Perturbation Theory Formalism

This approach tries to bridge the approach presented by J. J. Sakurai and which you find on the web with the more standard approach we'd see on, e.g., Wikipedia. The idea is to convert things into matrix operations so as to make the logic of the problem more obvious.

$$H = H^{(0)} + W$$

Derivation

We'll write our Hamiltonian out as

$$H = H^{(0)} + \lambda H^{(1)} + \lambda^2 H^{(2)} + \ldots$$

and we'll write our states out as

$$|n\rangle = |n\rangle^{(0)} + \lambda |n\rangle^{(1)} + \lambda^2 |n\rangle^{(2)} + \ldots$$

where

$$H^{(0)} |n\rangle^{(0)} = E_n^{(0)} |n\rangle^{(0)}$$

Now we want to solve

$$H |n\rangle = E |n\rangle$$

which expands out as

$$\left(H^{(0)} + \lambda H^{(1)} + \lambda^2 H^{(2)} + \ldots\right)\left(|n\rangle^{(0)} + \lambda |n\rangle^{(1)} + \lambda^2 |n\rangle^{(2)} + \ldots\right) =$$
$$\left(E_n^{(0)} + \lambda E_n^{(1)} + \lambda^2 E_n^{(2)} + \ldots\right)\left(|n\rangle^{(0)} + \lambda |n\rangle^{(1)} + \lambda^2 |n\rangle^{(2)} + \ldots\right)$$

At this point we've got to do some combinatoric nonsense, by collecting terms that have the same order in λ, which gives us

$$H^{(0)} |n^{(0)}\rangle = E_n^{(0)} |n^{(0)}\rangle$$
$$H^{(0)} |n^{(1)}\rangle + H^{(1)} |n^{(0)}\rangle = E_n^{(0)} |n^{(1)}\rangle + E_n^{(1)} |n^{(0)}\rangle$$
$$H^{(0)} |n^{(2)}\rangle + H^{(1)} |n^{(1)}\rangle + H^{(2)} |n^{(0)}\rangle = E_n^{(0)} |n^{(2)}\rangle + E_n^{(1)} |n^{(1)}\rangle + E_n^{(2)} |n^{(0)}\rangle$$

then we can expand our corrections as expansion vectors, like

$$|n^{(1)}\rangle = \sum_\alpha c_\alpha^{(1)} |\alpha^{(0)}\rangle$$
$$|n^{(2)}\rangle = \sum_\alpha c_\alpha^{(2)} |\alpha^{(0)}\rangle$$

and then we project onto $|n^{(0)}\rangle$, giving us

$$\langle n^{(0)}| H^{(0)} |n^{(1)}\rangle + \langle n^{(0)}| H^{(1)} |n^{(0)}\rangle = \langle n^{(0)}| E_n^{(0)} |n^{(1)}\rangle + \langle n^{(0)}| E_n^{(1)} |n^{(0)}\rangle$$
$$\langle n^{(0)}| H^{(0)} |n^{(2)}\rangle + \langle n^{(0)}| H^{(1)} |n^{(1)}\rangle + \langle n^{(0)}| H^{(2)} |n^{(0)}\rangle = \langle n^{(0)}| E_n^{(0)} |n^{(2)}\rangle + \langle n^{(0)}| E_n^{(1)} |n^{(1)}\rangle + \langle n^{(0)}| E_n^{(2)} |n^{(0)}\rangle$$

which obviously looks like garbage, but don't worry, we're going somewhere.

Generalization & Higher-Order Corrections

Rather than work out every order of correction on by one, we'll reorganize things to give a cleaner formulation. To start, we'll rewrite our Trash Equations in a more general form as

$$\sum_{i=0}^{k} H^{(k-i)} \left|n^{(i)}\right\rangle = \sum_{i=0}^{k} {E_n}^{(k-i)} \left|n^{(i)}\right\rangle$$

we can then operate again on both sides with $\left\langle n^{(0)}\right|$ giving us

$$\begin{aligned}\sum_{i=0}^{k} \left\langle n^{(0)}\right| H^{(k-i)} \left|n^{(i)}\right\rangle &= \sum_{i=0}^{k} \left\langle n^{(0)}\right| {E_n}^{(k-i)} \left|n^{(i)}\right\rangle \\ &= \sum_{i=0}^{k} {E_n}^{(k-i)} \left\langle n^{(0)} \middle| n^{(i)}\right\rangle\end{aligned}$$

and in particular this means

$${E_n}^{(k)} = \left\langle n^{(0)}\right| H^{(k)} \left|n^{(0)}\right\rangle + \sum_{i=1}^{k} \left\langle n^{(0)}\right| H^{(k-i)} \left|n^{(i)}\right\rangle - {E_n}^{(k-i)} \left\langle n^{(0)} \middle| n^{(i)}\right\rangle$$

and noting that the k-term in that sum is

$$\begin{aligned}\left\langle n^{(0)}\right| H^{(0)} \left|n^{(k)}\right\rangle - {E_n}^{(0)} \left\langle n^{(0)} \middle| n^{(k)}\right\rangle &= {E_n}^{(0)} \left\langle n^{(0)} \middle| n^{(k)}\right\rangle - {E_n}^{(0)} \left\langle n^{(0)} \middle| n^{(k)}\right\rangle \\ &= 0\end{aligned}$$

we get

$${E_n}^{(k)} = \left\langle n^{(0)}\right| H^{(k)} \left|n^{(0)}\right\rangle + \sum_{i=1}^{k-1} \left\langle n^{(0)}\right| H^{(k-i)} \left|n^{(i)}\right\rangle - {E_n}^{(k-i)} \left\langle n^{(0)} \middle| n^{(i)}\right\rangle$$

next, we need an extra condition to get unique solutions, so we choose to preserve orthonormality at all orders of λ, giving us

$$\begin{aligned}\left\langle n \middle| m\right\rangle^{(k)} &= \sum_{i=0}^{k} \sum_{j=0}^{k-i} \left\langle n^{(i)} \middle| m^{(j)}\right\rangle \\ &= \left\langle n \middle| m\right\rangle^{(k-1)} + \sum_{i=0}^{k} \left\langle n^{(i)} \middle| m^{(k-i)}\right\rangle \\ &= \delta_{nm}\end{aligned}$$

then as

$$\left\langle n \middle| m\right\rangle^{(0)} = \left\langle n^{(0)} \middle| m^{(0)}\right\rangle = \delta_{nm}$$

we get

$$\sum_{i=0}^{k} \left\langle n^{(i)} \middle| m^{(k-i)}\right\rangle = 0$$

finally, we'll note that in general

$$\left(H^{(0)}-E_n^{(0)}\right)\left|n^{(k)}\right\rangle=\sum_{i=0}^{k-1}\left(E_n^{(k-i)}-H^{(k-i)}\right)\left|n^{(i)}\right\rangle$$

and so if we can $\left|n^{(0)}\right\rangle \ldots \left|n^{(k-1)}\right\rangle$ and $E_n^{(k)}$, we should be able to get $\left|n^{(k)}\right\rangle$

First-Order Correction Revisited

We'll redo our first order correction in this context. First, this gives us

$$E_n^{(1)}=\left\langle n^{(0)}\right| H^{(1)}\left|n^{(0)}\right\rangle$$

then we have

$$\left\langle n^{(1)}\middle|m^{(0)}\right\rangle+\left\langle n^{(0)}\middle|m^{(1)}\right\rangle=0$$

which tells us that

$$\begin{aligned}\left\langle n^{(1)}\middle|n^{(0)}\right\rangle&=-\left\langle n^{(0)}\middle|n^{(1)}\right\rangle=-\left\langle n^{(0)}\middle|n^{(1)}\right\rangle^*\\ \therefore\left\langle n^{(1)}\middle|n^{(0)}\right\rangle&=\left\langle n^{(0)}\middle|n^{(1)}\right\rangle=0\end{aligned}$$

