Nuclear Talent course: Computational Many-body Methods for Nuclear Physics

Morten Hjorth-Jensen$^{(1)}$, Calvin Johnson$^{(2)}$, Francesco Pederiva$^{(3)}$, and Kevin Schmidt$^{(4)}$

$^{(1)}$Michigan State University and University of Oslo  
$^{(2)}$San Diego State University  
$^{(3)}$University of Trento  
$^{(4)}$Arizona State University

ECT*, June 25-July 13  
Lecture slides with exercise for Thursday July 5 2012
Thursday: introduction to shell-model diagonalization

- Outline of shell-model project
- Solving a linear differential equation as a matrix – why we do diagonalization
- Slater determinants as basis states
An alternative approach to solving the many-body problem is to treat it as diagonalization of a (large) matrix. In these lectures we will cover:

- Motivation for solving a differential equation as a matrix eigenvalue problem
- The technical tools for casting the many-body problem as a matrix: Review of occupation representation and use of (fermion) creation/annihilation
- How to represent the many-body basis states as Slater determinants
- How to compute matrix elements of the Hamiltonian between Slater determinants
- How to solve the matrix eigenvalue problem: Givens, Householder, and Lanczos algorithms
- Advanced technical topics
Outline of shell-model project

Specific tasks you will do towards building a working shell-model diagonalization code:

- Today (Thursday): Read in quantum numbers of the single-particle space and construct M-scheme basis;
- Friday: Eigensolution algorithms and applying Lanczos to a pre-determined matrix (pairing problem)
- Monday: Read in two-body matrix elements and compute many-body Hamiltonian matrix elements
- Tuesday: Continue to compute many-body Hamiltonian matrix elements; first solution using Householder
- Wednesday: Solve for many-body eigenvalues using Lanczos
- Thursday: Apply to 4 neutrons in 4 major shells using Mafliet-Tjon interaction

Good luck!
Many-body quantum mechanics

Why (and how) do we write the many-body Schrödinger equation as a matrix diagonalization problem?
Many-body quantum mechanics

We all know how to write down and solve the Schrödinger equation in coordinate space for one particle:

$$-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \psi(x) + V(x)\psi(x) = E\psi(x)$$

The generalization to A-body systems is straightforward to write down, e.g.,

$$\left( \sum_i -\frac{\hbar^2}{2m} \nabla_i^2 + U(\vec{r}_i) + \sum_{i,j} V(\vec{r}_i - \vec{r}_j) \right) \psi(\vec{r}_1, \vec{r}_2, \vec{r}_3, \ldots)$$

$$= E\psi(\vec{r}_1, \vec{r}_2, \vec{r}_3, \ldots)$$

But it is easier to write down an equation than to solve it!
Linear differential equations

Instead we solve a linear differential equation as a matrix problem.

Let \( \hat{L} \) be a linear differential operator, operating in a suitably defined vector function space, \( \{ |f\rangle \} \). (NB: You are assumed to be comfortable with the concept of linear vector spaces.) For any vector space we can choose a basis set, e.g. \( \{ |\phi_\alpha\rangle \} \).

Then, because of the properties of a linear vector space, we can always expand any function in terms of the basis.

\[
|f\rangle = \sum_\alpha c_\alpha |\phi_\alpha\rangle
\]

This is a fairly abstract concept, but you should get used to thinking abstractly!
Solving a linear differential equation as a matrix

Now it’s straightforward to rewrite a differential equation as a matrix equation

Start with the differential equation

\[ \hat{L}|\psi\rangle = |f\rangle \]

Take an inner produce with a basis state \( \langle \phi_\alpha | \):

\[ \langle \phi_\alpha | \hat{L}|\psi\rangle = \langle \phi_\alpha | f \rangle \]

and then insert the completeness relation

\[ 1 = \sum_\beta |\phi_\beta\rangle\langle \phi_\beta | \]

to get

\[ \sum_\beta \langle \phi_\alpha | \hat{L}|\phi_\beta\rangle\langle \phi_\beta |\psi\rangle = \langle \phi_\alpha | f \rangle \]
Solving a linear differential equation as a matrix

We call

\[ L_{\alpha \beta} = \langle \phi_{\alpha} | \hat{L} | \psi \rangle \]

the *matrix element* of the operator \( L \) (these are c-numbers and the values depend on the choice of basis), while \( \psi_{\beta} = \langle \phi_{\beta} | \psi \rangle \) and \( f_{\alpha} = \langle \phi_{\alpha} | f \rangle \) are the *vector components*, also basis-dependent, of the functions (vectors) \( |\psi\rangle \) and \( |f\rangle \). We now left with the matrix equation

\[ L \vec{\psi} = \vec{f} \]

In quantum mechanics we often seek to solve an eigenvalue equation for a Hamiltonian operator \( \hat{H} \), that is,

\[ \hat{H} |\psi\rangle = E |\psi\rangle \]

which gets rewritten in matrix form

\[ H \vec{\psi} = E \vec{\psi} \]
Solving a linear differential equation as a matrix

In these lectures we will outline in detail how to:

▸ Construct (orthonormal) basis states for many-particle systems, called *Slater determinants*; and

▸ Show how to compute *matrix elements* of operators between these Slater determinant basis states using creation and annihilation operators

▸ Discuss in detail the computational linear algebra needed to find the eigensolutions, primarily the Householder and Lanczos diagonalization algorithms.

It’s important that you master these basic ideas, so if you get lost, please ask questions! You probably aren’t the only one lost.
Consider a linear differential equation, for example,

$$\hat{H} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} - \lambda \delta(x).$$

This problem is standard in beginning quantum mechanics; the ground state energy

$$E = -\frac{m\lambda^2}{2\hbar^2}.$$ 

We’re going to compute this as a matrix using two very different bases:

- cosines in a box of length $L$;
- harmonic oscillator wavefunctions.
First we restrict ourselves to a box from $x = -L/2$ to $x = L/2$. The basis eigenstates are sine and cosine functions; the ones that interest us here are the even functions $\phi_n(x) = \langle x|n \rangle = \sqrt{2/L} \cos((2n + 1)\pi x/L)$, $n = 0, 1, 2, \ldots$, which vanish at the walls of the box.

