Add, Multiple, Divide...and Conquer:
Factorization Algorithms for Large-Scale Many-Body Problems

Collaborators:

W. Erich Ormand, Lawrence Livermore

Plamen G. Krastev, SDSU

Hai Ah Nam, SDSU/ Oak Ridge

Collaborators-in-training:

Joshua Staker & Micah Schuster, SDSU

THE KEY IDEAS

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

Lanczos algorithm

fundamental operation is *matrix-vector multiply*

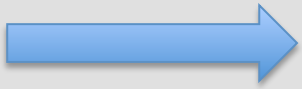
Despite sparsity, nonzero matrix elements can require TB of storage

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 BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix



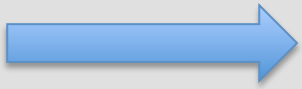
Lanczos algorithm

fundamental operation is *matrix-vector multiply*

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.

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix



Lanczos algorithm

fundamental operation is *matrix-vector multiply*

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

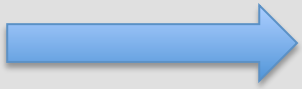
$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \sum_{i<j} V(\vec{r}_i - \vec{r}_j)\right)\Psi(\vec{r}_1,\vec{r}_2,\vec{r}_3\ldots) = E\Psi$$

or

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

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

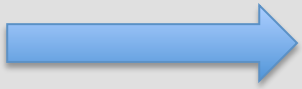
$$\left(\sum_i -\frac{\hbar^2}{2m}\nabla^2 + U(r_i) + \sum_{i<j} V(\vec{r}_i - \vec{r}_j) \right) \Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3 \ldots) = E\Psi$$

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



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

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

THE BASIC PROBLEM

Find extremal eigenvalues of very large, very sparse Hermitian matrix

→ Lanczos algorithm

fundamental operation is *matrix-vector multiply*

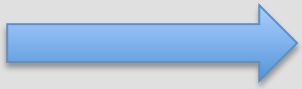
$$H_{\alpha\beta} = \langle \alpha | \hat{\mathbf{H}} | \beta \rangle$$

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

→ Lanczos algorithm!

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**: Householder

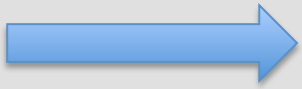
Find orthogonal matrix **U** such that $\mathbf{U}^T \mathbf{A} \mathbf{U} = \mathbf{B}$, a tridiagonal matrix

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*

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

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

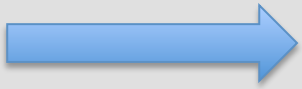
$$\mathbf{A}\vec{v}_3 = \beta_2\vec{v}_2 + \alpha_3\vec{v}_3 + \beta_3\vec{v}_4$$

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

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

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

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

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

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")

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

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"

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

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

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

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

FACTORIZATION

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

A quantum number is the eigenvalue of an operator

generally a operator that exactly commutes with the Hamiltonian

e.g. angular momentum $\mathbf{J}^2$ and z-component $\mathbf{J}_z$

$$\hat{J}^2|\Psi\rangle = J(J+1)|\Psi\rangle \qquad \hat{J}_z|\Psi\rangle = M|\Psi\rangle$$

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

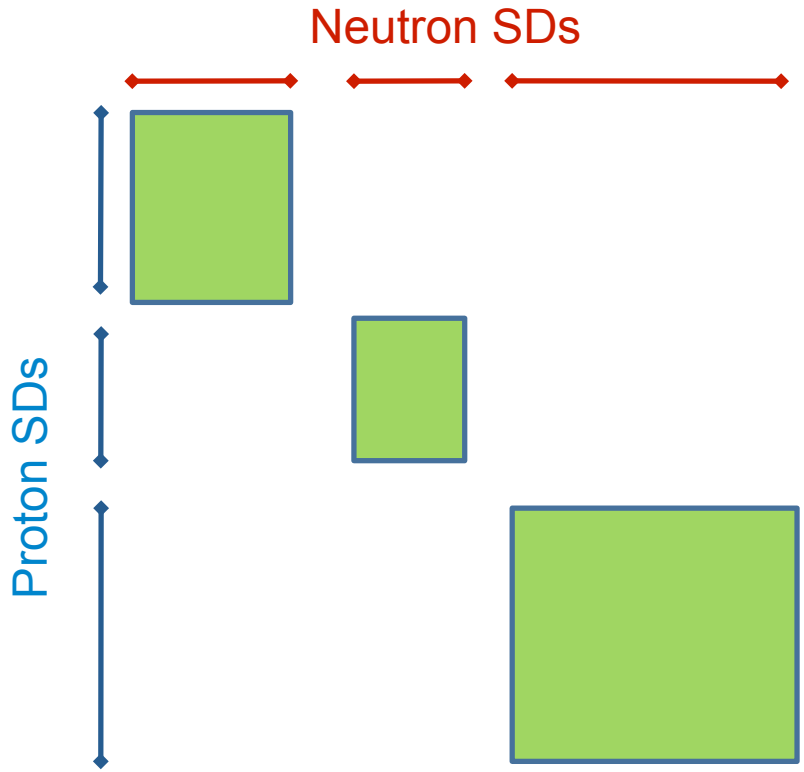
$M_z(\pi) = -4$: 2 SDs | $M_z(\upsilon) = +4$: 24 SDs | 48 combined

