3.6 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{10}B

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{max}=6$	12 M	60 Gb	1.3 Gb	1.6 Tb	22 Gb
$N_{max}=8$	165 M	1.3 Tb	16 Gb	52 Tb	360 Gb

Factorization

Comparison of nonzero matrix storage with factorization

^{12}C

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{max}=6$	32 M	170 Gb	3 Gb	5 Tb	60 Gb
$N_{max}=8$	590 M	5 Tb	45 Gb	200 Tb	1 Tb
$N_{max}=10$	8 G	100 Tb	440 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{shell}=3$	82 M	400 Gb	0.1 Gb	9 Tb	1.5 Gb
$N_{shell}=4$	600 G	10 Pb	43 Gb	580 Tb	0.9 Tb

Factorization

Comparison of nonzero matrix storage with factorization

^{16}O

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=4	0.3 M	1 Gb	70 Mb	17 Gb	0.7 Gb
N_{max}=6	26 M	140 Gb	3 Gb	4 Tb	53 Gb
N_{max}=8	1 G	8.6 Tb	70 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=3	800 M	6 Tb	0.7 Gb	140 Tb	7.5 Gb

Drawbacks of factorization/on-the-fly algorithms:

Much more complicated to code up (even matrix storage is not trivial)

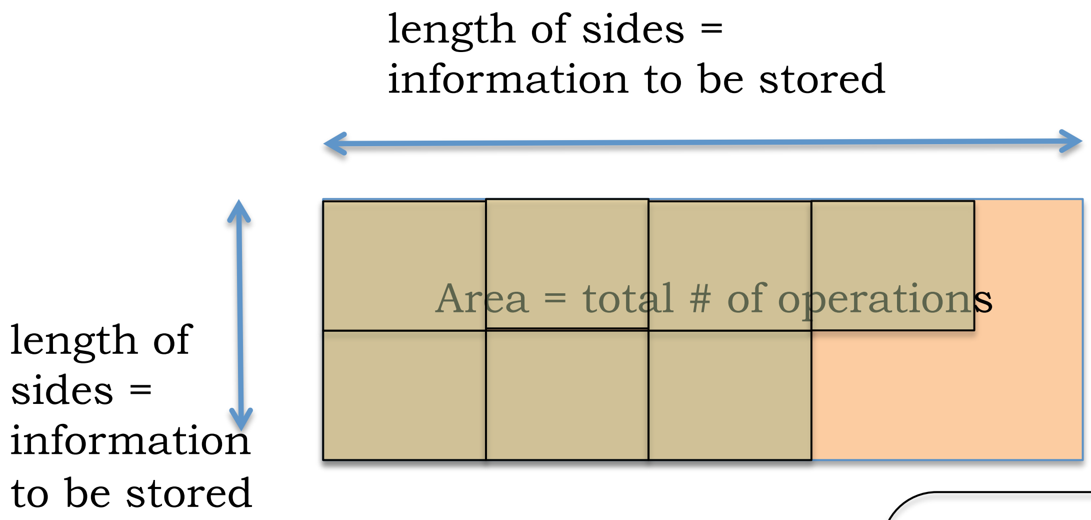
Less flexible in basis—for example, importance truncation much harder (if even possible)

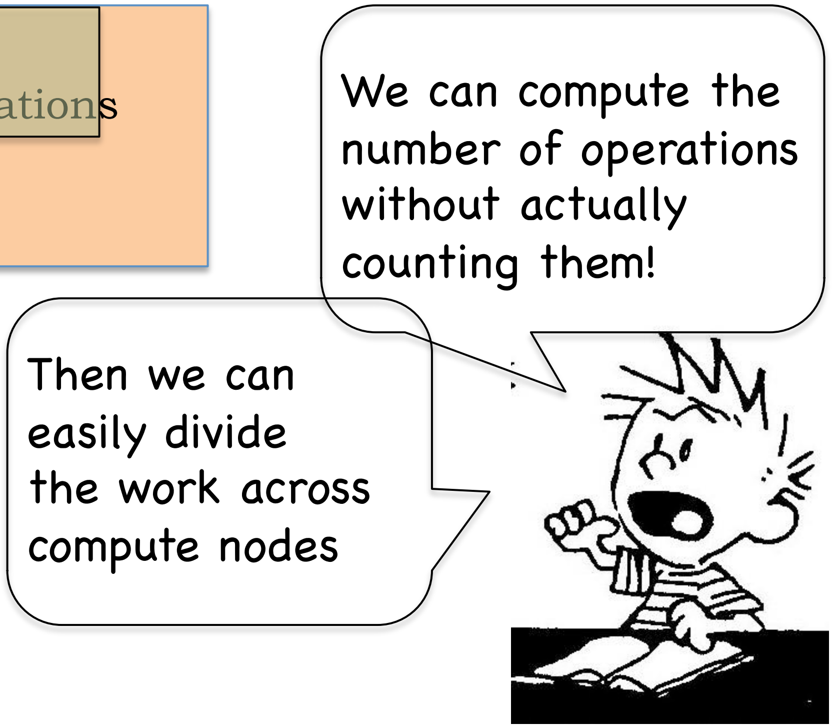
4-body is in principle straightforward

Experience in going from 2-body to 3-body shows most difficult part is correctly matching indices of input interaction to internal representation (+ induced phases etc) – useful to have *small* cases with known solutions for debugging