The matrix elements of the kinetic energy are diagonal in the basis of sines and cosines (which is why I choose it; for a general basis the kinetic energy matrix elements will not be diagonal):

$$\left\langle m \left| -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \right| n \right\rangle = \delta_{mn} \frac{\hbar^2 \pi^2 (2n + 1)^2}{2mL^2}.$$

The matrix element of the delta-function potential between any two of these is simple:

$$\langle m | -\lambda \delta(x) | n \rangle = -\frac{\lambda}{L},$$

independent of $m$ and $n$. 

Solving the $\delta$-potential as a matrix
Solving the $\delta$-potential as a matrix

Define

$$L = L_0 \frac{\hbar^2}{m\lambda}$$

where $L_0$ is a dimensionless parameter.

Then the matrix element is

$$H_{mn} = \frac{m\lambda^2}{\hbar^2} \times \left( -\frac{2}{L_0} + \delta_{mn} \frac{\pi^2}{2L_0^2} (2n + 1)^2 \right).$$

Clearly we have a dependence on the box size parameterized by $L_0$. This is often true of many calculations and is a crude example of a cutoff. We’ll soon see there are tradeoffs involved in the choice of $L_0$. 
Solving the $\delta$-potential as a matrix

First choose $L_0 = \pi^2/2$ (which happens to be motivated by variational theory).

If we take only the lowest state, $n = 0$, then

$E \approx H_{00} = -(2/\pi^2)m\lambda^2/\hbar^2 = -0.2026m\lambda^2/\hbar^2$ which should be compared to exact ground state of $-0.5m\lambda^2/\hbar^2$.

Now let’s take $n = 0, 1$. The matrix is now a $2 \times 2$

$$
\begin{pmatrix}
\frac{2}{\pi^2} \frac{m\lambda^2}{\hbar^2} & 0 \\
0 & \frac{2}{\pi^2} \frac{m\lambda^2}{\hbar^2}
\end{pmatrix}
\begin{pmatrix}
-1 & -2 \\
-2 & 7
\end{pmatrix}, \tag{1}
$$

and the ground state energy is $(-6 + 4\sqrt{5})/\pi^2 \times (m\lambda^2/\hbar^2) = -0.2983m\lambda^2/\hbar^2$ (a small improvement)
Solving the $\delta$-potential as a matrix

We can easily choose larger matrices, and examine how the ground state energy converges:
By looking at the error on a log plot, we can see the tradeoffs: while a larger box (cutoff) ultimately converges to a better result, for small dimensions the smaller boxes yield better approximations.

This illustrates the difficulty sometimes in choosing the ‘best’ basis.
Solving the \( \delta \)-potential as a matrix

We can also see how the exact wavefunction, \( \psi(x) = (\sqrt{m\lambda/\hbar}) \exp(-m\lambda|x|/\hbar^2) \). is built up:
Solving the $\delta$-potential as a matrix

It’s important to note that we have considerable freedom in choosing a basis, and that the indexing of the basis is somewhat arbitrary. We’ve already examined choosing cosine functions of different size; these were labeled by $n = 0, 1, 2, 3$.

Now let’s choose harmonic oscillator wavefunctions $\psi_n(x)$. We only need even $n$, as odd wavefunctions vanish at the origin and do not ‘see’ the $\delta$-potential.

We characterize the harmonic oscillator by the length parameter $b = \sqrt{\hbar/m\omega}$. The kinetic energy is straightforward to compute, though no longer diagonal as it was for the cosine basis:

$$\langle n' | \hat{p}^2/2m | n \rangle = \frac{\hbar^2}{4mb^2} \left( \delta_{n',n}(2n+1) - \delta_{n',n+2} \sqrt{(n+1)(n+2)} - \delta_{n,n'+2} \sqrt{(n'+1)(n'+2)} \right).$$

while the matrix element of the $\delta$-potential is

$$\langle n' | \delta(x) | n \rangle = \int \psi_{n'}^*(x)\delta(x)\psi_n(x)dx = \psi_{n'}(0)\psi_n(0),$$

For $n$ even,

$$\psi_n(0) = (-1)^{n/2} \sqrt{\frac{n!}{b2^n(\sqrt{\pi}\frac{n!}{2})^2}},$$
Solving the $\delta$-potential as a matrix

By applying the variational theorem on the expectation value for the state $n = 0$, we get an optimal value

$$b = \frac{\hbar^2 \sqrt{\pi}}{2m\lambda}.$$ 

Taking $n = 0, 2, 4, 6$ we have

$$H = \frac{m\lambda^2}{\hbar^2 \pi} \begin{pmatrix}
-1 & 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{5}}{2} \\
0 & 4 & -3\sqrt{3}/2 & -\sqrt{5}/8 \\
-\sqrt{3}/2 & -3\sqrt{3}/2 & 33/4 & -7\sqrt{30}/8 \\
\sqrt{5}/2 & -\sqrt{5}/8 & -7\sqrt{30}/8 & 99/8
\end{pmatrix}.$$
Solving the $\delta$-potential as a matrix

Although it starts off at a lower value, the numerical convergence is slower with the harmonic oscillator basis than for the cosine basis.
Solving the $\delta$-potential as a matrix

To again dissect the convergence we put the difference between the numerical energy and the exact value on a log-log plot.
What have we learned?

- We can rewrite a linear differential equation (including eigenvalue equations) as matrix equations;
- We can choose different basis states and truncate the number of basis states;
- As we increase the number of basis states, the better our answer;
- The rate at which we converge to the correct answer depends on the choice of basis;
- Furthermore, because of approximations in the basis (e.g., a cutoff), we might not ultimately converge to the correct answer, nonetheless judicious choice in the cutoff might speed convergence.

Now we are going to do the same, but for many-body systems.
The simplest possible choice for many-body wavefunctions are *product* wavefunctions.

\[ \psi(x_1, x_2, x_3, \ldots, x_A) \approx \phi_1(x_1) \phi_2(x_2) \phi_3(x_3) \ldots \]

because we are really only good at thinking about one particle at a time. Such product wavefunctions, without correlations (e.g. Jastrow-type functions), are easy to work with; for example, if the single-particle states \( \{\phi_i(x)\} \) are orthonormal, then the product wavefunctions are easy to orthonormalize.

Similarly, computing matrix elements of operators are relatively easy, because the integrals factorize.