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

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

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

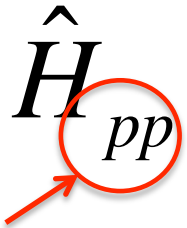
Move 2 protons; *neutrons are spectators*

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*



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

$$\begin{matrix} |\pi_1\rangle \\ |\pi_2\rangle \end{matrix} \qquad H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \qquad \begin{matrix} |\nu_1\rangle \\ |\nu_2\rangle \\ |\nu_3\rangle \\ |\nu_4\rangle \\ \vdots \\ |\nu_{24}\rangle \end{matrix}$$

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

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

$$\begin{aligned}
|\nu_1\rangle \quad & H_{pp}|\pi_1\rangle|\nu_1\rangle = H_{11}|\pi_1\rangle|\nu_1\rangle + H_{12}|\pi_2\rangle|\nu_1\rangle \\
|\nu_2\rangle \quad & H_{pp}|\pi_2\rangle|\nu_1\rangle = H_{12}|\pi_1\rangle|\nu_1\rangle + H_{22}|\pi_2\rangle|\nu_1\rangle \\
|\nu_3\rangle \quad & H_{pp}|\pi_1\rangle|\nu_2\rangle = H_{11}|\pi_1\rangle|\nu_2\rangle + H_{12}|\pi_2\rangle|\nu_2\rangle \\
|\nu_4\rangle \quad & H_{pp}|\pi_2\rangle|\nu_2\rangle = H_{12}|\pi_1\rangle|\nu_2\rangle + H_{22}|\pi_2\rangle|\nu_2\rangle \\
\vdots \quad & \vdots \\
|\nu_{24}\rangle \quad & H_{pp}|\pi_1\rangle|\nu_{24}\rangle = H_{11}|\pi_1\rangle|\nu_{24}\rangle + H_{12}|\pi_2\rangle|\nu_{24}\rangle \\
& H_{pp}|\pi_2\rangle|\nu_{24}\rangle = H_{12}|\pi_1\rangle|\nu_{24}\rangle + H_{22}|\pi_2\rangle|\nu_{24}\rangle
\end{aligned}$$

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*

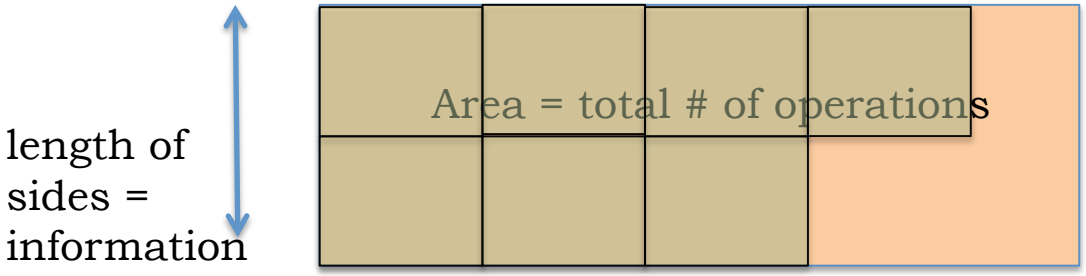
Comparison of nonzero matrix storage with factorization

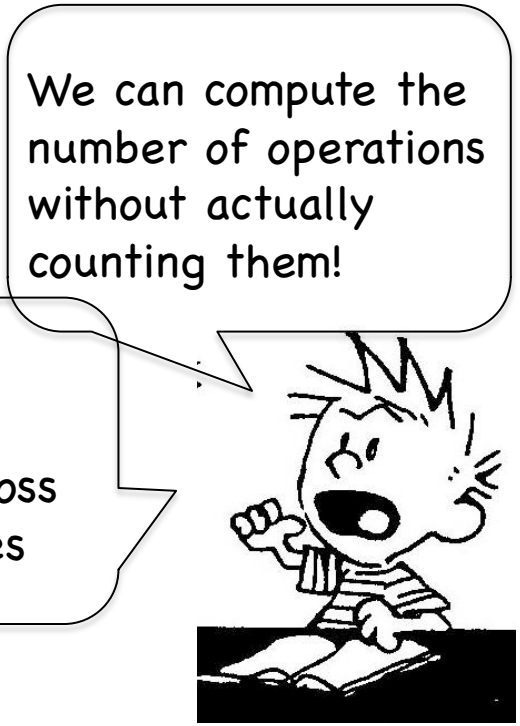
Nuclide	Space	Basis dim	matrix store	factorization
^{56}Fe	*pf*	501 M	290 Gb	0.72 Gb
^{7}Li	N_{max}=12	252 M	3600 Gb	96 Gb
^{7}Li	N_{max}=14	1200 M	23 Tb	624 Gb
^{12}C	N_{max}=6	32M	196 Gb	3.3 Gb
^{12}C	N_{max}=8	590M	5000 Gb	65 Gb
^{12}C	N_{max}=10	7800M	111 Tb	1.4 Tb
^{16}O	N_{max}=6	26 M	142 Gb	3.0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