Next, we want to get an expression for $\left|n^{(1)}\right\rangle$ itself, so we expand it out in our initial basis

$$\left|n^{(1)}\right\rangle=\sum_m c_m^{(1)}\left|m^{(0)}\right\rangle$$

where we know that $c_n^{(1)}=0$

Then returning to our original equation, we have

$$\begin{aligned}\left(H^{(0)}-E_n^{(0)}\right)\sum_m c_m^{(1)}\left|m^{(0)}\right\rangle&=\sum_m c_m^{(1)}\left(H^{(0)}-E_n^{(0)}\right)\left|m^{(0)}\right\rangle\\ &=\sum_m c_m^{(1)}\left(H^{(0)}\left|m^{(0)}\right\rangle-E_n^{(0)}\left|m^{(0)}\right\rangle\right)\\ &=\sum_m c_m^{(1)}\left(E_m^{(0)}-E_n^{(0)}\right)\left|m^{(0)}\right\rangle\end{aligned}$$

$$\begin{aligned}\left(E_n^{(1)}-H^{(1)}\right)\left|n^{(0)}\right\rangle+\sum_{i=1}^{k-1}\left(E_n^{(k-i)}-H^{(k-i)}\right)\left|n^{(i)}\right\rangle&=\left(E_n^{(1)}-H^{(1)}\right)\left|n^{(0)}\right\rangle+\sum_{i=1}^{0}\left(E_n^{(k-i)}-H^{(k-i)}\right)\left|n^{(i)}\right\rangle\\ &=\left(E_n^{(1)}-H^{(1)}\right)\left|n^{(0)}\right\rangle\\ \sum_m c_m^{(1)}\left(E_m^{(0)}-E_n^{(0)}\right)\left|m^{(0)}\right\rangle&=\left(E_n^{(1)}-H^{(1)}\right)\left|n^{(0)}\right\rangle\end{aligned}$$

and so assuming $m\neq n$, by projecting with $\left\langle m^{(0)}\right|$ we find that

$$\begin{aligned}c_m^{(1)}&=-\frac{\left\langle m^{(0)}\right| H^{(1)}\left|n^{(0)}\right\rangle}{E_m^{(0)}-E_n^{(0)}}\\ &=\frac{\left\langle m^{(0)}\right| H^{(1)}\left|n^{(0)}\right\rangle}{E_n^{(0)}-E_m^{(0)}}\end{aligned}$$

The Perturbation Operator

We will find it convenient to define two operators so that we can do less term-manipulation, first we'd like to introduce an operator that projects out the n^{th} state from our terms

$$R_n=\left(I-\left|n^{(0)}\right\rangle\left\langle n^{(0)}\right|\right)=\sum_{\alpha\neq n}\left|m^{(0)}\right\rangle\left\langle m^{(0)}\right|$$

with this defined, we can then directly construct our coefficient vectors by inverting $H^{(0)}$ and defining our "perturbation operator"

$$\Pi_n=\left(E_n^{(0)}-H^{(0)}\right)^{-1}R_n$$

that can be directly applied to $H^{(1)}$ to give

$$\left|n^{(1)}\right\rangle=\Pi_n H^{(1)}\left|n^{(0)}\right\rangle$$

Second-Order Correction

We'll return to our motivating equations

$$E_n{}^{(k)}=\left\langle n^{(0)}\right|H^{(k)}\left|n^{(0)}\right\rangle+\sum_{i=1}^{k}\left\langle n^{(0)}\right|H^{(k-i)}\left|n^{(i)}\right\rangle-E_n{}^{(k-i)}\left\langle n^{(0)}\middle|n^{(i)}\right\rangle$$

$$\sum_{i=0}^{k}\left\langle n^{(i)}\middle|m^{(k-i)}\right\rangle=0$$

$$\left(H^{(0)}-E_n^{(0)}\right)\left|n^{(k)}\right\rangle=\left(E_n{}^{(k)}-H^{(k)}\right)\left|n^{(0)}\right\rangle+\sum_{i=1}^{k-1}\left(E_n{}^{(k-i)}-H^{(k-i)}\right)\left|n^{(i)}\right\rangle$$

and then letting $k=2$ we have

$$E_n{}^{(2)}=\left\langle n^{(0)}\right|H^{(2)}\left|n^{(0)}\right\rangle+\sum_{i=1}^{2}\left\langle n^{(0)}\right|H^{(2-i)}\left|n^{(i)}\right\rangle-E_n{}^{(2-i)}\left\langle n^{(0)}\middle|n^{(i)}\right\rangle$$

$$\sum_{i=0}^{2}\left\langle n^{(i)}\middle|m^{(2-i)}\right\rangle=0$$

$$\left(H^{(0)}-E_n^{(0)}\right)\left|n^{(2)}\right\rangle=\left(E_n{}^{(2)}-H^{(2)}\right)\left|n^{(0)}\right\rangle+\sum_{i=1}^{1}\left(E_n{}^{(2-i)}-H^{(2-i)}\right)\left|n^{(i)}\right\rangle$$

taking these one-by-one we have

$$\begin{aligned}E_n{}^{(2)}&=\left\langle n^{(0)}\right|H^{(2)}\left|n^{(0)}\right\rangle+\left\langle n^{(0)}\right|H^{(1)}\left|n^{(1)}\right\rangle-E_n{}^{(1)}\left\langle n^{(0)}\middle|n^{(1)}\right\rangle+\\&\qquad\left\langle n^{(0)}\right|H^{(0)}\left|n^{(2)}\right\rangle-E_n{}^{(0)}\left\langle n^{(0)}\middle|n^{(2)}\right\rangle\\&=\left\langle n^{(0)}\right|H^{(2)}\left|n^{(0)}\right\rangle+\left\langle n^{(0)}\right|H^{(1)}\left|n^{(1)}\right\rangle-E_n{}^{(1)}\left\langle n^{(0)}\middle|n^{(1)}\right\rangle+\\&\qquad E_n{}^{(0)}\left\langle n^{(0)}\middle|n^{(2)}\right\rangle-E_n{}^{(0)}\left\langle n^{(0)}\middle|n^{(2)}\right\rangle\\&=\left\langle n^{(0)}\right|H^{(2)}\left|n^{(0)}\right\rangle+\left\langle n^{(0)}\right|H^{(1)}\left|n^{(1)}\right\rangle\end{aligned}$$

$$= \langle n^{(0)} | \; H^{(2)} \; | n^{(0)} \rangle + \; \langle n^{(0)} | \; H^{(1)} \, \Pi_n \, H^{(1)} \; | n^{(0)} \rangle$$

$$\langle n^{(0)} \, | \, m^{(2)} \rangle + \langle n^{(1)} \, | \, m^{(1)} \rangle + \langle n^{(2)} \, | \, m^{(0)} \rangle = 0$$

$$\langle n^{(0)} \, | \, n^{(2)} \rangle = -\frac{1}{2} \langle n^{(1)} \, | \, n^{(1)} \rangle$$

$$\left(H^{(0)} - E_n^{(0)} \right) | n^{(2)} \rangle = \left(E_n^{(2)} - H^{(2)} \right) | n^{(0)} \rangle + \left(E_n^{(1)} - H^{(1)} \right) | n^{(1)} \rangle$$

$$= \left(E_n^{(2)} - H^{(2)} \right) | n^{(0)} \rangle + \left(E_n^{(1)} - H^{(1)} \right) \Pi_n \, H^{(1)} \; | n^{(0)} \rangle$$

then we'll note that the final equation gives us

$$R_n \, | n^{(2)} \rangle = \Pi_n \left(H^{(2)} + H^{(1)} \, \Pi_n \, H^{(1)} - E_n^{(2)} - E_n^{(1)} \, \Pi_n \, H^{(1)} \right) | n^{(0)} \rangle$$

$$= \left(\Pi_n \, H^{(2)} + \left(\Pi_n \, H^{(1)} \right)^2 - E_n^{(1)} (\Pi_n)^2 \, H^{(1)} \right) | n^{(0)} \rangle$$

and then adding on the projection in the direction of $| n^{(0)} \rangle$ we get

$$| n^{(2)} \rangle = R_n \, | n^{(2)} \rangle + | n^{(0)} \rangle \, \langle n^{(0)} \, | \, n^{(2)} \rangle$$

and so in total we have

$$| n^{(2)} \rangle = \left(\Pi_n \, H^{(2)} + \left(\Pi_n \, H^{(1)} \right)^2 - E_n^{(1)} (\Pi_n)^2 \, H^{(1)} - \frac{1}{2} \langle n^{(1)} \, | \, n^{(1)} \rangle \right) | n^{(0)} \rangle$$

