Factorization Algorithms for Large-Scale Many-Body Problems

Calvin Johnson, Department of Physics *and*
Computational Science Research Center
San Diego State University
cjohnson @ mail . sdsu . edu

Collaborators:
W. Erich Ormand, Lawrence Livermore
Plamen G. Krastev, SDSU/Harvard
Ken McElvain, UC Berkeley/LBNL

Supported by U.S. Department of Energy

C. W. Johnson, W. E. Ormand, and P. G. Krastev,
Comp. Phys. Comm. **184**, 2761-2774 (2013)

THE BASIC PROBLEM

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

To answer this, we attempt 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

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

The Basic Problem

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

This differential equation is too difficult to solve directly

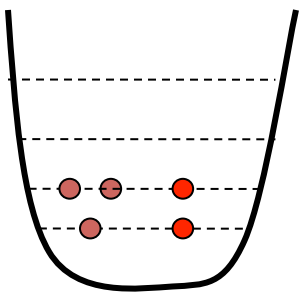
$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \sum_{i<j} V(\vec{r}_i - \vec{r}_j)\right)\Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3 \ldots) = E\Psi$$

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.
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:

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

single-particle energies

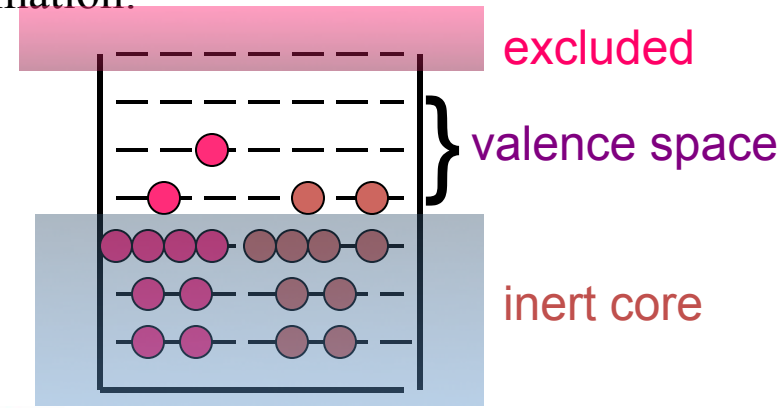
two-body matrix elements

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: 63

Single Particle Energies: 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
1	2	1		2	0	-3.8253000
				2	1	-0.3[illegible]
				3	0	[illegible]

Single Particle States

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

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

Summary:

The Schrödinger eqn has become a matrix eigenvalue equation

$$\sum_B H_{AB} \mathrm{v}_\mathrm{B} = E_A \mathrm{v}_\mathrm{A}$$

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 (and sometimes three—body) matrix elements* and solves for eigenvalues/vectors.

A Problem….

Despite sparsity, nonzero matrix elements can require TB of storage

Nuclide	Space	Basis dim	matrix store
^{56}Fe	*pf*	501 M	3.5 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

Spread nonzero matrix elements over many MPI compute nodes: For example, the Iowa MFDn code

THE KEY IDEAS

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

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 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)

The Basic Problem

Find extremal eigenvalues of very large, very sparse Hermitian matrix

→ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

we use the matrix formalism

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

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

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

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

→ 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 (in M-scheme).
* We usually only want a few low-lying states

→ Lanczos algorithm!
(or variant, e.g. LOBPCG)

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

→ 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

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

Lanczos algorithm

fundamental operation is *matrix-vector multiply*

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

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

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

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

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

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

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

Product wavefunction ("Slater Determinant")

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

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

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

A SPARSE MATRIX, BUT....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

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

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"

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

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

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

A SPARSE MATRIX, BUT....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

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

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

$$\hat{a}^{+}_{3}\hat{a}^{+}_{6}\hat{a}_{4}\hat{a}_{5}\left|\alpha=1\right\rangle = \left|\alpha=2\right\rangle$$

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- How the Hamiltonian is represented

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

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

$$\hat{a}^{+}_{2}\hat{a}^{+}_{4}\hat{a}_{1}\hat{a}_{7}\left|\alpha=2\right\rangle = \left|\alpha=3\right\rangle$$

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

- Why most matrix elements are zero

"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

$$\hat{a}^{+}_{2}\hat{a}^{+}_{4}\hat{a}^{+}_{6}\hat{a}_{1}\hat{a}_{5}\hat{a}_{7}\left|\alpha=1\right\rangle = \left|\alpha=3\right\rangle$$

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	2×10^{-4}

This corresponds to 2 Tb of data!

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 x 10^6	3600
^{52}Fe	"pf"	6	6	90 x 10^9	21,500

Atom	space	# nonzero	# unique
Be	CVB3	110x10^6	521,000
B	CVB2	1.4x10^9	379,000
C	CVB1	260x10^6	40,751

FACTORIZATION

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

A quantum number is the eigenvalue of an operator

For composite systems, one can apply the operator to each component separately:

$$\hat{O}|\Psi\rangle = \left(\hat{O}_1 + \hat{O}_2 + \hat{O}_3 + \ldots\right)\left(|\Psi_1\rangle \otimes |\Psi_2\rangle \otimes |\Psi_3\rangle \otimes \ldots\right)$$

Sometimes the total quantum number is a simple sum/product as is the case for $\mathbf{J}_z$ or parity....

$$\hat{J}_z|\Psi\rangle = M|\Psi\rangle = (m_1 + m_2 + m_3 + \ldots)|\Psi\rangle$$

...but in other cases the addition is complicated (e.g. for $\mathbf{J}^2$)

FACTORIZATION

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

I consider composite many-fermion systems,
in particular those with 2 major components
protons and *neutrons*
or
spin-up and *spin-down* electrons

$$\left|\Psi\right\rangle = \left|\Psi_1\right\rangle \otimes \left|\Psi_2\right\rangle$$

Each component itself is a Slater determinant which is composed of many particles

$$\hat{J}_z\left|\Psi\right\rangle = M\left|\Psi\right\rangle \qquad M = M_1 + M_2$$

$$M_1 = m_1^{(1)} + m_1^{(2)} + m_1^{(2)} + \ldots$$

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

$M_z(\pi) = -4$	$M_z(\upsilon) = +4$
$M_z(\pi) = -3$	$M_z(\upsilon) = +3$
$M_z(\pi) = -2$	$M_z(\upsilon) = +2$