The price we pay is the lack of correlations, which we must build up by using many, many product wavefunctions. (Thus we have a trade-off: compact representation of correlations but difficult integrals versus easy integrals but many states required.)
Slater determinants as basis states

Because we have fermions, we are required to have antisymmetric wavefunctions, e.g.

$$\Psi(x_1, x_2, x_3, \ldots, x_A) = -\Psi(x_2, x_1, x_3, \ldots, x_A)$$

etc. This is accomplished formally by using the determinantal formalism:

$$\Psi(x_1, x_2, \ldots, x_N) = \frac{1}{\sqrt{N!}} \det \begin{vmatrix} \phi_1(x_1) & \phi_1(x_2) & \ldots & \phi_1(x_N) \\ \phi_2(x_1) & \phi_2(x_2) & \ldots & \phi_2(x_N) \\ \vdots & \vdots & & \vdots \\ \phi_N(x_1) & \phi_N(x_2) & \ldots & \phi_N(x_N) \end{vmatrix}$$

Product wavefunction + antisymmetry = Slater determinant.
Slater determinants as basis states

\[
\Psi(x_1, x_2, \ldots, x_N) = \frac{1}{\sqrt{N!}} \det \begin{vmatrix}
\phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_N) \\
\phi_2(x_1) & \phi_2(x_2) & \cdots & \phi_2(x_N) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_N(x_1) & \phi_N(x_2) & \cdots & \phi_N(x_N)
\end{vmatrix}
\]

Properties of the determinant (interchange of any two rows or any two columns yields a change in sign; thus no two rows and no two columns can be the same) lead to the Pauli principle:

- No two particles can be at the same place (two columns the same); and
- No two particles can be in the same state (two rows the same).
Slater determinants as basis states

As a practical matter, however, Slater determinants beyond $N = 4$ quickly become unwieldy. Thus we turn to the *occupation representation* or *second quantization* to simplify calculations.

The occupation representation, using fermion *creation* and *annihilation* operators, is compact and efficient. It is also abstract and, at first encounter, not easy to internalize. It is inspired by other operator formalism, such as the ladder operators for the harmonic oscillator or for angular momentum, but unlike those cases, the operators do not have coordinate space representations.

Instead, one can think of fermion creation/annihilation operators as a game of symbols that compactly reproduces what one would do, albeit clumsily, with full coordinate-space Slater determinants.
Instant tour of the occupation representation

We start with a set of orthonormal single-particle states \( \{ \phi_i(x) \} \). (Note: this requirement, and others, can be relaxed, but leads to a more involved formalism.) Any orthonormal set will do.

To each single-particle state \( \phi_i(x) \) we associate a creation operator \( \hat{a}^\dagger_i \) and an annihilation operator \( \hat{a}_i \).

When acting on the vacuum state \( |0\rangle \), the creation operator \( \hat{a}^\dagger_i \) causes a particle to occupy the single-particle state \( \phi_i(x) \):

\[
\phi_i(x) \rightarrow \hat{a}^\dagger_i |0\rangle
\]
But with multiple creation operators we can occupy multiple states:

$$\phi_i(x)\phi_j(x')\phi_k(x'') \rightarrow \hat{a}_i\hat{a}_j\hat{a}_k|0\rangle.$$ 

Now we impose antisymmetry, by having the fermion operators satisfy anticommutation relations:

$$\hat{a}_i\hat{a}_j + \hat{a}_j\hat{a}_i = [\hat{a}_i, \hat{a}_j] = \{\hat{a}_i, \hat{a}_j\} = 0$$

so that

$$\hat{a}_i\hat{a}_j = -\hat{a}_j\hat{a}_i$$

Because of this property, automatically $\hat{a}_i\hat{a}_j = 0$, enforcing the Pauli exclusion principle. Thus when writing a Slater determinant using creation operators,

$$\hat{a}_i\hat{a}_j\hat{a}_k \ldots |0\rangle$$

each index $i, j, k, \ldots$ must be unique.
Instant tour of the occupation representation

The annihilation operators also anticommute \( \{ \hat{a}_i, \hat{a}_j \} = 0 \) so that

\[
\hat{a}_i \hat{a}_j = -\hat{a}_j \hat{a}_i
\]

Furthermore, in case it is not obvious, the adjoint of a Slater determinant is

\[
\left( \hat{a}_j^\dagger \hat{a}_j^\dagger \hat{a}_k^\dagger \ldots |0\rangle \right)^\dagger = \langle 0| \ldots \hat{a}_k \hat{a}_j \hat{a}_i.
\]

We need two more rules. The first is that an annihilation operator \( \hat{a}_i \) acting on a ket vacuum \( |0\rangle \ldots \text{annihilates} \) it entirely:

\[
\hat{a}_i |0\rangle = 0
\]

(also the adjoint: \( \langle 0| \hat{a}_i^\dagger = 0 \));

furthermore the creation and annihilation operators with the same index \( i \) have a special anticommutator:

\[
\{ \hat{a}_i, \hat{a}_j^\dagger \} = \delta_{ij}
\]

so that if \( i, j \) are different the operators always anticommute.
Instant tour of the occupation representation

This last anticommutation relation

\[ \{ \hat{a}_i, \hat{a}_j^\dagger \} = \delta_{ij} \]

allows for correct normalization of many-body states.

Example: Consider the overlap between two 1-body states, \( \phi_i \) and \( \phi_j \). In the occupation representation:

\[ \phi_i \rightarrow \hat{a}_i^\dagger |0\rangle \]

and

\[ \phi_j \rightarrow \hat{a}_j^\dagger |0\rangle \]

The overlap of the two states is thus

\[ \langle \phi_i | \phi_j \rangle \]

\[ = \langle 0 | \hat{a}_i \hat{a}_j^\dagger |0\rangle \]
We now use the anticommutation relation and make the substitution

\[ \hat{a}_i \hat{a}^\dagger_j + \hat{a}^\dagger_j \hat{a}_i = \delta_{ij} \]

and obtain

\[ \langle 0 | \hat{a}_i \hat{a}^\dagger_j | 0 \rangle = \delta_{ij} \langle 0 | 0 \rangle - \langle 0 | \hat{a}^\dagger_j \hat{a}_i | 0 \rangle \]

But in the second term the vacuum is annihilated! Then assuming the vacuum is normalized \( \langle 0 | 0 \rangle = 1 \), we get

\[ \langle 0 | \hat{a}_i \hat{a}^\dagger_j | 0 \rangle = \delta_{ij} \]

as we’d expect.