Higher-Order Corrections

We'll take a moment to note that we didn't need to totally expand the second-order correction. And in the name of building a general form for these corrections, it's worth thinking about how this would work iteratively.

So let's go back to our equations

$$E_n^{(k)} = \langle n^{(0)} | \; H^{(k)} \; | n^{(0)} \rangle + \; \sum_{i=1}^{k-1} \langle n^{(0)} | \; H^{(k-i)} \; | n^{(i)} \rangle - E_n^{(k-i)} \, \langle n^{(0)} \, | \, n^{(i)} \rangle$$

$$\sum_{i=0}^{k} \langle n^{(i)} \, | \, m^{(k-i)} \rangle = 0$$

$$\left(H^{(0)} - E_n^{(0)} \right) | n^{(k)} \rangle = \sum_{i=0}^{k-1} \left(E_n^{(k-i)} - H^{(k-i)} \right) | n^{(i)} \rangle$$

and now let's assume that we've gotten all of the corrections up to order $k-1$. That means we have $E_n^{(k)}$ directly. Now we'll turn to the second term and note that it gives us

$$\langle n^{(0)} \, | \, m^{(k)} \rangle + \langle n^{(k)} \, | \, m^{(0)} \rangle = -\sum_{i=1}^{k-1} \langle n^{(i)} \, | \, m^{(k-i)} \rangle$$

and when $m=n$ we have

$$\langle n^{(0)} \, | \, n^{(k)} \rangle = -\frac{1}{2} \sum_{i=1}^{k-1} \langle n^{(i)} \, | \, n^{(k-i)} \rangle$$

finally following our previous prescription we have

$$\left|n^{(k)}\right\rangle = R_n \left|n^{(k)}\right\rangle + \left|n^{(0)}\right\rangle \left\langle n^{(0)} \middle| n^{(k)}\right\rangle$$

$$R_n \left|n^{(k)}\right\rangle = \sum_{i=0}^{k-1} \Pi_n \left(E_n{}^{(k-i)} - H^{(k-i)}\right) \left|n^{(i)}\right\rangle$$

$$\therefore \left|n^{(k)}\right\rangle = \sum_{i=0}^{k-1} \Pi_n \left(E_n{}^{(k-i)} - H^{(k-i)}\right) \left|n^{(i)}\right\rangle + \left|n^{(0)}\right\rangle \left\langle n^{(0)} \middle| n^{(k)}\right\rangle$$

Expressions

Here are the relevant equations for any level of approximation

$$E_n{}^{(k)} = \left\langle n^{(0)}\right| H^{(k)} \left|n^{(0)}\right\rangle + \sum_{i=1}^{k-1} \left\langle n^{(0)}\right| H^{(k-i)} \left|n^{(i)}\right\rangle - E_n{}^{(k-i)} \left\langle n^{(0)} \middle| n^{(i)}\right\rangle$$

$$\left\langle n^{(0)} \middle| n^{(k)}\right\rangle = -\frac{1}{2} \sum_{i=1}^{k-1} \left\langle n^{(i)} \middle| n^{(k-i)}\right\rangle$$

$$\left|n^{(k)}\right\rangle = \sum_{i=0}^{k-1} \Pi_n \left(E_n{}^{(k-i)} - H^{(k-i)}\right) \left|n^{(i)}\right\rangle + \left|n^{(0)}\right\rangle \left\langle n^{(0)} \middle| n^{(k)}\right\rangle$$

Table of Evaluated Terms

Annoyingly, this arrangement of the terms isn't very popular. Instead scientists usually write out direct expanded expressions in terms of the zero-order states. So we're gonna tabulate first, second, and third order correction expressions to make working with existing stuff cleaner. We won't do full expansions of the terms, because that's a pain in the ass, but we'll get part of the way there.

First Order

$$E_n{}^{(1)} = \left\langle n^{(0)}\right| H^{(1)} \left|n^{(0)}\right\rangle$$

$$\left\langle n^{(0)} \middle| n^{(1)}\right\rangle = 0$$

$$\left|n^{(1)}\right\rangle = \Pi_n \left(H^{(1)} - E_n{}^{(1)}\right) \left|n^{(0)}\right\rangle$$

$$= \Pi_n H^{(1)} \left|n^{(0)}\right\rangle$$

Second Order

$$E_n{}^{(2)} = \left\langle n^{(0)}\right| H^{(2)} \left|n^{(0)}\right\rangle + \left\langle n^{(0)}\right| H^{(1)} \left|n^{(1)}\right\rangle - E_n{}^{(1)} \left\langle n^{(0)} \middle| n^{(1)}\right\rangle$$

$$\left\langle n^{(0)} \middle| n^{(2)}\right\rangle = -\frac{1}{2} \left\langle n^{(1)} \middle| n^{(1)}\right\rangle$$

$$\left|n^{(2)}\right\rangle = \Pi_n \left(H^{(2)} - E_n{}^{(2)}\right) \left|n^{(0)}\right\rangle + \Pi_n \left(H^{(1)} - E_n{}^{(1)}\right) \left|n^{(1)}\right\rangle + \left\langle n^{(0)} \middle| n^{(2)}\right\rangle \left|n^{(0)}\right\rangle$$

$$= \Pi_n H^{(2)} \left|n^{(0)}\right\rangle + \Pi_n \left(H^{(1)} - E_n{}^{(1)}\right) \Pi_n H^{(1)} \left|n^{(0)}\right\rangle - \frac{1}{2} \left\langle n^{(1)} \middle| n^{(1)}\right\rangle \left|n^{(0)}\right\rangle$$

$$= \Pi_n H^{(2)} \left|n^{(0)}\right\rangle + \Pi_n H^{(1)} \Pi_n H^{(1)} \left|n^{(0)}\right\rangle - E_n^{(1)} \Pi_n \Pi_n H^{(1)} \left|n^{(0)}\right\rangle - \frac{1}{2} \left\langle n^{(1)} \middle| n^{(1)} \right\rangle \left|n^{(0)}\right\rangle$$