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

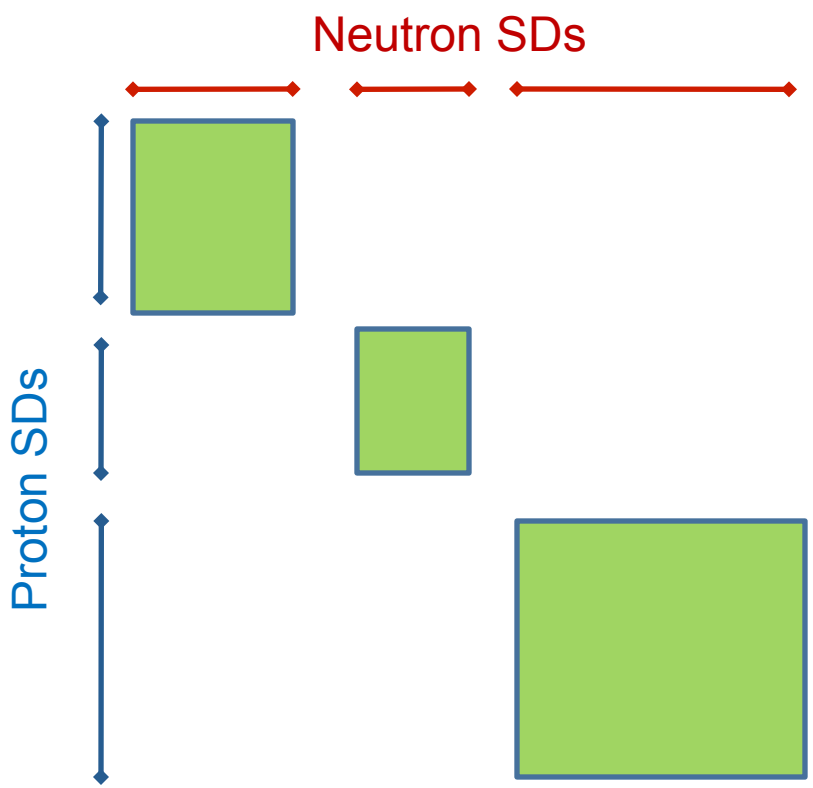
$M_z(\pi) = -4$: 2 SDs	$M_z(\upsilon) = +4$: 24 SDs	48 combined

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

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

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

$$|\pi_1\rangle \qquad |\pi_2\rangle \qquad H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$$

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

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

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*

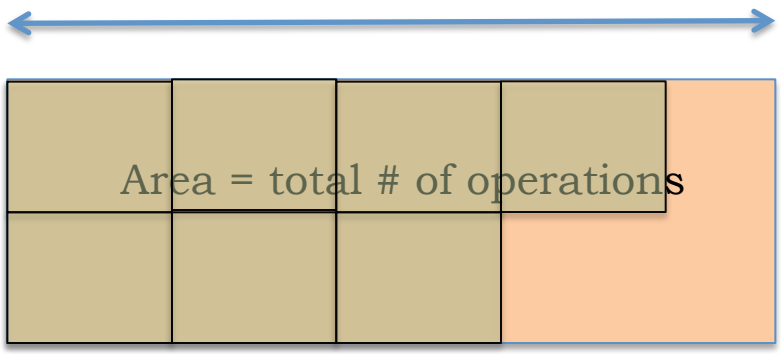
Comparison of nonzero matrix storage with factorization

Nuclide	Space	Basis dim	matrix store	factorization
^{56}Fe	*pf*	501 M	3500 Gb	0.72 Gb
^{7}Li	N_{max}=12	252 M	3800 Gb	61 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