Parallel Implementation

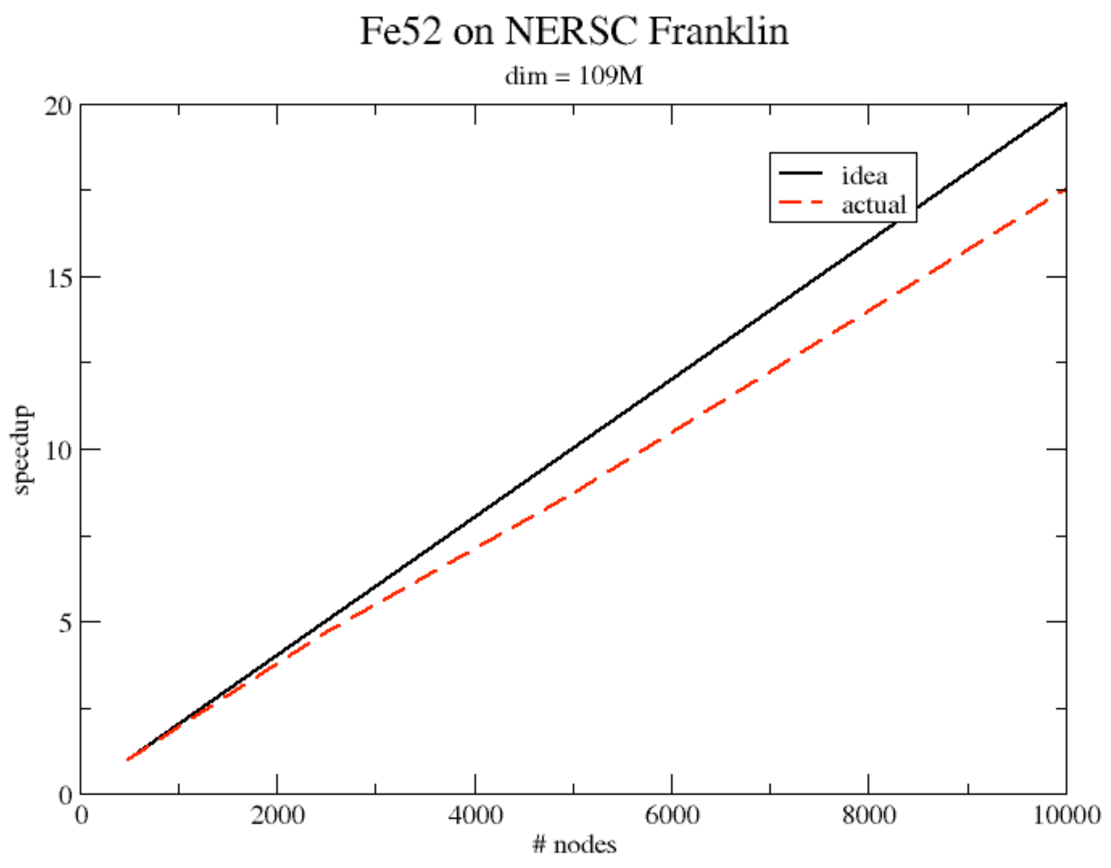
Factorization makes it easier to compute workload and distribute across multiple nodes


length of sides = information to be stored
length of sides = information to be stored
Area = total # of operations


We can compute the number of operations without actually counting them!
Then we can easily divide the work across compute nodes

Parallel Implementation

Factorization makes it easier to compute workload and distribute across multiple nodes



THE BIGSTICK CODE

Many-fermion code: 2nd generation after REDSTICK code
(started in *Baton Rouge, La.*)

Arbitrary single-particle radial waveforms
Allows local or nonlocal two-body interaction
Applies to both nuclear and atomic cases

Runs on both desktop and parallel machines
--can run at least dimension 100M+ on desktop
(20 Lanczos iterations in 300 CPU minutes)

20-30k lines of codes
Fortran 90 + MPI + OpenMP
Partially funded by SciDAC
Plans to run on 50,000-100,000 compute nodes
Plans to publish code

The BIGSTICK code

What about other codes?

MFDn (Vary et al): M-scheme code, includes 3-body, mostly stores H in memory; optimized for ab initio and is leading CI code today for ab initio.
Less flexible in model space, awkward to run on desktop machines.

NuShell(X) (Rae, Brown,Horoi et al): J-scheme code. Uses factorization. Flexible single-particle space. Leading CI code for phenomenological shell model.
No 3-body yet, awkward for Nmax truncations

ANTOINE (Caurier et al): M-scheme code. Uses factorization. Flexible single-particle space. Includes 3-body. Was leading CI code.
Not (fully) parallelized

MCSM (Otsuka et al): Samples J-scheme basis stochastically. Includes (?) 3-body. Significant effort has gone into numerics
Cannot truncate many-body space

CONCLUSIONS

Factorization allows one to represent and store a many-body Hamiltonian efficiently and compactly.

The trade-off is a more complex algorithm with many subtleties in parallelization.

BIGSTICK attempts to approach “best of both codes”, the *ab initio* capability (in truncation and 3-body) of MFDn and the efficiency and flexibility of NuShellX.

LOOKING TO PARTNER WITH FEW-BODY / INTERACTION SPECIALISTS