Third Order

$$\begin{aligned}
E_n^{(3)} &= \left\langle n^{(0)}\right| H^{(3)} \left|n^{(0)}\right\rangle + \sum_{i=1}^{2} \left\langle n^{(0)}\right| H^{(3-i)} \left|n^{(i)}\right\rangle - E_n^{(3-i)} \left\langle n^{(0)} \middle| n^{(i)} \right\rangle \\
&= \left\langle n^{(0)}\right| H^{(3)} \left|n^{(0)}\right\rangle + \left\langle n^{(0)}\right| H^{(2)} \left|n^{(1)}\right\rangle - E_n^{(2)} \left\langle n^{(0)} \middle| n^{(1)} \right\rangle + \left\langle n^{(0)}\right| H^{(1)} \left|n^{(2)}\right\rangle - E_n^{(1)} \left\langle n^{(0)} \middle| n^{(2)} \right\rangle \\
\left\langle n^{(0)} \middle| n^{(3)} \right\rangle &= -\frac{1}{2} \sum_{i=1}^{2} \left\langle n^{(i)} \middle| n^{(3-i)} \right\rangle \\
&= -\frac{1}{2} \left(\left\langle n^{(1)} \middle| n^{(2)} \right\rangle + \left\langle n^{(2)} \middle| n^{(1)} \right\rangle \right) \\
&= -\left\langle n^{(1)} \middle| n^{(2)} \right\rangle \\
\left|n^{(3)}\right\rangle &= \sum_{i=0}^{2} \Pi_n \left(H^{(3-i)} - E_n^{(3-i)} \right) \left|n^{(i)}\right\rangle + \left|n^{(0)}\right\rangle \left\langle n^{(0)} \middle| n^{(3)} \right\rangle \\
&= \Pi_n \left(H^{(3)} - E_n^{(3)} \right) \left|n^{(0)}\right\rangle + \Pi_n \left(H^{(2)} - E_n^{(2)} \right) \left|n^{(1)}\right\rangle + \Pi_n \left(H^{(1)} - E_n^{(1)} \right) \left|n^{(2)}\right\rangle + \left|n^{(0)}\right\rangle \left\langle n^{(0)} \middle| n^{(3)} \right\rangle \\
&= \Pi_n H^{(3)} \left|n^{(0)}\right\rangle + \Pi_n H^{(2)} \left|n^{(1)}\right\rangle + \Pi_n H^{(1)} \left|n^{(2)}\right\rangle - E_n^{(2)} \Pi_n \left|n^{(1)}\right\rangle - \left\langle n^{(1)} \middle| n^{(2)} \right\rangle \left|n^{(0)}\right\rangle
\end{aligned}$$

Fourth Order Energy

$$\begin{aligned}
{}_n^{(k)} &= \left\langle n^{(0)}\right| H^{(4)} \left|n^{(0)}\right\rangle + \left\langle n^{(0)}\right| H^{(3)} \left|n^{(1)}\right\rangle + \left\langle n^{(0)}\right| H^{(2)} \left|n^{(2)}\right\rangle + \left\langle n^{(0)}\right| H^{(1)} \left|n^{(3)}\right\rangle - E_n^{(2)} \left\langle n^{(0)} \middle| n^{(2)} \right\rangle - E_n^{(1)} \left\langle n^{(0)} \middle| n^{(3} \right. \\
{}^{(2)}\big\rangle &= \Pi_n H^{(2)} \left|n^{(0)}\right\rangle + \Pi_n H^{(1)} \Pi_n H^{(1)} \left|n^{(0)}\right\rangle - E_n^{(1)} \Pi_n \Pi_n H^{(1)} \left|n^{(0)}\right\rangle - \frac{1}{2} \left\langle n^{(1)} \middle| n^{(1)} \right\rangle \left|n^{(0)}\right\rangle \\
{}^{(3)}\big\rangle &= \Pi_n H^{(3)} \left|n^{(0)}\right\rangle + \Pi_n H^{(2)} \left|n^{(1)}\right\rangle + \Pi_n H^{(1)} \left|n^{(2)}\right\rangle - E_n^{(2)} \Pi_n \left|n^{(1)}\right\rangle - \left\langle n^{(1)} \middle| n^{(2)} \right\rangle \left|n^{(0)}\right\rangle
\end{aligned}$$

Operator Representations

Often we have an operator A and need to get something like

$$\left\langle n\right| A \left|m\right\rangle = \left(\sum_k \left\langle n^{(k)}\right| \lambda^k \right) A \left(\sum_k \left|m^{(l)}\right\rangle \lambda^k \right)$$

naively, we might think that we can just evaluate this out directly. Unfortunately, because of the truncation done in the perturbation expansion, this would introduce errors. Instead, we need to first expand A itself out as a series of perturbations

$$A = \sum_k A^{(k)} \lambda^k$$

and then we can compute the perturbative correction to the expectation value by matching up terms with the same total value of λ^k. Since we're going to be adding up terms that look like

$$\left\langle n^{(\alpha)}\right| A^{(\gamma)} \left|m^{(\beta)}\right\rangle$$

we unfortunately have three indices to sum over, i.e. we're looking for non-negative, integral solutions to

$$\alpha+\beta+\gamma=k$$

which we'll want to do in two steps:

- Fix a value for $\alpha \in \{0...k\}$
- Sum over the solutions to $\beta+\gamma=k-\alpha$

or written out explicitly

$$\langle n| A |m\rangle^{(k)} = \sum_{\alpha=0}^{k} \sum_{\beta=0}^{k-\alpha} \langle n^{(\alpha)}| A^{(k-\alpha-\beta)} |m^{(\beta)}\rangle$$

which if done out explicitly for $k = 2$ gives us

$$\langle n| A |m\rangle^{(2)} = \langle n^{(0)}| A^{(2)} |m^{(0)}\rangle + \langle n^{(0)}| A^{(1)} |m^{(1)}\rangle + \langle n^{(0)}| A^{(0)} |m^{(2)}\rangle + \langle n^{(1)}| A^{(1)} |m^{(0)}\rangle + \langle n^{(1)}| A^{(0)} |m^{(1)}\rangle + \langle n^{(2)}| A^{(0)} |m^{(0)}\rangle$$

Selection Rules

It's worth considering how this play with selection rules, to improve efficiency. First off, we will note that when we have

$$\langle n^{(\alpha)}| A^{(k-\alpha-\beta)} |m^{(\beta)}\rangle$$

in general $\langle n^{(\alpha)}|$ will be an expansion over a select set of initial states and $|m^{(\beta)}\rangle$ will be an expansion over a different one. We'll write this as

$$\langle n^{(\alpha)}| = \sum_k c_k^{(n)} \langle \phi_k^{(0)}|$$
$$|m^{(\beta)}\rangle = \sum_j c_j^{(m)} |\phi_j{}^{(0)}\rangle$$

Now if $A^{(k-\alpha-\beta)}$ has an associated set of selection rules, $\{\Delta\}$, then we can generate the set of transformed states

$$\{\Delta\} \otimes \{|\phi_j{}^{(0)}\rangle\}$$

and take the intersection of this set with $\{\langle \phi_k^{(0)}|\}$ to determine what our active space is.

Alternately, as long as our operator is Hermitian, we can apply $\{\Delta\}$ to $\{\langle \phi_k^{(0)}|\}$

Degeneracies

The idea in degenerate perturbation theory is to use non-degenerate perturbation theory to create a contracted representation and then do a regular variational calculation in this contracted subspace. To that effect we start with the regular perturbation theory representation of the Hamiltonian

$$H = \sum_k H^{(k)}$$

then we take our initial basis of states $\{|n\rangle^{(0)}\}$ and we partition it into a non-degenerate subspace and a nearly-degenerate subspace, which we're calling $\{|\chi_i\rangle\}$.

We then, temporarily, artificially set all of the problematic off-diagonal elements to zero, i.e. we say

$$\langle\chi_n| H |\chi_m\rangle = \langle\chi_n| H |\chi_n\rangle \delta_{nm}$$

or in terms of matrices, we say

$$H = H_{\text{non-deg}} + H_{\text{deg}}$$

We then treat $H^{\text{non-deg}}$ as usual using non-degenerate perturbation theory.

This gives us a new basis of states to work in $\{|n\rangle\}$, and we can use this to represent the entire Hamiltonian. First, we'll note that

$$\langle n| H^{\text{non-deg}} |m\rangle = \langle n| \sum_k H^{(k)}_{\text{non-deg}} |n\rangle \delta_{nm}$$

and so the difficulty is really just in dealing with the omitted terms.