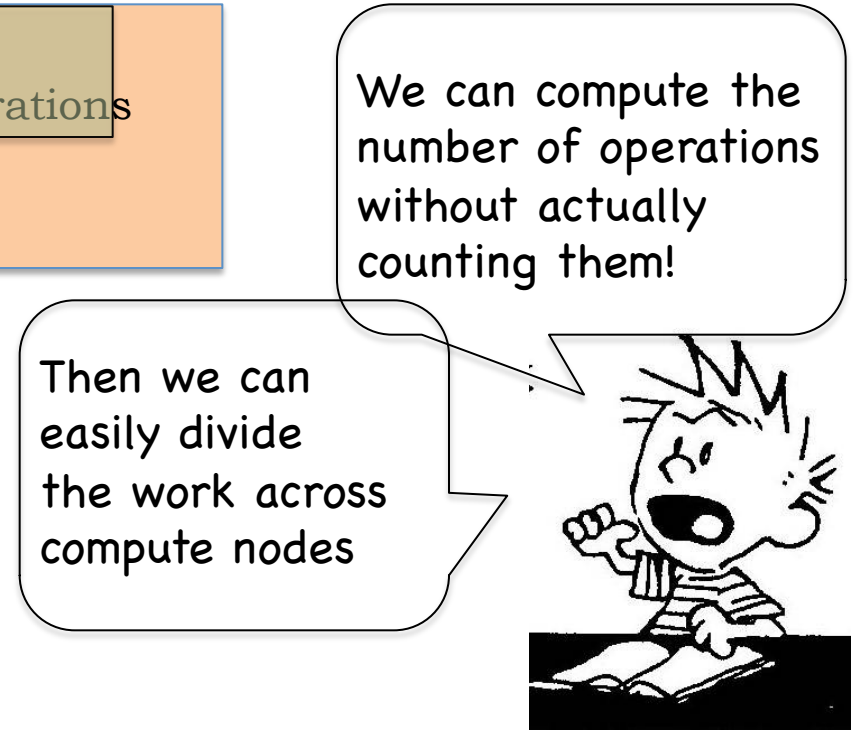
Parallel Implementation

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

length of sides = information to be stored

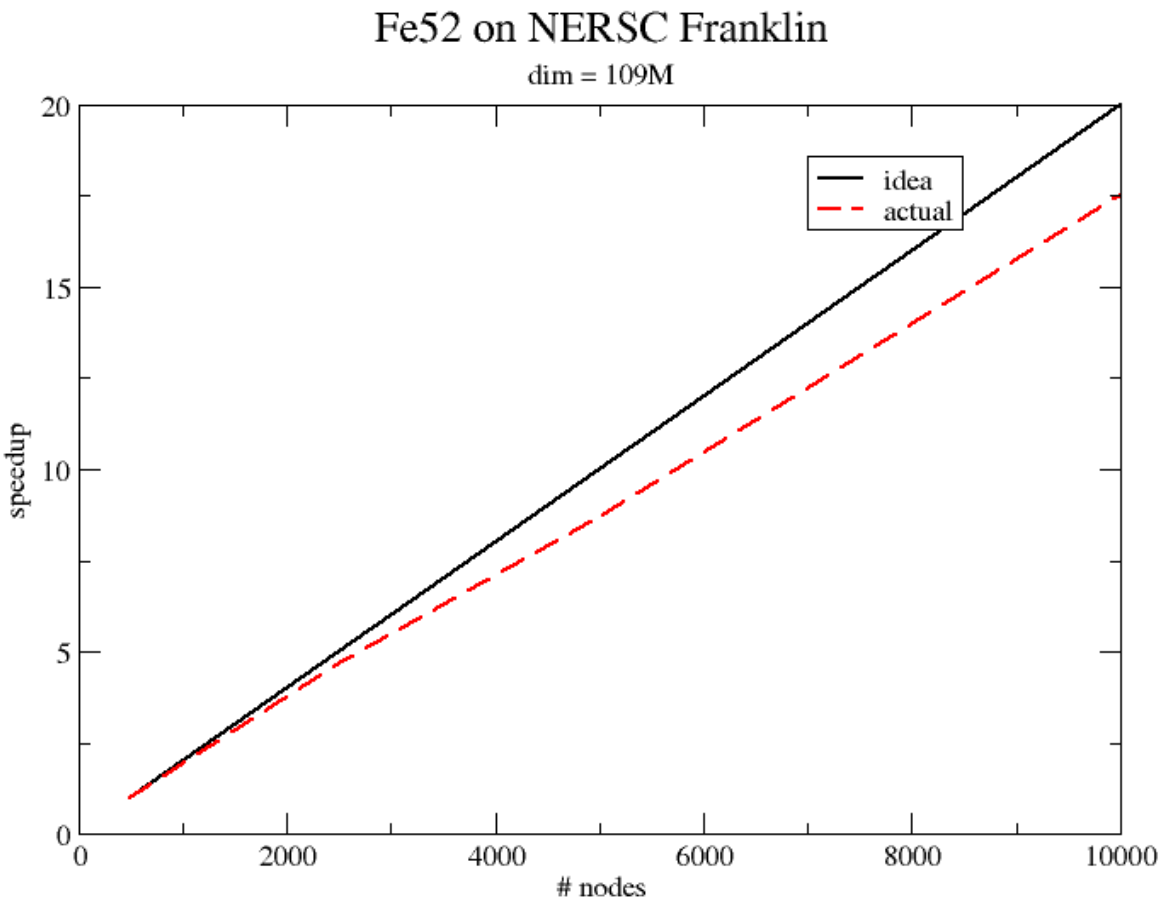


length of sides = information to be stored



Parallel Implementation

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



Some Shell-Model Codes

Matrix storage:

Oak Ridge-Rochester (small matrices)

Glasgow-Los Alamos (M-scheme, stored on disk; introduced Lanczos)

OXBASH /Oxford-MSU (J-scheme, stored on disk)

CMSM / Central Michigan (M-scheme, stored on disk farm)

MSHELL64 (?) / Mizusaki *et al*

MFDn/ Iowa State (M-scheme, stored in RAM; plans for J-scheme, SU(3)-scheme w/LSU)

Factorization:

ANTOINE Strasbourg (M-scheme; originator of factorization)

NATHAN Strasbourg (J-scheme)

EICODE (J-scheme)

NuShell/NuShellX (J-scheme)

KSHELL CNS U Tokyo (M-scheme Shimuzu; cf. arviv:1310.5431)

REDSTICK+BIGSTICK/ LSU-SDSU-Livermore

Some Shell-Model Codes

Other approaches:

"Shell-model Monte Carlo" (Koonin, Johnson, Ormand, Dean, Alhassid, Langanke..) Rewrite as path integral, evaluate by Monte Carlo

"Monte Carlo Shell-Model" (Otsuka *et al*) Sample basis from Monte Carlo random walk over path integral

Importance truncation (R. Roth *et al*, **a** Darmstadt group): Truncate basis using perturbation theory

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 2015

SCIENCE APPLICATIONS:

- Benchmarking approximations:
  - -- J-projected Hartree-Fock (J. Staker)
  - -- Quartet approximation (N. Sandulescu)
  - -- Spin-cutoff parameterization for level densities (W. Spinella)
- Transition strength functions, e.g. Gamow-Teller
  - -- ½ million transitions in *sd* shell for astro library (M. Bertolli)
  - -- Transitions from excited state/Axel-Brink (M. Schuster)
  - -- Isospin violation corrections to superallowed (Kruse & Ormand)
  - -- SRG-evolved operators (M. Schuster and S. Quaglioni)
- Spin-Orbit (LS) analysis of wavefunctions

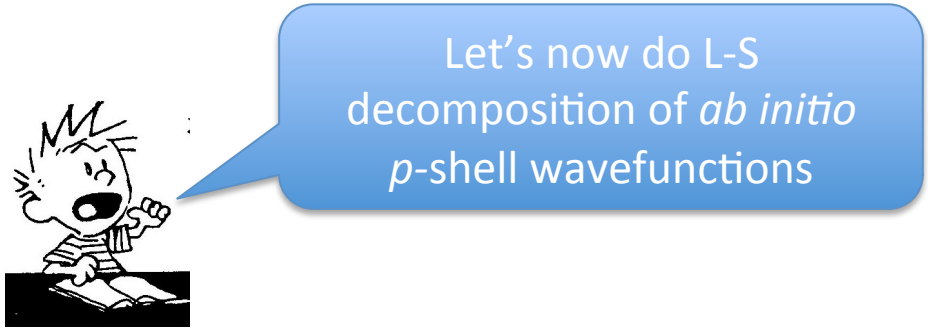
SCIENCE APPLICATION: L-S decomposition of nuclear wavefunctions

Spin-orbit dissection of nuclear wavefunctions

--look at fraction of total L or total S

Nuclide	Model space	Interaction	g.s. =	g.s. =
^{48}Ca	pf	KB3G	90 % $(0f_{7/2})^8$	20% L = 0
^{24}O	sd	USDB	91% $(0d_{5/2})^6 (1s_{½})^2$	34% L = 0
^{22}O	sd	USDB	75% $(0d_{5/2})^6$	38% L = 0
^{8}He	p	Cohen-Kurath	53 % $(0p_{3/2})^4$	96% L = 0
^{32}S	sd	USDB	29 % $(0d_{5/2})^{12} (1s_{½})^4$	34% L = 0
^{28}Si	sd	USDB	21% $(0d_{5/2})^{12}$	36% L = 0
^{12}C	p	Cohen-Kurath	37% $(0p_{3/2})^8$	82% L = 0

This illustrates a (once) well-known fact: that L-S coupling is a better approximation in the *p*-shell than *j-j* coupling.



Why?

-- To see if this pattern holds for *ab initio* interactions

-- How well do phenomenological interactions match *ab initio*?