**Exercise**: Show that the two-particle states \( \hat{a}_1^{\dagger} \hat{a}_2^{\dagger} | 0 \rangle \), \( \hat{a}_1^{\dagger} \hat{a}_3^{\dagger} | 0 \rangle \), and \( \hat{a}_4^{\dagger} \hat{a}_5^{\dagger} | 0 \rangle \) are all normalized, and all orthogonal to each other.
For computational work, we can go further and represent Slater determinants one of two ways:

- We can simply write down a list of occupied states, e.g. \(|1, 2\rangle \langle 2, 3|, |4, 5\rangle, \) etc.;
- or

- We can use bit notation: a one \(1\) for occupied and a zero \(0\) for unoccupied: \(|11000\rangle, |01100\rangle, |00011\rangle\) etc.

Both have their advantages and disadvantages; for small number of particles the list of occupied states is more compact, but for large number of particles storing Slater determinants in bit form can be advantageous and is commonly used.
Building a many-body basis

*Your mission, should you choose to accept it...*

Your assigned project for this module is to construct a working code that constructs the many-body Hamiltonian matrix in a basis of Slater determinants and to find the low-lying eigenenergies. This is referred to as the configuration-interaction method or shell-model diagonalization (or the interacting shell model).

The first step in such codes—and in your project—is to construct the many-body basis. While the formalism is independent of the choice of basis, the *effectiveness* of a calculation will certainly be basis dependent.

Furthermore there are common conventions useful to know.
Building a many-body basis

First, the single-particle basis has angular momentum as a good quantum number. You can imagine the single-particle wavefunctions being generated by a one-body Hamiltonian, for example a harmonic oscillator. Modifications include harmonic oscillator plus spin-orbit splitting, or self-consistent mean-field potentials, or the Woods-Saxon potential which mocks up the self-consistent mean-field.

For nuclei, the harmonic oscillator, modified by spin-orbit splitting, provides a useful language for describing single-particle states.

Each single-particle state is labeled by the following quantum numbers:

- Orbital angular momentum $l$
- Intrinsic spin $s = 1/2$ for protons and neutrons
- Angular momentum $j = l \pm 1/2$
- $z$-component $j_z$ (or $m$)
- Some labeling of the radial wavefunction, typically $n$ the number of nodes in the radial wavefunction, but in the case of harmonic oscillator one can also use the principal quantum number $N$, where the harmonic oscillator energy is $(N + 3/2)\hbar\omega$. 
Building a many-body basis

In this format one labels states by $n(l)_j$, with $(l)$ replaced by a letter: $s$ for $l = 0$, $p$ for $l = 1$, $d$ for $l = 2$, $f$ for $l = 3$, and thenceforth alphabetical.

```
0s \frac{1}{2}  
0p \frac{3}{2}  
0p \frac{1}{2}  
0d \frac{5}{2}  
1s \frac{1}{2}  
0d \frac{3}{2}  
of \frac{7}{2}  
```

In practice the single-particle space has to be severely truncated. This truncation is typically based upon the single-particle energies, which is the effective energy from a mean-field potential.

Sometimes we freeze the core and only consider a valence space. For example, one may assume a frozen $^4$He core, with 2 protons and 2 neutrons in the $0s_{1/2}$ shell, and then only allow active particles in the $0p_{1/2}$ and $0p_{3/2}$ orbits.

Another example is a frozen $^{16}$O core, with 8 protons and 8 neutrons filling the $0s_{1/2}$, $0p_{1/2}$ and $0p_{3/2}$ orbits, with valence particles in the $0d_{5/2}$, $1s_{1/2}$ and $0d_{3/2}$ orbits.

Sometimes we refer to nuclei by the valence space where their last nucleons go. So, for example, we call $^{12}$C a $p$-shell nucleus, while $^{26}$Al is an $sd$-shell nucleus and $^{56}$Fe is a $pf$-shell nucleus.
Building a many-body basis

There are different kinds of truncations.

For example, one can start with ‘filled’ orbits (almost always the lowest), and then allow one, two, three... particles excited out of those filled orbits. These are called 1p-1h, 2p-2h, 3p-3h excitations.
Building a many-body basis

There are different kinds of truncations.

For example, one can start with ‘filled’ orbits (almost always the lowest), and then allow one, two, three... particles excited out of those filled orbits. These are called 1p-1h, 2p-2h, 3p-3h excitations.
Building a many-body basis

There are different kinds of truncations.

Alternately, one can state a maximal orbit and allow all possible configurations with particles occupying states up to that maximum. This is called *full configuration*.

```
------------  0f_{7/2}

--------------  0d_{3/2}

------------  1s_{1/2}  0d_{5/2}

-------------  0p_{3/2}

---------------  0p_{1/2}

-------------  0s_{1/2}
```
Building a many-body basis

There are different kinds of truncations.

Finally, for particular use in nuclear physics, there is the energy truncation, also called the $N\hbar\Omega$ or $N_{\text{max}}$ truncation.

Here one works in a harmonic oscillator basis, with each major oscillator shell assigned a principal quantum number $N = 0, 1, 2, 3, \ldots$

<table>
<thead>
<tr>
<th>Principal Quantum Number</th>
<th>Major Oscillator Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0s_{1/2}$</td>
</tr>
<tr>
<td>1</td>
<td>$0p_{1/2}$, $0p_{3/2}$</td>
</tr>
<tr>
<td>2</td>
<td>$0d_{5/2}$, $1s_{1/2}$</td>
</tr>
<tr>
<td>3</td>
<td>$0f_{7/2}$</td>
</tr>
</tbody>
</table>
Building a many-body basis

The $N\hbar\Omega$ or $N_{max}$ truncation:

Any configuration is given an noninteracting energy, which is the sum of the single-particle harmonic oscillator energies. (Thus this ignores spin-orbit splitting.)

![Diagram showing single-particle levels and occupation numbers for N=0, N=1, N=2, and N=3]
Building a many-body basis

The $N\hbar\Omega$ or $N_{\text{max}}$ truncation:

Excited states are labeled relative to the lowest configuration by the number of harmonic oscillator quanta.