To do that, we note that from our non-degenerate treatment we have

$$|n\rangle = \sum_m c_m |n\rangle^{(0)}$$

and so then for any two nearly-degenerate states, $|\chi_i\rangle$ and $|\chi_j\rangle$, we need to pull the appropriate c_m coefficients to allow them to couple. We then want to use this as a transformation on the non-zero elements of H^{deg} to give the representation of H^{deg} in the basis of non-degenerate modes. Once we have that, we can diagonalize add it onto our diagonal representation of $H^{\text{non-deg}}$, and diagonalize this to get a transformation from non-degenerate space to the degenerate space. This transformation can then be applied to anything that is represented in the non-degenerate space to take it to the degenerate one.

Sample Implementation

This is a totally general implementation of non-degenerate perturbation theory up to arbitrary order. You've got to supply the perturbations, though.

getVPT2Corrections

Perturbation Theory Hamiltonians

Background

In our representation of our Hamiltonian, we have

$$\mathrm{H}^{(1)}=\sum_{ijk}\frac{1}{2}\frac{\partial g_{ij}}{\partial Q_k}p_i\,Q_k\,p_j+\frac{1}{6}\frac{\partial^3 V}{\partial Q_i\,Q_j\,Q_k}Q_i\,Q_j\,Q_k$$

then if we ask for element of this

$$\begin{aligned}\langle n|\,\mathrm{H}^{(1)}\,|m\rangle&=\langle n|\sum_{ijk}\frac{1}{2}\frac{\partial g_{ij}}{\partial Q_k}p_i\,Q_k\,p_j+\frac{1}{6}\frac{\partial^3 V}{\partial Q_i\,Q_j\,Q_k}Q_i\,Q_j\,Q_k\,|m\rangle\\&=\sum_{ijk}\frac{1}{2}\frac{\partial g_{ij}}{\partial Q_k}\langle n|\,p_i\,Q_k\,p_j\,|m\rangle+\frac{1}{6}\frac{\partial^3 V}{\partial Q_i\,Q_j\,Q_k}\langle n|\,Q_i\,Q_j\,Q_k\,|m\rangle\end{aligned}$$

we realize that we need to get representations of the operators pQp and QQQ

The form that these operator representations should take may not be entirely obvious.

A 2D Example

Representation of $\mathrm{H}^{(1)}$

With all this in place, we can now think about how we will handle just the second part of $\mathrm{H}^{(1)}$

$$\sum_{ijk}\frac{\partial^3 V}{\partial Q_i\,Q_j\,Q_k}Q_i\,Q_j\,Q_k$$

as we've shown previously, we can get a full tensor representation of the derivatives, i.e.

$$(V_{Q^3})_{ijk}=\frac{\partial^3 V}{\partial Q_i\,Q_j\,Q_k}$$

This term has dimension $(M,\ M,\ M)$ where M is the total number of coordinates in our system.

The question, then, is what to do with Q^3?

Here we need to decide how many quanta of excitation we will allow to be in each of our modes. We will let this be N_i. This means our basis elements look like

$$\phi_n=\prod_{i=1}^{M}\varphi_{n_i}$$

and we have a total basis size of

$$N=\prod_{i=1}^{M}N_i$$

and so we expect that $\mathrm{H}^{(1)}$ should be have dimension $(N,\ N)$

Thus we expect to have representation of QQQ that has dimension $(M,\ M,\ M,\ N,\ N)$, which can of course get memory intensive *very* quickly unless we can control the size of N or by computing individual elements of $\mathrm{H}^{(1)}$, like

$$\begin{aligned}\left(\mathrm{H}^{(1)}\right)_{n,m} &= \left(\mathrm{H}^{(1)}\right)_{n_{x_1}\, n_{x_2} \ldots n_{x_M}, m_{x_1}\, m_{x_2} \ldots m_{x_M}} \\ &= V_{Q^3} \odot \left(Q^3_{\ldots\, \mathrm{nm}}\right)\end{aligned}$$

where the $\odot$ is telling us to contract the axes of V_{Q^3} with the first three axes of Q^3

But, saving that for another day, we're still left with the question of how to represent Q^3, given that for $M > 3$ the indices $i,\ j,\ k$ don't cover the full set of combinations of axes.

In this case, we can think of this as

$$Q^3 = Q^3\ I^{(M-3)}$$

and so we end up having

$$Q^3 = I_{N_1} \otimes I_{N_2} \otimes \ldots \otimes Q_i \otimes I_{N_{i+1}} \otimes \ldots \otimes Q_j \otimes I_{N_{j+1}} \otimes \ldots\ Q_k \otimes I_{N_{k+1}} \otimes \ldots \mathrm{I}_{N_M}$$

which is a truly massive tensor

On the other hand, it is also an extremely *sparse* tensor, and so by using sparse matrix methods we can end up storing very little of it, if we decide that we *do* want to do a direct calculation of the tensor.

Explicit Form

There is also a mild complication that we haven't dealt with here, either, which is that sometimes we won't have three distinct indices. That is sometimes we'll have $Q_x\ Q_y\ Q_z$ but other times we might have $Q_x\ Q_y\ Q_x$. These terms will have different representations:

$$\begin{aligned}Q_x\ Q_y\ Q_z &\Rightarrow\ Q_x \otimes Q_y \otimes Q_z \\ Q_x\ Q_y\ Q_x &\Rightarrow {Q_x}^2 \otimes Q_y\end{aligned}$$

where that ${Q_x}^2$ term is, as usual, a matrix power of the representation of Q_x, not a direct term wise multiplication.

This can also make indexing something of a challenge to think about. Let's consider three modes (we'll call them x, y, and z) and in each of these we put up to 1 quantum of excitation.

Then if we're asking for QQQ_{nm} we are really asking for this tensor

$$\mathrm{QQQ}_{nm}=\begin{pmatrix}\begin{pmatrix}(Q_x\,Q_x\,Q_x)_{nm} & (Q_x\,Q_x\,Q_y)_{nm} & (Q_x\,Q_x\,Q_z)_{nm}\\ (Q_x\,Q_y\,Q_x)_{nm} & (Q_x\,Q_y\,Q_y)_{nm} & (Q_x\,Q_y\,Q_z)_{nm}\\ (Q_x\,Q_z\,Q_x)_{nm} & (Q_x\,Q_z\,Q_y)_{nm} & (Q_x\,Q_z\,Q_z)_{nm}\end{pmatrix}\\ \begin{pmatrix}(Q_y\,Q_x\,Q_x)_{nm} & (Q_y\,Q_x\,Q_y)_{nm} & (Q_y\,Q_x\,Q_z)_{nm}\\ (Q_y\,Q_y\,Q_x)_{nm} & (Q_y\,Q_y\,Q_y)_{nm} & (Q_y\,Q_y\,Q_z)_{nm}\\ (Q_y\,Q_z\,Q_x)_{nm} & (Q_y\,Q_z\,Q_y)_{nm} & (Q_y\,Q_z\,Q_z)_{nm}\end{pmatrix}\\ \begin{pmatrix}(Q_z\,Q_x\,Q_x)_{nm} & (Q_z\,Q_x\,Q_y)_{nm} & (Q_z\,Q_x\,Q_z)_{nm}\\ (Q_z\,Q_y\,Q_x)_{nm} & (Q_z\,Q_y\,Q_y)_{nm} & (Q_z\,Q_y\,Q_z)_{nm}\\ (Q_z\,Q_z\,Q_x)_{nm} & (Q_z\,Q_z\,Q_y)_{nm} & (Q_z\,Q_z\,Q_z)_{nm}\end{pmatrix}\end{pmatrix}$$