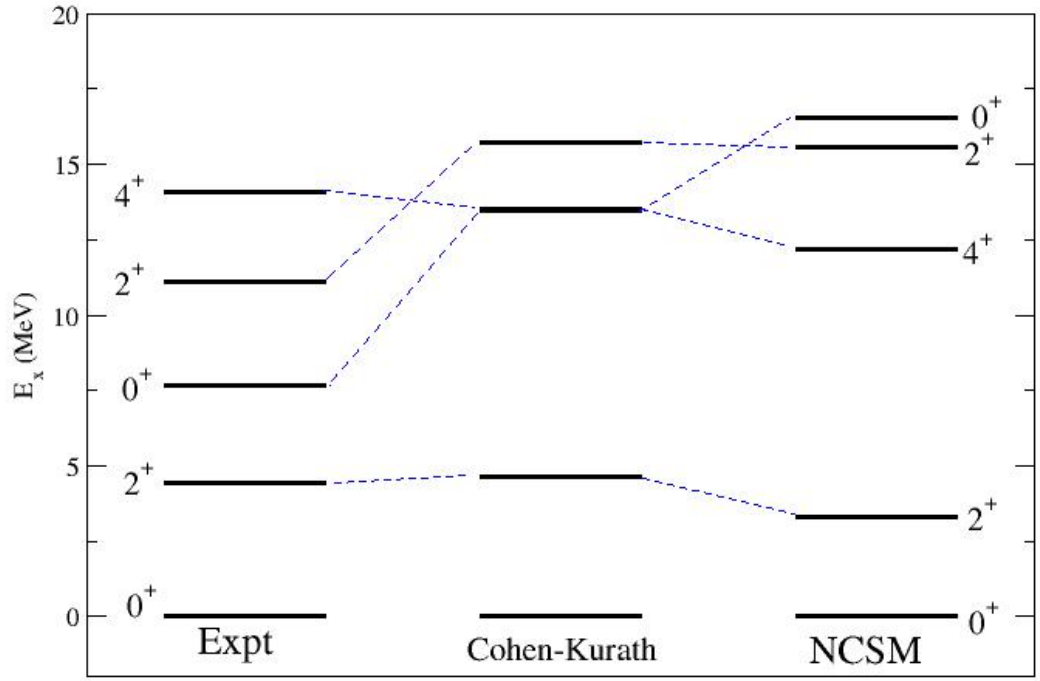
-- Crucially, we know the 3-body forces strongly affects the spin-orbit force. Can we see this happen directly?

Note: In this talk I only give 2-body results. Need 3-body forces…

^{12}C

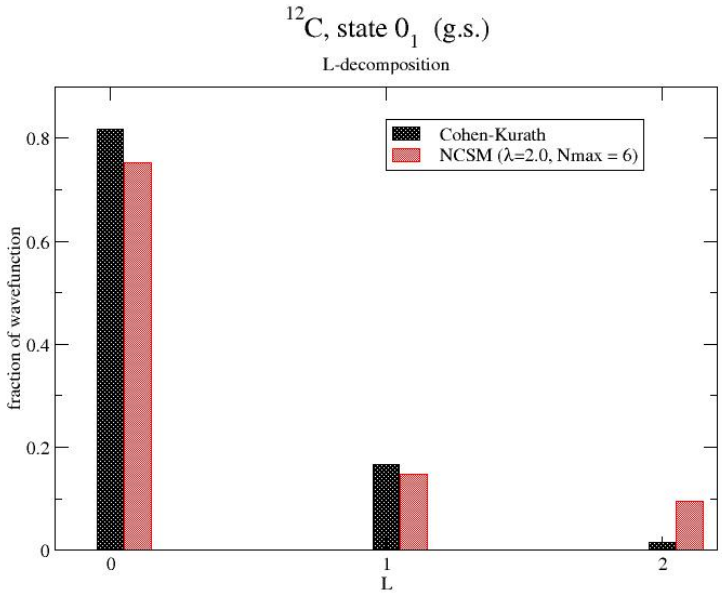
Phenomenological Cohen-Kurath force (1965) in 0*p* shell
m-scheme dimension: 51

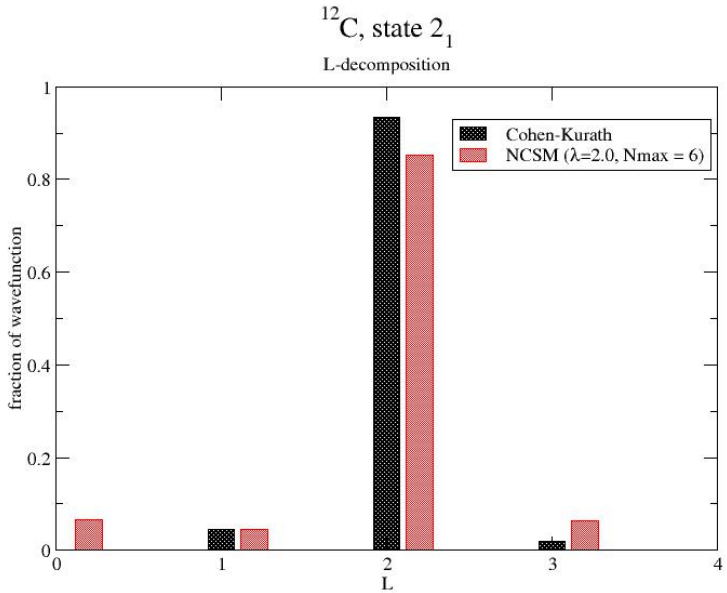
NCSM: N3LO chiral 2-body force SRG evolved* to λ = 2.0 fm^{-1}, N_{max} = 6, $\hbar\omega$=22 MeV
m-scheme dimension: 35 million