A case of $N = 2$. 
Building a many-body basis

The $N\hbar\Omega$ or $N_{\text{max}}$ truncation:

Excited state are labeled relative to the lowest configuration by the number of harmonic oscillator quanta

Another case of $N = 2$.

This truncation is useful because: if one includes all configuration up to some $N_{\text{max}}$, and has a translationally invariant interaction, then the intrinsic motion and the center-of-mass motion factor. In other words, we can know exactly the center-of-mass wavefunction.
In almost all cases, the many-body Hamiltonian is rotationally invariant. This means it commutes with the operators $\hat{J}^2$, $\hat{J}_z$ and so eigenstates will have good $J, M$. Furthermore, the eigenenergies do not depend upon the orientation $M$.

Therefore we can choose to construct a many-body basis which has fixed $M$; this is called an $M$-scheme basis.

Alternately, one can construct a many-body basis which has fixed $J$, or a $J$-scheme basis.

The Hamiltonian matrix will have smaller dimensions (a factor of 10 or more) in the $J$-scheme than in the $M$-scheme. On the other hand, as we’ll show in the next slide, the $M$-scheme is very easy to construct with Slater determinants, while the $J$-scheme basis states, and thus the matrix elements, are more complicated, almost always being linear combinations of $M$-scheme states. $J$-scheme bases are important and useful, but we’ll focus on the simpler $M$-scheme.
Building a many-body basis

The quantum number $m$ is additive (because the underlying group is Abelian): if a Slater determinant $\hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k^\dagger \ldots |0\rangle$ is built from single-particle states all with good $m$, then the total

$$M = m_i + m_j + m_k + \ldots$$

This is not true of $J$, because the angular momentum group SU(2) is not Abelian.

The upshot is that

- It is easy to construct a Slater determinant with good total $M$;
- It is trivial to calculate $M$ for each Slater determinant;
- So it is easy to construct an $M$-scheme basis with fixed total $M$.

Note that the individual $M$-scheme basis states will not, in general, have good total $J$. Because the Hamiltonian is rotationally invariant, however, the eigenstates will have good $J$. (The situation is muddied when one has states of different $J$ that are nonetheless degenerate.)
Building a many-body basis

Example: two $j = 1/2$ orbits:

<table>
<thead>
<tr>
<th>Index</th>
<th>$n$</th>
<th>$l$</th>
<th>$j$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>-1/2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>-1/2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Note: the order is arbitrary

There are $\binom{4}{2} = 6$ two-particle states, which we list with the total $M$:

<table>
<thead>
<tr>
<th>Occupied</th>
<th>$M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2</td>
<td>0</td>
</tr>
<tr>
<td>1,3</td>
<td>-1</td>
</tr>
<tr>
<td>1,4</td>
<td>0</td>
</tr>
<tr>
<td>2,3</td>
<td>0</td>
</tr>
<tr>
<td>2,4</td>
<td>1</td>
</tr>
<tr>
<td>3,4</td>
<td>0</td>
</tr>
</tbody>
</table>

There are 4 states with $M = 0$, and 1 each with $M = \pm 1$. 
Building a many-body basis

Example: consider using only single particle states from the $0d_{5/2}$ space. They have the following quantum numbers

<table>
<thead>
<tr>
<th>Index</th>
<th>$n$</th>
<th>$l$</th>
<th>$j$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>-5/2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>-3/2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>-1/2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>1/2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>3/2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>5/2</td>
<td>5/2</td>
</tr>
</tbody>
</table>

There are $\binom{6}{2} = 15$ two-particle states, which we list with the total $M$:

<table>
<thead>
<tr>
<th>Occupied</th>
<th>$M$</th>
<th>Occupied</th>
<th>$M$</th>
<th>Occupied</th>
<th>$M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2</td>
<td>-4</td>
<td>2,3</td>
<td>-2</td>
<td>3,5</td>
<td>1</td>
</tr>
<tr>
<td>1,3</td>
<td>-3</td>
<td>2,4</td>
<td>-1</td>
<td>3,6</td>
<td>2</td>
</tr>
<tr>
<td>1,4</td>
<td>-2</td>
<td>2,5</td>
<td>0</td>
<td>4,5</td>
<td>2</td>
</tr>
<tr>
<td>1,5</td>
<td>-1</td>
<td>2,6</td>
<td>1</td>
<td>4,6</td>
<td>3</td>
</tr>
<tr>
<td>1,6</td>
<td>0</td>
<td>3,4</td>
<td>0</td>
<td>5,6</td>
<td>4</td>
</tr>
</tbody>
</table>

There are 3 states with $M = 0$, 2 with $M = 1$, and so on.
HOMEWORK!
The basic goal of these exercises are for you to build your own configuration-interaction shell-model code. The code will be fairly basic; it will assume one a single species of particles, e.g. only neutrons, and you will read in uncoupled two-body matrix elements. Furthermore the pieces of the code will not be the most efficient. Nonetheless it will be usable; most importantly, you will gain a good idea of what goes into a many-body shell-model code.
The goal of these exercise is to construct the $M$-scheme basis of Slater determinants in binary representation. Here $M$-scheme means the total $J_z$ of the many-body states is fixed.

The steps are:

- Read in a user-supplied file of single-particle states (we will provide examples);
- Ask for the total $M$ of the system and the number of particles $N$;
- Construct all the $N$-particle states with given $M$. You will validate the code by comparing both the number of states and specific states.
Homework Part 1: Thursday, 5 July 2012

We will supply you with sample files for single-particle states (ending in .spstates, e.g. sd.spstates). The format of the file is as follows

```
12 ! number of single-particle states
1  1  0  1  -1 ! index nrnodes l 2 x j
2  1  0  1  1
3  0  2  3  -3
4  0  2  3  -1
5  0  2  3  1
6  0  2  3  3
7  0  2  5  -5
8  0  2  5  -3
9  0  2  5  -1
10 0  2  5  1
11 0  2  5  3
12 0  2  5  5
```

This represents the $1s_{1/2}-0d_{3/2}-0d_{5/2}$ valence space, or the $sd$-space. There are twelve single-particle states, labeled by an overall index, and which have associated quantum numbers the number of radial nodes ($nrnodes$), the orbital angular momentum $l$, and the angular momentum $j$ and third component $j_z$. To keep everything a integer, we store $2 \times j$ and $2 \times j_z$. 
Homework Part 1: Thursday, 5 July 2012