$$=\begin{pmatrix}\begin{pmatrix}\left(Q_x^3\right)_{nm} & \left(Q_x^2\,Q_y\right)_{nm} & \left(Q_x^2\,Q_z\right)_{nm}\\ \left(Q_x^2\,Q_y\right)_{nm} & \left(Q_x\,Q_y^2\right)_{nm} & (Q_x\,Q_y\,Q_z)_{nm}\\ \left(Q_x^2\,Q_z\right)_{nm} & (Q_x\,Q_z\,Q_y)_{nm} & \left(Q_x\,Q_z^2\right)_{nm}\end{pmatrix}\\ \begin{pmatrix}\left(Q_x^2\,Q_y\right)_{nm} & \left(Q_x\,Q_y^2\right)_{nm} & (Q_y\,Q_x\,Q_z)_{nm}\\ \left(Q_x\,Q_y^2\right)_{nm} & \left(Q_y^3\right)_{nm} & \left(Q_y^2\,Q_z\right)_{nm}\\ (Q_y\,Q_z\,Q_x)_{nm} & \left(Q_y^2\,Q_z\right)_{nm} & \left(Q_y\,Q_z^2\right)_{nm}\end{pmatrix}\\ \begin{pmatrix}\left(Q_x^2\,Q_z\right)_{nm} & (Q_z\,Q_x\,Q_y)_{nm} & \left(Q_x\,Q_z^2\right)_{nm}\\ (Q_z\,Q_y\,Q_x)_{nm} & \left(Q_y^2\,Q_z\right)_{nm} & \left(Q_y\,Q_z^2\right)_{nm}\\ \left(Q_x\,Q_z^2\right)_{nm} & \left(Q_y\,Q_z^2\right)_{nm} & \left(Q_z^3\right)_{nm}\end{pmatrix}\end{pmatrix}$$

Then the question is: what is nm?

To answer this we need to think about how our total Hamiltonian is structured. A given element of this looks like

$$\langle n|\,\mathrm{H}\,|m\rangle=\langle n|\,\mathrm{T}\,|m\rangle+\langle n|\,\mathrm{V}\,|m\rangle$$

and keeping in mind that our basis is the direct product of the basis sets

$$\Phi_x=\{|0\rangle_x,\,|1\rangle_x\}$$
$$\Phi_y=\{|0\rangle_y,\,|1\rangle_y\}$$
$$\Phi_z=\{|0\rangle_z,\,|1\rangle_z\}$$

so our overall basis is

$$\Phi=\{|0\rangle_x|0\rangle_y|0\rangle_z,\,|1\rangle_x|0\rangle_y|0\rangle_z,\,|0\rangle_x|1\rangle_y|0\rangle_z,\,|0\rangle_x|0\rangle_y|1\rangle_z,$$
$$|1\rangle_x|1\rangle_y|0\rangle_z,\,|1\rangle_x|0\rangle_y|1\rangle_z,\,|0\rangle_x|1\rangle_y|1\rangle_z,\,|1\rangle_x|1\rangle_y|1\rangle_z,\}$$

and then n refers to one of these basis functions and m to another, i.e. (removing the redundant subscripts)

$$\Phi_n=|n_x\rangle\,|n_y\rangle\,|n_z\rangle$$

So overall

$$\mathrm{QQQ}_{nm} = \langle n_x| \langle n_y| \langle n_z| \begin{pmatrix} \begin{pmatrix} (Q_x{}^3) & (Q_x{}^2\, Q_y) & (Q_x{}^2\, Q_z) \\ (Q_x{}^2\, Q_y) & (Q_x\, Q_y{}^2) & (Q_x\, Q_y\, Q_z) \\ (Q_x{}^2\, Q_z) & (Q_x\, Q_z\, Q_y) & (Q_x\, Q_z{}^2) \end{pmatrix} \\ \begin{pmatrix} (Q_x{}^2\, Q_y) & (Q_x\, Q_y{}^2) & (Q_y\, Q_x\, Q_z) \\ (Q_x\, Q_y{}^2) & (Q_y{}^3) & (Q_y{}^2\, Q_z) \\ (Q_y\, Q_z\, Q_x) & (Q_y{}^2\, Q_z) & (Q_y\, Q_z{}^2) \end{pmatrix} \\ \begin{pmatrix} (Q_x{}^2\, Q_z) & (Q_z\, Q_x\, Q_y) & (Q_x\, Q_z{}^2) \\ (Q_z\, Q_y\, Q_x) & (Q_y{}^2\, Q_z) & (Q_y\, Q_z{}^2) \\ (Q_x\, Q_z{}^2) & (Q_y\, Q_z{}^2) & (Q_z{}^3) \end{pmatrix} \end{pmatrix} |m_x\rangle |m_y\rangle |m_z\rangle$$

this is a real pain to show in full, so we'll just focus on the three types of terms in the middle block

$$\begin{aligned}
\langle n_x| \langle n_y| \langle n_z|\, Q_y{}^3\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_y|\, Q_y{}^3\, |m_y\rangle \langle n_x|m_x\rangle \langle n_z|m_z\rangle \\
\langle n_x| \langle n_y| \langle n_z|\, Q_x{}^2\, Q_y\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_x|\, Q_x{}^2\, |m_x\rangle \langle n_y|\, Q_y\, |m_y\rangle \langle n_z|m_z\rangle \\
\langle n_x| \langle n_y| \langle n_z|\, Q_x\, Q_y{}^2\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_x|\, Q_x\, |m_x\rangle \langle n_y|\, Q_y{}^2\, |m_y\rangle \langle n_z|m_z\rangle \\
\langle n_x| \langle n_y| \langle n_z|\, Q_y{}^2\, Q_z\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_y|\, Q_y{}^2\, |m_y\rangle \langle n_z|\, Q_z\, |m_z\rangle \langle n_x|m_x\rangle \\
\langle n_x| \langle n_y| \langle n_z|\, Q_y\, Q_z{}^2\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_y|\, Q_y\, |m_y\rangle \langle n_z|\, Q_z{}^2\, |m_z\rangle \langle n_x|m_x\rangle \\
\langle n_x| \langle n_y| \langle n_z|\, Q_y\, Q_x\, Q_z\, |m_x\rangle |m_y\rangle |m_z\rangle &= \langle n_x|\, Q_x\, |m_x\rangle \langle n_y|\, Q_y\, |m_y\rangle \langle n_z|\, Q_z\, |m_z\rangle
\end{aligned}$$

and assuming that x, y, and z all have the same representation for Q this means the whole tensor is