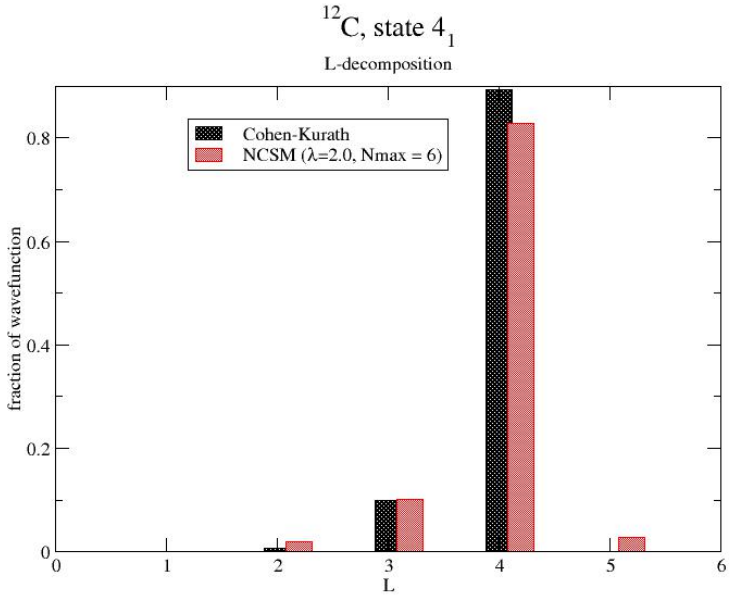


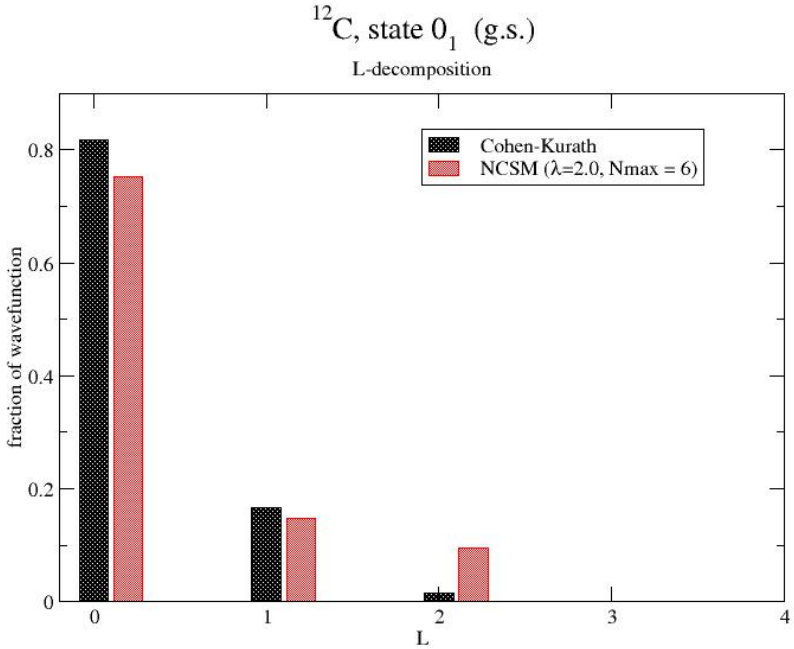
*code courtesy of P. Navratil,
any mistakes in using it are mine!

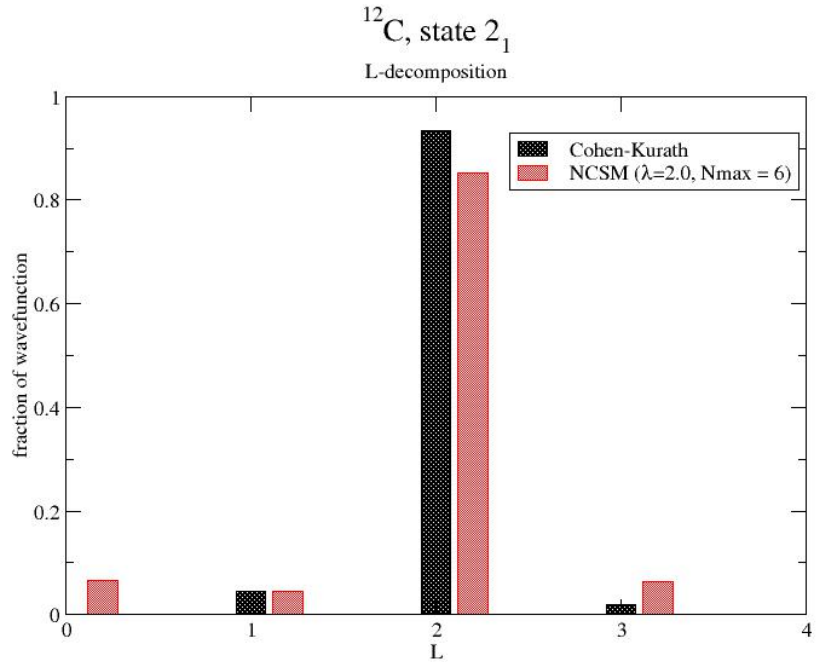
And the (rotational) band played on...