To read in the single-particle states you need to:

- Open the file
- Read the number of single-particle states (in the above example, 12); allocate memory; all you need is a single array storing $2 \times j_z$ for each state, labeled by the index;
- Read in the quantum numbers and store $2 \times j_z$ (and anything else you happen to want).
Opening files in Fortran
Here is some simple code that allows you to open a named file in Fortran:

```fortran
character(40) :: filename
integer :: ilast

write(6,*) 'Enter name of input file (.dat)'
read(5,'(a)')filename
ilast = index(filename,' ')-1
open(unit=2,file=filename(1:ilast)//'.dat',status='old',err=1111)
return
1111 continue
write(6,*) filename(1:ilast),'.dat does not exist'
stop
```

If you want to open a new, not-yet-existing file, use `status='new'`; for any status use `status='unknown'` (but this will allow you to overwrite an existing file).
The next step is to read in the number of particles $N$ and the fixed total $M$ (or, actually, $2 \times M$). For this assignment we assume only a single species of particles, say neutrons, although this can be relaxed. Note: Although it is often a good idea to try to write a more general code, given the short time allotted we would suggest you keep your ambition in check, at least in the initial phases of the project.

You should probably write an error trap to make sure $N$ and $M$ are congruent; if $N$ is even, then $2 \times M$ should be even, and if $N$ is odd then $2 \times M$ should be odd.
The final step is to generate the set of $N$-particle Slater determinants with fixed $M$. The Slater determinants will be stored in occupation representation. Although in many codes this representation is done compactly in bit notation (1s and 0s), but for greater transparency and simplicity we will list the occupied single particle states. Hence we can store the Slater determinant basis states as

$$sd(i, j)$$

an array of dimension $N_{SD}$, the number of Slater determinants, by $N$, the number of occupied state. So if for the 7th Slater determinant the 2nd, 3rd, and 9th single-particle states are occupied, then $sd(7, 1) = 2$, $sd(7, 2) = 3$, and $sd(7, 3) = 9$. 
We can construct an occupation representation of Slater determinants by the *odometer* method. Consider $N_{sp} = 12$ and $N = 4$. Start with the first 4 states occupied, that is:

$$\text{sd}(1,:) = 1, 2, 3, 4$$
(also written as $|1, 2, 3, 4\rangle$)

Now increase the last occupancy recursively:

$$\text{sd}(2,:) = 1, 2, 3, 5$$
$$\text{sd}(3,:) = 1, 2, 3, 6$$
$$\text{sd}(4,:) = 1, 2, 3, 7$$
$$\ldots$$
$$\text{sd}(9,:) = 1, 2, 3, 12$$

Then start over with

$$\text{sd}(10,:) = 1, 2, 4, 5$$

and again increase the rightmost digit

$$\text{sd}(11,:) = 1, 2, 4, 6$$
$$\text{sd}(12,:) = 1, 2, 4, 7$$
$$\ldots$$
$$\text{sd}(17,:) = 1, 2, 4, 12$$
To help you, here is the algorithm in Fortran90. Here $N$ is the number of particles, $N_{sp}$ is the number of single-particle states, and $occ$ is an array of dimension $N$. The following routine takes $occ$ and moves it to the next ‘odometer’ reading:

```fortran
do j = N,1,-1
  if(occ(j) < Nsp-N+j )then
    l = occ(j)
    do k = j,N
      occ(k) = l+ 1+k-j ! sequential order
    end do
  end if
end do
occ(:) = 0 ! flag
```

You need to confirm that you get all possible states. You might try a smaller value that is easily confirmed by hand. If there are $N_{sp}$ single-particle states and $N$ particles, the maximal number of Slater determinants is $\binom{N_{sp}}{N}$. You should confirm you get this.
When we restrict ourselves to an $M$-scheme basis, however, there will be fewer states. You have two options. The first is simplest (and simplest is often best, at least in the first draft of a code): generate all possible Slater determinants, and then extract from this initial list a list of those Slater determinants with a given $M$. (You will need to write a short function or routine that computes $M$ for any given occupation.)

Alternately, and not too difficult, is to run the odometer routine twice: each time, as as a Slater determinant is calculated, compute $M$, but do not store the Slater determinants except the current one. You can then count up the number of Slater determinants with a chosen $M$. Then allocated storage for the Slater determinants, and run the odometer algorithm again, this time storing Slater determinants with the desired $M$ (this can be done with a simple logical flag).
Some example solutions: Let’s begin with a simple case, the 0d_{5/2} space:

<table>
<thead>
<tr>
<th>number of single-particle states</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ! number of single-particle states</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

For two particles, there are a total of 15 states, which I list here with the total $M$:

\[
| 1, 2\rangle, M = -4; | 1, 3\rangle, M = -3 \\
| 1, 4\rangle, M = -2; | 1, 5\rangle, M = -1 \\
| 1, 5\rangle, M = 0; | 2, 3\rangle, M = -2 \\
| 2, 4\rangle, M = -1; | 2, 5\rangle, M = 0 \\
| 2, 6\rangle, M = 1; | 3, 4\rangle, M = 0 \\
| 3, 5\rangle, M = 1; | 3, 6\rangle, M = 2 \\
| 4, 5\rangle, M = 2; | 4, 6\rangle, M = 3 \\
| 5, 6\rangle, M = 4
\]

Of these, there are only 3 states with $M = 0$. 
You should try by hand to show that in this same single-particle space, that for \( N = 3 \) there are 3 states with \( M = 1/2 \) and for \( N = 4 \) there are also only 3 states with \( M = 0 \). To test your code, confirm the above.