$$\mathrm{QQQ}_{nm}=\begin{pmatrix} \begin{pmatrix} Q^3{}_{n_x m_x}\, \delta_{n_y m_y}\, \delta_{n_z m_z} & Q^2{}_{n_x m_x}\, Q_{n_y m_y}\, \delta_{n_z m_z} & Q^2{}_{n_x m_x}\, Q_{n_z m_z}\, \delta_{n_y m_y} \\ Q^2{}_{n_x m_x}\, Q_{n_y m_y}\, \delta_{n_z m_z} & Q_{n_x m_x}\, Q^2{}_{n_y m_y}\, \delta_{n_z m_z} & Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} \\ Q^2{}_{n_x m_x}\, Q_{n_z m_z}\, \delta_{n_y m_y} & Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} & Q_{n_x m_x}\, Q^2{}_{n_z m_z}\, \delta_{n_y m_y} \end{pmatrix} \\ \begin{pmatrix} Q^2{}_{n_x m_x}\, Q_{n_y m_y}\, \delta_{n_z m_z} & Q_{n_x m_x}\, Q^2{}_{n_y m_y}\, \delta_{n_z m_z} & Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} \\ Q_{n_x m_x}\, Q^2{}_{n_y m_y}\, \delta_{n_z m_z} & Q^3{}_{n_y m_y}\, \delta_{n_x m_x}\, \delta_{n_z m_z} & Q^2{}_{n_y m_y}\, Q_{n_z m_z}\, \delta_{n_x m_x} \\ Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} & Q^2{}_{n_y m_y}\, Q_{n_z m_z}\, \delta_{n_x m_x} & Q_{n_y m_y}\, Q^2{}_{n_z m_z}\, \delta_{n_x m_x} \end{pmatrix} \\ \begin{pmatrix} Q^2{}_{n_x m_x}\, Q_{n_z m_z}\, \delta_{n_y m_y} & Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} & Q_{n_x m_x}\, Q^2{}_{n_z m_z}\, \delta_{n_y m_y} \\ Q_{n_x m_x}\, Q_{n_y m_y}\, Q_{n_z m_z} & Q^2{}_{n_y m_y}\, Q_{n_z m_z}\, \delta_{n_x m_x} & Q_{n_y m_y}\, Q^2{}_{n_z m_z}\, \delta_{n_x m_x} \\ Q_{n_x m_x}\, Q^2{}_{n_z m_z}\, \delta_{n_y m_y} & Q_{n_y m_y}\, Q^2{}_{n_z m_z}\, \delta_{n_x m_x} & Q^3{}_{n_z m_z}\, \delta_{n_x m_x}\, \delta_{n_y m_y} \end{pmatrix} \end{pmatrix}$$

which looks imposing, but really ends up being quite simple. If, for instance, we have

$$\begin{array}{ll} n_x{=}1 & m_x{=}0 \\ n_y{=}0 & m_y{=}1 \\ n_z{=}1 & m_z{=}1 \end{array}$$

based on ordering our basis as

$$\Phi{=}\{|0\rangle_x |0\rangle_y |0\rangle_z,\ |1\rangle_x |0\rangle_y |0\rangle_z,\ |0\rangle_x |1\rangle_y |0\rangle_z,\ |0\rangle_x |0\rangle_y |1\rangle_z,$$

$$|1\rangle_x|1\rangle_y|0\rangle_z,\ |1\rangle_x|0\rangle_y|1\rangle_z,\ |0\rangle_x|1\rangle_y|1\rangle_z,\ |1\rangle_x|1\rangle_y|1\rangle_z,\}$$

we get

$$\mathrm{QQQ}_{65}=\begin{pmatrix}\begin{pmatrix} {Q^3}_{10}\,\delta_{01}\,\delta_{11} & {Q^2}_{10}\,Q_{01}\,\delta_{11} & {Q^2}_{10}\,Q_{11}\,\delta_{01}\\ {Q^2}_{10}\,Q_{01}\,\delta_{11} & Q_{10}\,{Q^2}_{01}\,\delta_{11} & Q_{10}\,Q_{01}\,Q_{11}\\ {Q^2}_{10}\,Q_{11}\,\delta_{01} & Q_{10}\,Q_{01}\,Q_{11} & Q_{10}\,{Q^2}_{11}\,\delta_{01}\end{pmatrix}\\ \begin{pmatrix} {Q^2}_{10}\,Q_{01}\,\delta_{11} & Q_{10}\,{Q^2}_{01}\,\delta_{11} & Q_{10}\,Q_{01}\,Q_{11}\\ Q_{10}\,{Q^2}_{01}\,\delta_{11} & {Q^3}_{01}\,\delta_{10}\,\delta_{11} & {Q^2}_{01}\,Q_{11}\,\delta_{10}\\ Q_{10}\,Q_{01}\,Q_{11} & {Q^2}_{01}\,Q_{11}\,\delta_{10} & Q_{01}\,{Q^2}_{11}\,\delta_{10}\end{pmatrix}\\ \begin{pmatrix} {Q^2}_{10}\,Q_{11}\,\delta_{01} & Q_{10}\,Q_{01}\,Q_{11} & Q_{10}\,{Q^2}_{11}\,\delta_{01}\\ Q_{10}\,Q_{01}\,Q_{11} & {Q^2}_{01}\,Q_{11}\,\delta_{10} & Q_{01}\,{Q^2}_{11}\,\delta_{10}\\ Q_{10}\,{Q^2}_{11}\,\delta_{01} & Q_{01}\,{Q^2}_{11}\,\delta_{10} & {Q^3}_{11}\,\delta_{10}\,\delta_{01}\end{pmatrix}\end{pmatrix}$$

assuming we're working with harmonic oscillator wavefunctions, this gives us

$$Q=\begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0\\ \frac{1}{\sqrt{2}} & 0 & 1\\ 0 & 1 & 0\end{pmatrix}\quad Q^2=\begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{\sqrt{2}}\\ 0 & \frac{3}{2} & 0\\ \frac{1}{\sqrt{2}} & 0 & \frac{5}{2}\end{pmatrix}\quad Q^3=\begin{pmatrix} 0 & \frac{3}{2\sqrt{2}} & 0\\ \frac{3}{2\sqrt{2}} & 0 & 3\\ 0 & 3 & 0\end{pmatrix}$$

and so

$$Q_{10}=\frac{1}{\sqrt{2}}\quad Q_{11}=0\quad Q_{01}=\frac{3}{\sqrt{2}}$$
$$Q^2_{10}=0\quad Q^2_{11}=\frac{3}{2}\quad Q^2_{01}=0$$
$$Q^3_{10}=\frac{3}{\sqrt{2}}\quad Q^3_{11}=0\quad Q^3_{01}=\frac{3}{\sqrt{2}}$$

and so

$$QQQ_{65}=\begin{pmatrix}\begin{pmatrix}\frac{3}{2\sqrt{2}}\delta_{01}\,\delta_{11} & 0*\frac{1}{\sqrt{2}}\,\delta_{11} & 0*0\,\delta_{01}\\ 0*\frac{1}{\sqrt{2}}\,\delta_{11} & \frac{1}{\sqrt{2}}*0\,\delta_{11} & \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0\\ 0*0\,\delta_{01} & \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0 & \frac{1}{\sqrt{2}}*\frac{3}{2}\,\delta_{01}\end{pmatrix}\\ \begin{pmatrix}0*\frac{1}{\sqrt{2}}\,\delta_{11} & \frac{1}{\sqrt{2}}*0\,\delta_{11} & \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0\\ \frac{1}{\sqrt{2}}*0\,\delta_{11} & \frac{3}{2\sqrt{2}}\delta_{10}\,\delta_{11} & 0*0\,\delta_{10}\\ \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0 & 0*0\,\delta_{10} & \frac{1}{\sqrt{2}}\,\frac{3}{2}\,\delta_{10}\end{pmatrix}\\ \begin{pmatrix}0*0\,\delta_{01} & \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0 & \frac{1}{\sqrt{2}}*\frac{3}{2}\,\delta_{01}\\ \frac{1}{\sqrt{2}}*\frac{1}{\sqrt{2}}*0 & 0*0\,\delta_{10} & \frac{1}{\sqrt{2}}*\frac{3}{2}\,\delta_{10}\\ \frac{1}{\sqrt{2}}*\frac{3}{2}\,\delta_{01} & \frac{1}{\sqrt{2}}*\frac{3}{2}\,\delta_{10} & 0\,\delta_{10}\,\delta_{01}\end{pmatrix}\end{pmatrix}=\begin{pmatrix}\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}\\ \begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}\\ \begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}\end{pmatrix}$$

And so looked like a very large tensor turns out to be identically 0. This is a pretty common phenomenon with these and if we are trying to be very computationally efficient we can use selection rules to guide which terms we actually need to consider.