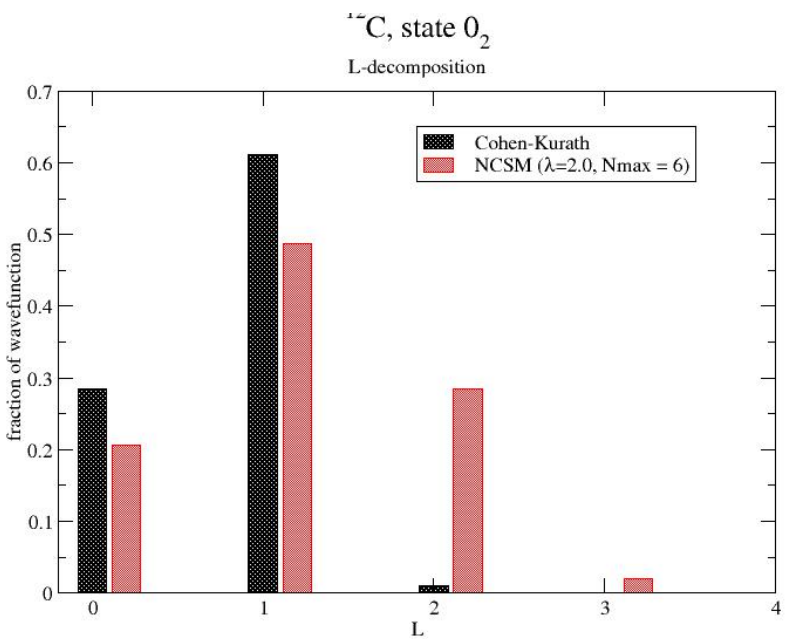


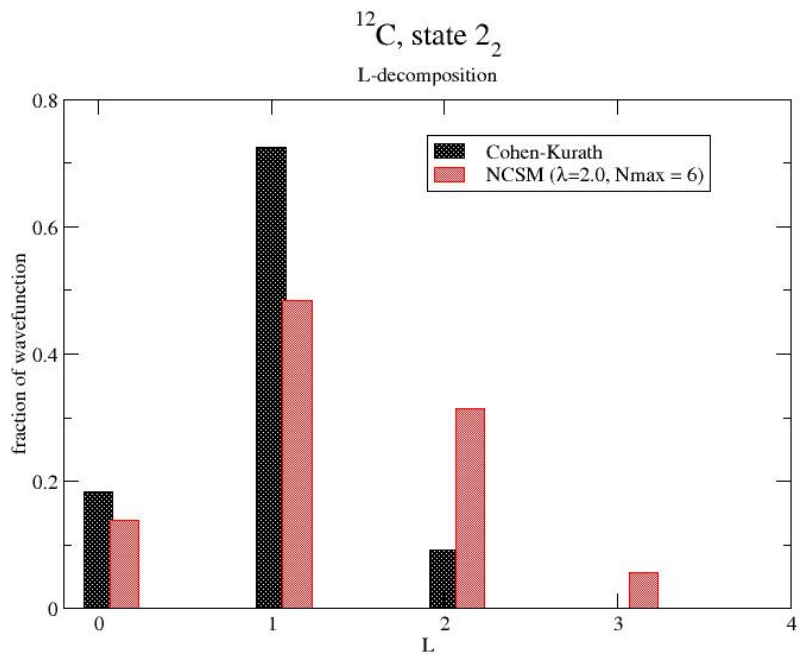





^{12}C, state 0_1 (g.s.)
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L


^{12}C, state 2_1
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

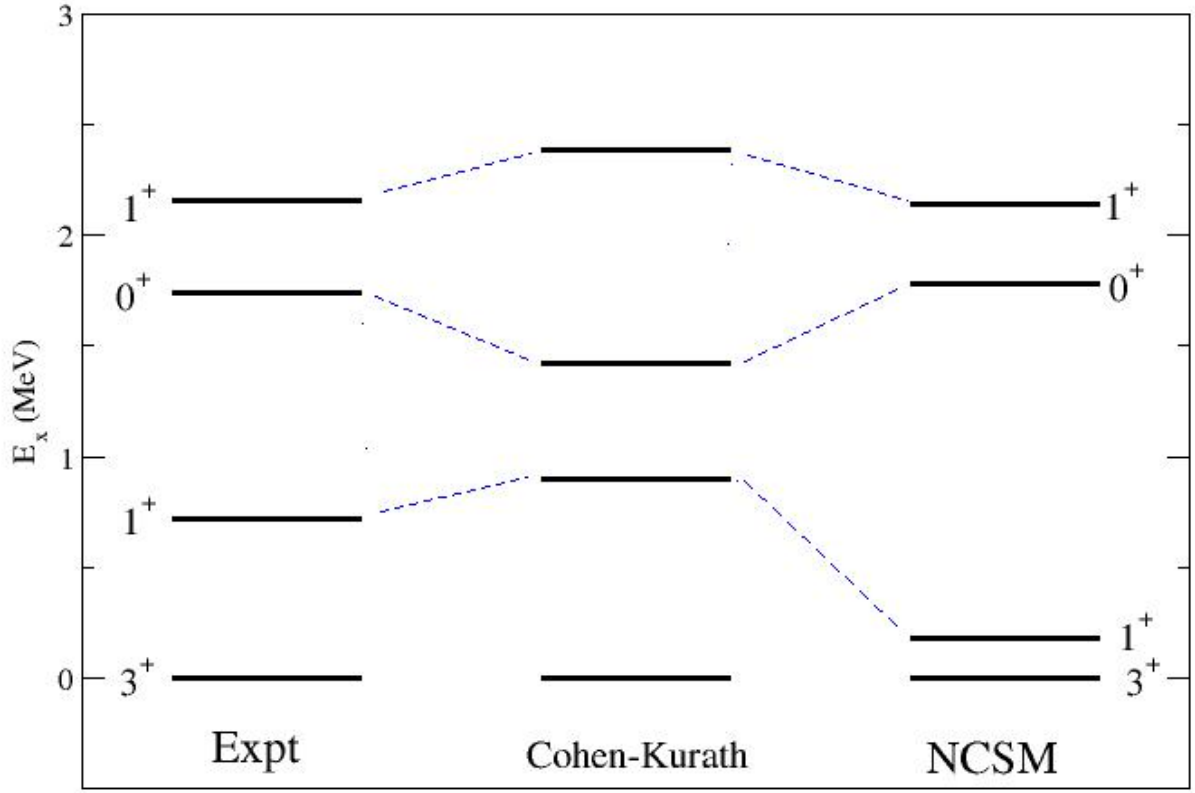

^{12}C, state 0_2
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

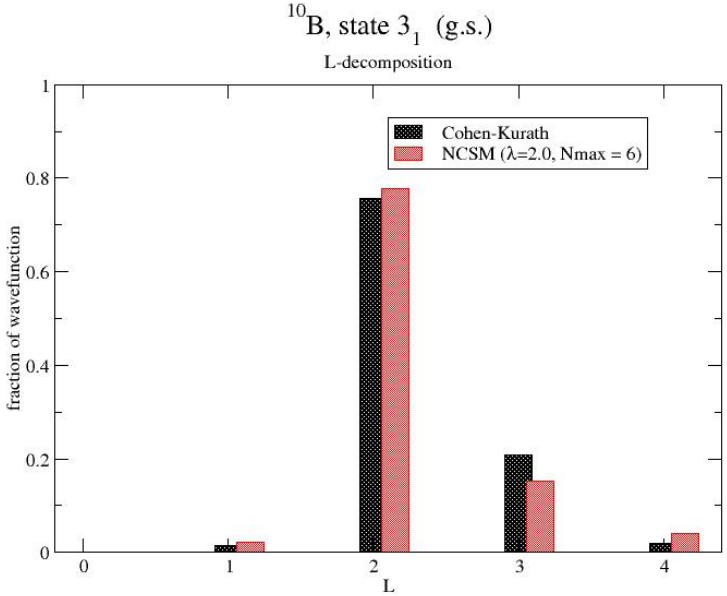