Also, for the \( sd \)-space given above, for \( N = 2 \) there are 14 states with \( M = 0 \), for \( N = 3 \) there are 37 states with \( M = 1/2 \), for \( N = 4 \) there are 81 states with \( M = 0 \).
Also, for much of the validation of your codes we will consider the pairing interaction. A simple space is the \((1/2)^2\) space

\[
\begin{array}{cccc}
4 & \text{number of single-particle states} \\
1 & 0 & 0 & 1 & -1 \\
2 & 0 & 0 & 1 & 1 \\
3 & 1 & 0 & 1 & -1 \\
4 & 1 & 0 & 1 & 1 \\
\end{array}
\]

For \(N = 2\) there are 4 states with \(M = 0\); show this by hand and confirm your code reproduces it.
Another, slightly more challenging space is the $\left(\frac{1}{2}\right)^4$ space, that is,

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

! number of single-particle states

For $N = 2$ there are 16 states with $M = 0$; for $N = 3$ there are 24 states with $M = 1/2$, and for $N = 4$ there are 36 states with $M = 0$. 
In the shell-model context we can interpret this as 4 $s_{1/2}$ levels, with $m = \pm 1/2$, we can also think of these as simple 4 pairs, $\pm k$, $k = 1, 2, 3, 4$. Later on we will assign single-particle energies, depending on the radial quantum number $n$, that is, $\epsilon_k = |k|\delta$ so that they are equally spaced:

\[ k = +/- 4 \]
\[ k = +/- 3 \]
\[ k = +/- 2 \]
\[ k = +/- 1 \]
For application in the pairing model we can go further and consider only states with no “broken pairs,” that is, if $+k$ is filled (or $m = +1/2$, so is $-k$ ($m = -1/2$). If you want, you can modify your code to accept only these, and obtain the following 6 states:

$|1, 2, 3, 4\rangle$,
$|1, 2, 5, 6\rangle$,
$|1, 2, 7, 8\rangle$,
$|3, 4, 5, 6\rangle$,
$|3, 4, 7, 8\rangle$,
$|5, 6, 7, 8\rangle$

We’ll be using this for an illustrative example.
Hints for coding

- Write small modules (routines/functions) ; avoid big subroutines that do everything. (But not too small.)
- Write lots of error traps, even for things that are ‘obvious.’
- Document as you go along. For each subroutine/function writer a header that includes: (A) Main purpose of routine; (B) names and brief explanation of input variables, if any; (C) names and brief explanation of output variables, if any; (D) subroutines and functions called by this routine; (E) called by which subroutines and functions
Hints for coding

- When debugging, print out intermediate values. It’s almost impossible to debug a code by looking at it—the code will almost always win a ‘staring contest.’

- Validate code with SIMPLE CASES. Validate early and often.

The number one mistake is using a too complex a system to test. *For example* If you are computing particles in a potential in a box, try removing the potential—you should get particles in a box. And start with one particle, then two, then three... Don’t start with eight particles.
Bit representation

Our recommended occupation representation, e.g. $|1, 2, 4, 8\rangle$, is easy to code, but numerically inefficient when one has hundreds of millions of Slater determinants.

In state-of-the-art shell-model codes, one generally uses *bit representation*, i.e. $|1101000100...\rangle$ where one stores the Slater determinant as a single (or a small number of) integer.

This is much more compact, but more intricate to code with considerable more overhead. There exist bit-manipulation functions (e.g. `iand`, `ishft`, etc., in Fortran).
Bit representation

Going from an occupation listing to a bit representation:

integer :: Np ! # of particles
integer :: occ(Np) ! occupation listing
integer :: imask ! a "mask"
integer :: bitSD ! bit representation of a Slater Determinant
integer :: prevocc ! the previous occupation
integer :: nshift ! number of positions to shift

bitSD = 0 ! initialize

imask = 1 ! putting a bit in the first location
prevocc = 1
do i = 1,Np
   nshift = occ(Np) - prevocc
   imask = ishft(imask,nshift) ! ishft shifts a binary word ! by nshift bits
   bitSD = bitSD+ imask
end do ! loop over i
Step by step...

Converting the occupation 1, 2, 4, 8 to a bit:
Initially:
bitSD = 0
Occupation of first particle is 1, imask = 1
bitSD = 1

Occupation of second particle is 2, imask = 2
bitSD = 1+2 = 3;

Occupation of third particle is 4, imask = 8
bitSD = 3+8 = 11;

Occupation of fourth particle is 8, imask = 128
bitSD = 11+128 = 139

Hence the occupation 1, 2, 4, 8 is written as a single integer 139.
Bit representation

Given a bit representation of a Slater determinant, to determine if a single-particle state is occupied, use `imask` and the `iand` (in Fortran) function.

One can then create and destroy bits.

While more compact and thus reduces the memory requirement, you can see this procedure adds significant computing overhead.
Example case: pairing Hamiltonian

Bonus round!

In preparation for Monday’s lectures on computing the Hamiltonian matrix elements, and as an example to be used in Friday’s lectures on diagonalization,

let’s consider an analytic case, the pairing Hamiltonian.
Example case: pairing Hamiltonian

We consider a space with $2\Omega$ single-particle states, with each state labeled by $k = 1, 2, 3, \Omega$ and $m = \pm 1/2$. The convention is that the state with $k > 0$ has $m = +1/2$ while $-k$ has $m = -1/2$.

The Hamiltonian we consider is

$$\hat{H} = -G\hat{P}_+\hat{P}_-,$$

where

$$\hat{P}_+ = \sum_{k>0} \hat{a}^\dagger_k \hat{a}^\dagger_{-k},$$

and $\hat{P}_- = (\hat{P}_+)^\dagger$.

We will first solve this using a trick, the quasi-spin formalism, to obtain the exact results. Then we will try again using the explicit Slater determinant formalism.
Example case: pairing Hamiltonian

One can show (and this is a good exercise!) that

$$\left[ \hat{P}_+, \hat{P}_- \right] = \sum_{k>0} \left( \hat{a}_k^\dagger \hat{a}_k + \hat{a}_{-k}^\dagger \hat{a}_{-k} - 1 \right) = \hat{N} - \Omega.$$

Now define

$$\hat{P}_z = \frac{1}{2}(\hat{N} - \Omega).$$

Finally you can show

$$\left[ \hat{P}_z, \hat{P}_\pm \right] = \pm \hat{P}_\pm.$$

This means the operators $\hat{P}_\pm, \hat{P}_z$ form an SU(2) algebra, and we can bring to bear all our intuition about angular momentum, even though there is no actual angular momentum involved.

So we rewrite the Hamiltonian to make this explicit:

$$\hat{H} = -G\hat{P}_+ \hat{P}_- = -G \left( \hat{P}^2 - \hat{P}_z^2 + \hat{P}_z \right).$$
Example case: pairing Hamiltonian

Because of the SU(2) algebra, we know that the eigenvalues of $\hat{P}^2$ must be of the form $p(p + 1)$, with $p$ either integer or half-integer, and the eigenvalues of $\hat{P}_z$ are $m_p$ with $p \geq |m_p|$, with $m_p$ also integer or half-integer.