PARALLEL IMPLEMENTATION

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

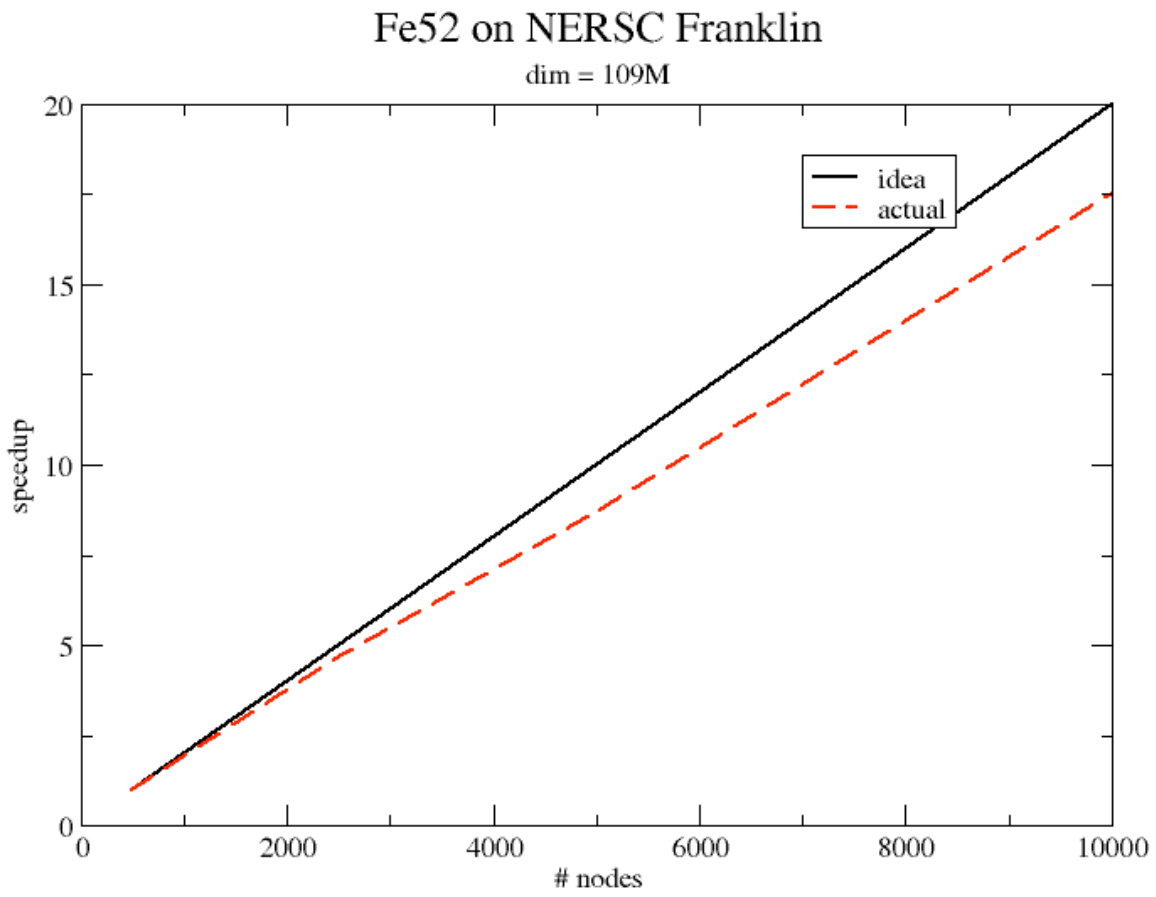
length of sides = information to be stored





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 late 2011

SCIENCE APPLICATIONS

Comparison of "exact" results versus approximations
-- *e.g.* mean-field, density-functional, time-dependent mean-field, projection of mean-field onto exact symmetries, *etc.*

How "unique" in the nucleon-nucleon interaction?
--*i.e.*, the input V_{ijkl} look like random numbers; what happens if we actually use random numbers?
-- can we use ambiguity in the interaction to our advantage?

Looking for and using broken symmetries
--isospin breaking and the unitarity of the CKM matrix

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*