^{12}C, state 2_2
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

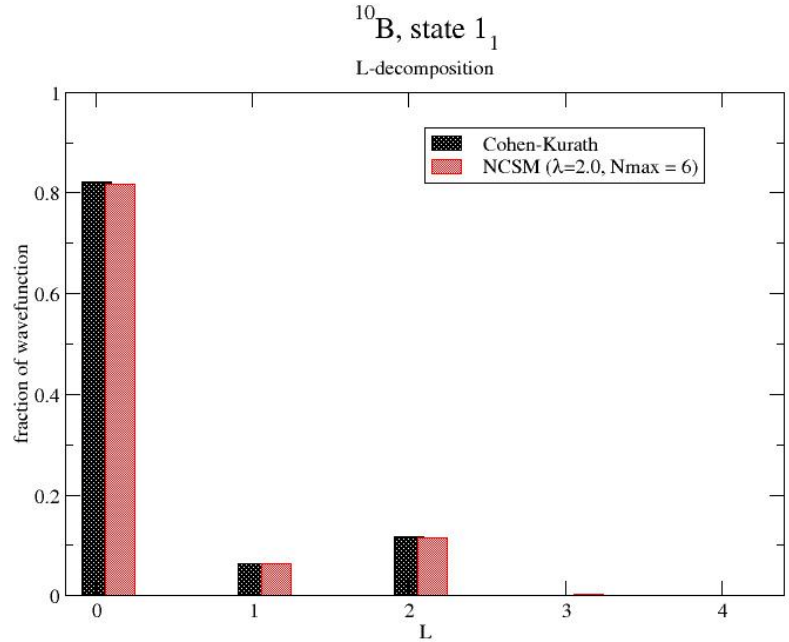
^{10}B

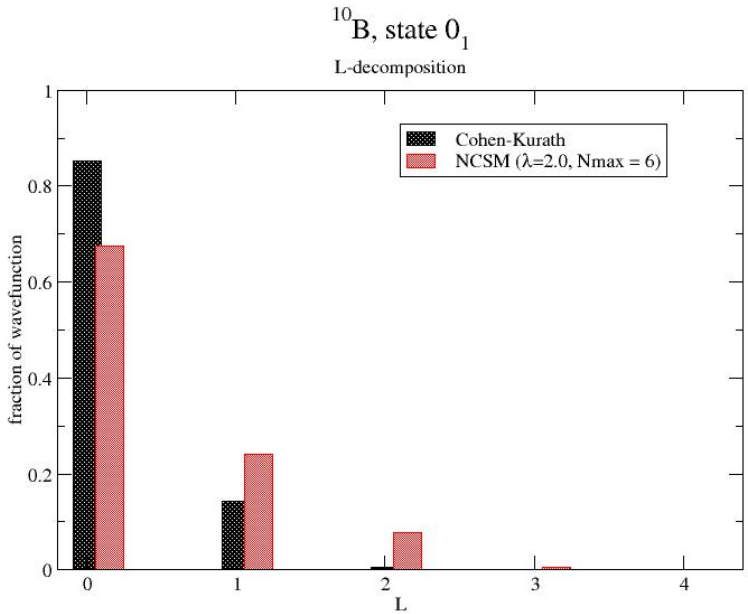
Phenomenological Cohen-Kurath *m*-scheme dimension: 84

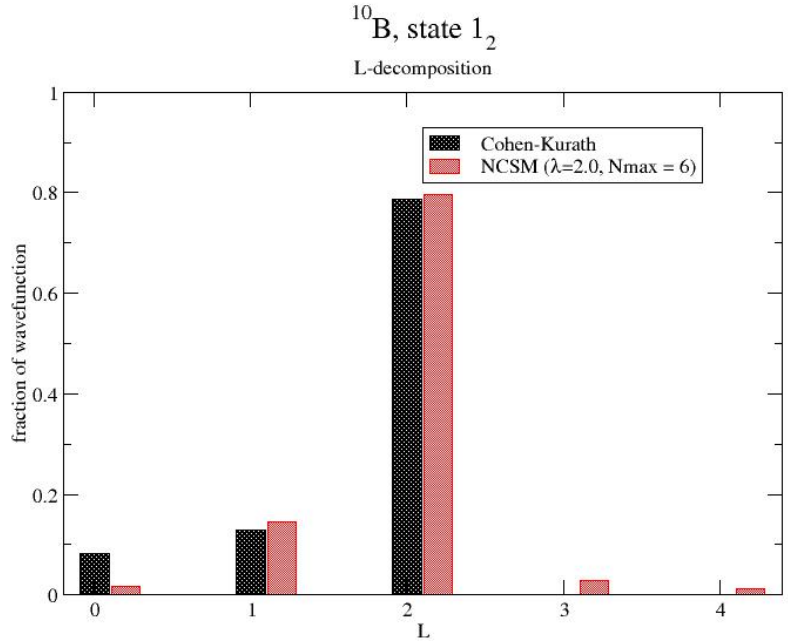
NCSM: N3LO chiral 2-body force SRG evolved to λ = 2.0 fm^{-1}, N_{max} = 6, $\hbar\omega$=22 MeV
m-scheme dimension: 12 million