But because $\hat{P}_z = (1/2)(\hat{N} - \Omega)$, we know that for $N$ particles the value $m_p = (N - \Omega)/2$. Furthermore, the values of $m_p$ range from $-\Omega/2$ (for $N = 0$) to $+\Omega/2$ (for $N = 2\Omega$, with all states filled).

We deduce the maximal $p = \Omega/2$ and for a given $n$ the values range of $p$ range from $|N - \Omega|/2$ to $\Omega/2$ in steps of 1 (for an even number of particles)

Following Racah we introduce the notation $p = (\Omega - v)/2$ where $v = 0, 2, 4, ..., \Omega - |N - \Omega|$ With this it is easy to deduce that the eigenvalues of the pairing Hamiltonian are

$$-G(N - v)(2\Omega + 2 - N - v)/4$$

This also works for $N$ odd, with $v = 1, 3, 5, \ldots$. 
Example case: pairing Hamiltonian

Let’s take a specific example: $\Omega = 3$ so there are 6 single-particle states, and $N = 3$, with $\nu = 1, 3$. Therefore there are two distinct eigenvalues,

$$E = -2G, 0$$

Now let’s work this out explicitly. The single particle space we write as

<table>
<thead>
<tr>
<th>Index</th>
<th>$k$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1/2</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-1/2</td>
</tr>
<tr>
<td>4</td>
<td>-2</td>
<td>1/2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>-1/2</td>
</tr>
<tr>
<td>6</td>
<td>-3</td>
<td>1/2</td>
</tr>
</tbody>
</table>

There are $\binom{6}{3} = 20$ three-particle states, but there are 9 states with $M = +1/2$: $|1, 2, 3\rangle, |1, 2, 5\rangle, |1, 4, 6\rangle, |2, 3, 4\rangle, |2, 3, 6\rangle, |2, 4, 5\rangle, |2, 5, 6\rangle, |3, 4, 6\rangle, |4, 5, 6\rangle$. 
Example case: pairing Hamiltonian

In this basis, the operator
\[ \hat{\mathcal{P}}_+ = \hat{a}^\dagger_1 \hat{a}^\dagger_2 + \hat{a}^\dagger_3 \hat{a}^\dagger_4 + \hat{a}^\dagger_5 \hat{a}^\dagger_6 \]

From this we can determine that
\[ \hat{\mathcal{P}}_- |1, 4, 6\rangle = \hat{\mathcal{P}}_- |2, 3, 6\rangle = \hat{\mathcal{P}}_- |2, 4, 5\rangle = 0 \]

so those states all have eigenvalue 0.
Example case: pairing Hamiltonian

Now for further example,

\[ \hat{P}_- |1, 2, 3\rangle = |3\rangle \]

so

\[ \hat{P}_+ \hat{P}_- |1, 2, 3\rangle = |1, 2, 3\rangle + |3, 4, 3\rangle + |5, 6, 3\rangle \]

The second term vanishes because state 3 is occupied twice, and reordering the last term we get

\[ \hat{P}_+ \hat{P}_- |1, 2, 3\rangle = |1, 2, 3\rangle + |3, 5, 6\rangle \]

without picking up a phase.
Example case: pairing Hamiltonian

Continuing in this fashion, with the previous ordering of the many-body states \(|1, 2, 3\rangle, |1, 2, 5\rangle, |1, 4, 6\rangle, |2, 3, 4\rangle, |2, 3, 6\rangle, |2, 4, 5\rangle, |2, 5, 6\rangle, |3, 4, 6\rangle, |4, 5, 6\rangle\) the Hamiltonian matrix of this system is

\[
H = -G = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

This will be a useful check on your work next week. One can by hand confirm that there are 3 eigenvalues \(-2G\) and 6 with value zero.
Example case: pairing Hamiltonian

Another example
Using the \((1/2)^4\) single-particle space

\[
\begin{array}{cccc}
8 & \text{! number of single-particle states} \\
1 & 0 & 0 & 1 & -1 \\
2 & 0 & 0 & 1 & 1 \\
3 & 1 & 0 & 1 & -1 \\
4 & 1 & 0 & 1 & 1 \\
5 & 2 & 0 & 1 & -1 \\
6 & 2 & 0 & 1 & 1 \\
7 & 3 & 0 & 1 & -1 \\
8 & 3 & 0 & 1 & 1 \\
\end{array}
\]

and then taking only 4-particle, \(M = 0\) states that have no ‘broken pairs’, there are six basis Slater determinants:

\[
\begin{align*}
|1, 2, 3, 4\rangle, \\
|1, 2, 5, 6\rangle, \\
|1, 2, 7, 8\rangle, \\
|3, 4, 5, 6\rangle, \\
|3, 4, 7, 8\rangle, \\
|5, 6, 7, 8\rangle \\
\end{align*}
\]
Example case: pairing Hamiltonian

Now we take the following Hamiltonian:

\[ \hat{H} = \sum_n n\delta \hat{N}_n - G \hat{P}^\dagger \hat{P} \]

where

\[ \hat{N}_n = \hat{a}^\dagger_{n,m=+1/2} \hat{a}_{n,m=+1/2} + \hat{a}^\dagger_{n,m=-1/2} \hat{a}_{n,m=-1/2} \]

and

\[ \hat{P}^\dagger = \sum_n \hat{a}^\dagger_{n,m=+1/2} \hat{a}^\dagger_{n,m=-1/2} \]

We can write down the $6 \times 6$ Hamiltonian in the basis from the prior slide:

\[
H = \begin{pmatrix}
2\delta - G & -G & -G & -G & -G & 0 \\
-G & 4\delta - G & -G & -G & -G & -G \\
-G & -G & 6\delta - G & 0 & -G & -G \\
-G & -G & 0 & 6\delta - G & -G & -G \\
-G & -G & -G & -G & 8\delta - G & -G \\
0 & -G & -G & -G & -G & 10\delta - G
\end{pmatrix}
\]

(You should check by hand that this is correct.) For $\delta = 0$ we can determine algebraically the g.s. energy is $-6G$. Tomorrow (Friday) you will be asked to solve this numerically for $\delta \neq 0$. 