^{10}B, state 3_1 (g.s.)
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L


^{10}B, state 1_1
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

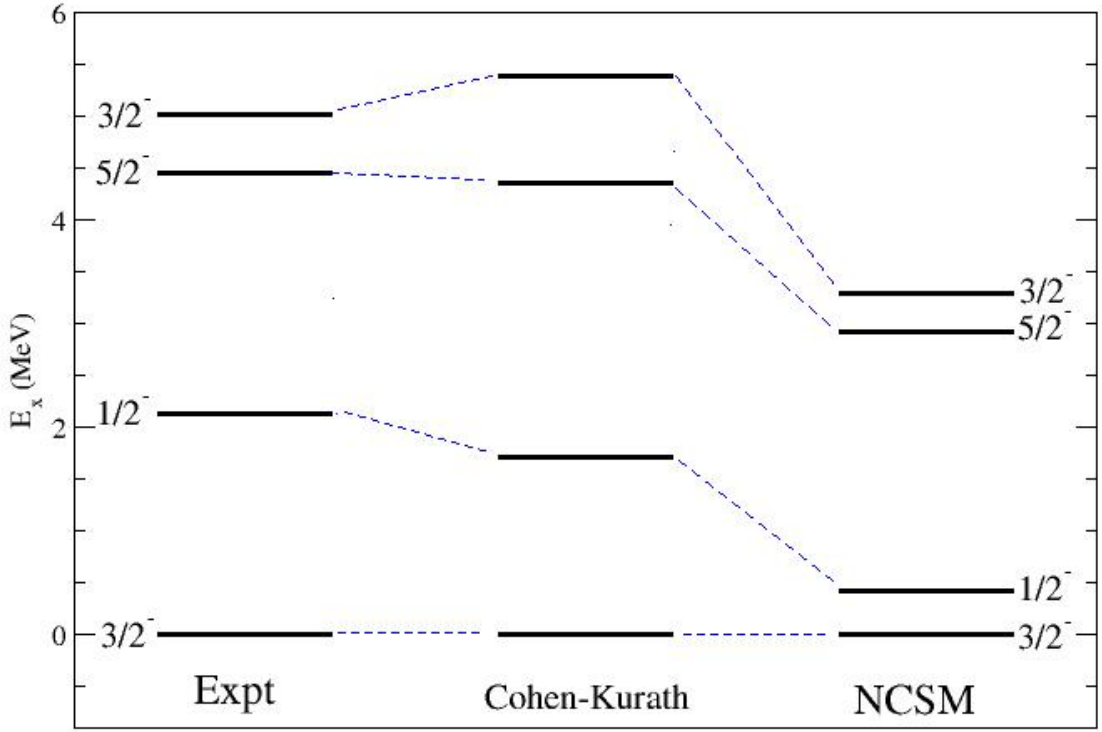

^{10}B, state 0_1
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

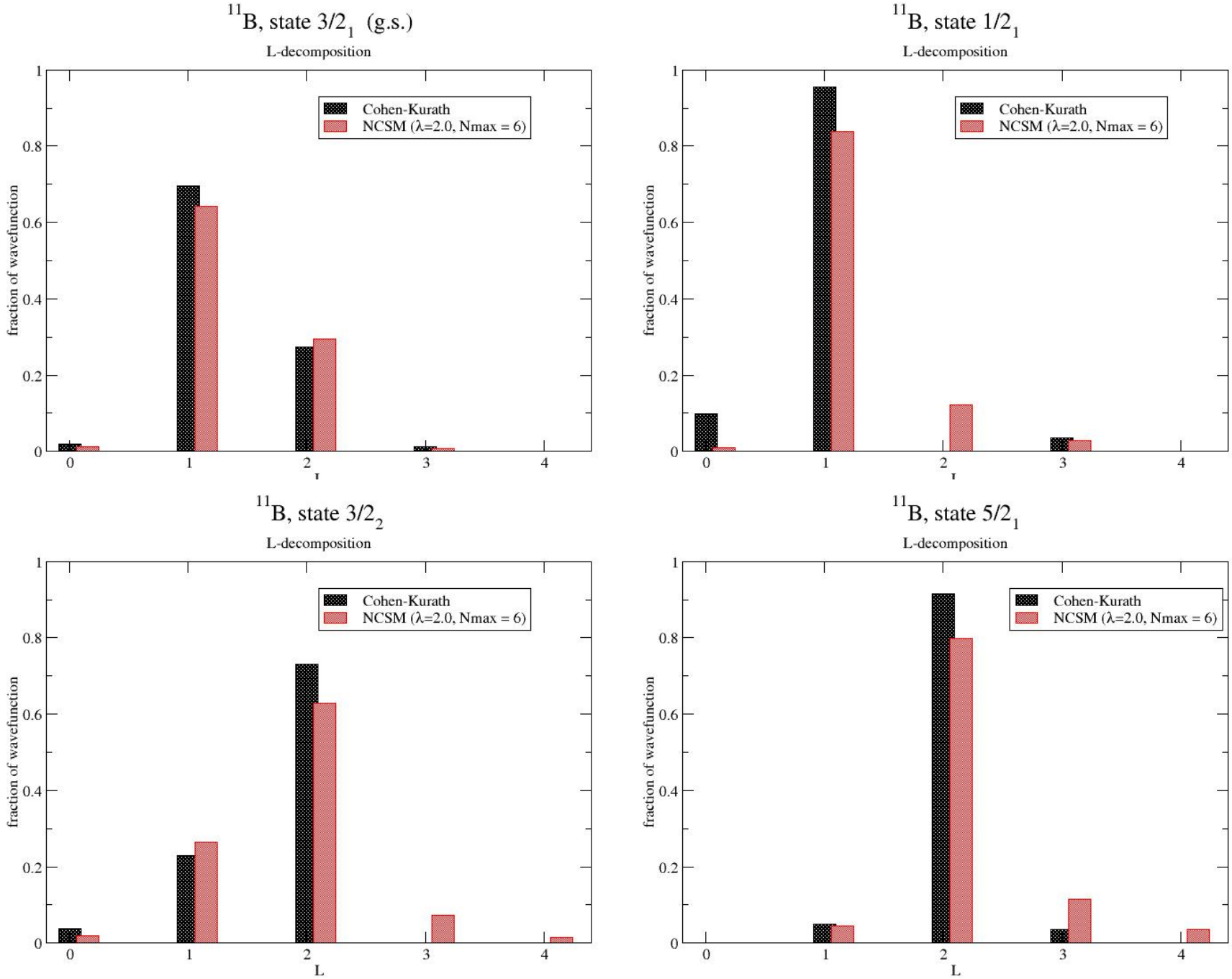

^{10}B, state 1_2
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

^{11}B

Phenomenological Cohen-Kurath *m*-scheme dimension: 62

NCSM: N3LO chiral 2-body force SRG evolved to λ = 2.0 fm^{-1}, N_{max} = 6, $\hbar\omega$=22 MeV
m-scheme dimension: 20 million




^{11}B, state 3/2$_1$ (g.s.)
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
^{11}B, state 1/2$_1$
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
^{11}B, state 3/2$_2$
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L
^{11}B, state 5/2$_1$
L-decomposition
Cohen-Kurath
NCSM (λ=2.0, Nmax = 6)
fraction of wavefunction
L

BONUS ROUND

Advanced topic in factorization: Even *more* divide and conquer!

Consider (proton) Slater determinants:

$n(l)_j$:			$0d_{5/2}$			
m	: 5/2	3/2	½	-1/2	-3/2	-5/2
	1	1	1	0	0	0
	1	1	0	1	0	0
	1	1	0	0	1	0
	1	0	1	1	0	0
	…..					

First, a Slater determinant composed of all single particle states with $m > 0$…

Next, a Slater determinant composed of all single particle states with $m < 0$…

Advanced topic in factorization: Even *more* divide and conquer!

We can combine these *half Slater determinants* into a "full" Slater determinant, in the same way that we combined proton and neutron Slater determinants into the final many-body basis.

Nuclide	Basis dim	# pSDs	# half Slater Determinants
^{20}Ne	640	66	22
^{24}Mg	28,503	495	57
^{28}Si	93,710	924	64
^{48}Cr	1,963,461	4895	386
^{52}Fe	109,954,620	38,760	848
^{56}Ni	1,087,455,228	125,970	1,013

Advanced topic in factorization:
Even *more* divide and conquer!

Sample numbers:

Nuclide	Basis dim	# pSDs	# half Slater Determinants
^{12}C (4hw)	1.1 M	33,475	5448
^{12}C (6hw)	32.6 M	381,159	40,247
^{12}C (8hw)	594 M	2.9 M	232,553
^{16}O (8hw)	996 M	5M	497,493

Advanced topic in factorization:
Even *more* divide and conquer!

Note that while all proton and neutron SDs have the same particle number, we build SDs from half-SDs with differing # of particles (but the sum is fixed—just another quantum number).

This leads to another innovation.

The fundamental operation on half-SDs is not jumps but "hops" which are single-particle creation/annihilation.

This turns out to be natural, easy, and quick.

Half-SDs are generated recursively:

$N_h = 0$:	000000				
$N_h = 1$:	100000	010000	001000	000100	….
$N_h = 2$:	110000	011000	001100	000110	….
	101000	010100	001010	000101	….
	100100	010010	001001		….
	100010	010001			….
	100001				

Each half-SD has a fixed number of *destruction hops*:
it takes only a very short search to find the final half-SD:

$N_h = 0$: 000000

$N_h = 1$: 100000 010000 001000 000100 ….

$N_h = 2$: 110000 011000 001100 000110 ….
101000 010100 001010 000101 ….
100100 010010 001001 ….
100010 010001 ….
100001

Finding all the *creation hops* is even easier,because
we just reverse the destruction hops:

Like the number of half-SDs, the number of hops is small

^{28}Si: 192 hops

^{52}Fe: 3820 hops

^{12}C (6hw): 171,409 hops

^{12}C (8hw): 1,061,255 hops

Using hops we can build arbitrary operations :
1-body jumps, 2-body jumps, 3-body jumps,
spectroscopic factors, etc, all using the same underlying structure.

The technology of half-Slater determinants (or *haikus*) and hops are incorporated into BIGSTICK

- Speeds up basis generation by factor of 3x to 4x
- Speeds up construction of jumps by factor of 10x
- Will allow natural method to compute spectroscopic factors (in progress)

CONCLUSIONS

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

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*