Operator Tensor Symmetries

When we build our representations, one key thing we can take advantage of is the symmetry of the problem. In particular, for *unshared* indices, we should be able to symmetrize at will, so

$$p_i\,Q_j\,p_k = p_i\,p_k\,Q_j = p_k\,p_i\,Q_j$$
$$p_i\,Q_j\,p_i = p_i\,p_i\,Q_j$$

but when indices are shared we lose that flexibility, as we note that

$$p_i\,Q_i\,p_j \neq Q_i\,p_i\,p_j$$

this is effectively due to the lack of commutativity of these terms

This means that when we are determining if two terms are equivalent we need to ask ourselves one main thing: *have we transposed our terms?*

The issue is that this can be tough to figure out. Algorithmically we need to first know which of our terms can be treated as equivalent (e.g. Q and Q, but not Q and p). Then we check to see if

any of our terms have been swapped around.

Sample Code

This provides Mathematica code to obtain explicit symbolic forms of tensor elements

$$(p_i\ Q_j\ Q_k\ p_l)_{1111} = p_1\ Q_1\ Q_1\ p_1$$

productOperatorTensor

testQQ

Tests

State Selection Rules

The number of states that are required to be accurate enough with the perturbation theory grows exponentially, as do the number of potentially relevant transitions. If we have N_{vib} vibrational modes and want to put up to M quanta of excitation into any of these modes, assuming $M < N_{\text{vib}}$ the number of states we're working with can be defined recursively for $M > 1$ as

$$\rho(N_{\text{vib}},\ M) = \rho(N_{\text{vib}},\ M-1) + \rho(N_{\text{vib}}-1,\ M)$$

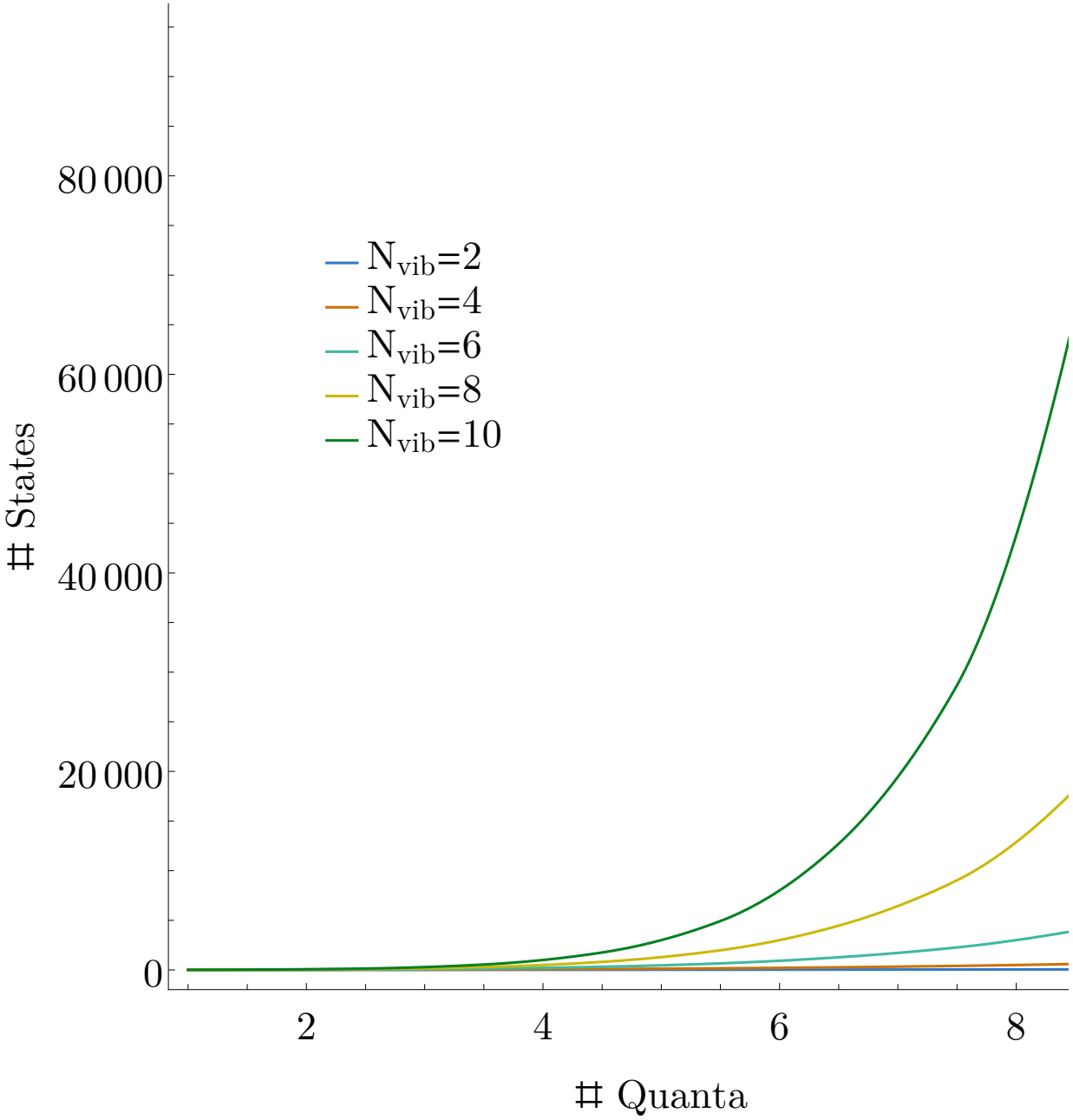
where the $\rho(N_{\text{vib}},\ M-1)$ contribution comes from the fact that the $M-1$ states also have fewer than M quanta and the $\rho(N_{\text{vib}}-1,\ M)$ comes from adding an extra quantum to all of those states & dealing with duplicates. I might actually be mildly over counting in this formula, but at the very least it suffices to show the exponential blow-up we expect.

When $N_{\text{vib}} = 2$, these will grow as the triangular numbers, defined as

$$a(n) = \binom{n+1}{2}$$

and when $\rho(N_{\text{vib}},\ 1) = N_{\text{vib}}$ so this building recursively we're going to by something like a binomial expansion on the binomial numbers. I'll leave the proper analysis of this kind of thing to the number theorists, who have certainly already studied problems like this and just leave you with a plot of what this looks like

Out[2450]=



This is already bad for studying the entire space of highly-excited vibrations. On the other hand, we're mostly interested in vibrations that put up to two quanta in given modes, and especially up to two quanta in modes we care about.

To that effect, rather than computing our Hamiltonian elements in the total set of possibly-coupled states, we'd like to take a set of initial states of interest and generate the space of states that really do couple to those ones. To generate this space, we need to understand how our zero-order transform under application of our perturbations. We'll do this with the harmonic Hamiltonian that we're mostly to want to use, but the ideas apply to any initial basis you might care

about.

$H^{(1)}$

By assumption, $H^{(0)}$ is diagonal, so first we'll focus in on $H^{(1)}$ where we have

$$H^{(1)}=\nabla_Q G\odot pQp+\nabla_Q F\odot QQQ$$

and now if we stick any two states in there we have

$$\langle n_1,\ n_2,\ ...|\ H^{(1)}\ |m_1,\ m_2,\ ...\rangle=$$
$$\nabla_Q G\odot\langle n_1,\ n_2,\ ...|\ pQp\ |m_1,\ m_2,\ ...\rangle+\nabla_Q F\odot\langle n_1,\ n_2,\ ...|\ QQQ\ |m_1,\ m_2,\ ...\rangle$$

and now we have to deal with the terms that look like

$$\langle n_1,\ n_2,\ ...|\ p_iQ_jp_k\ |m_1,\ m_2,\ ...\rangle=\langle n_i,\ n_j,\ n_k|\ p_iQ_jp_k\ |m_i,\ m_j,\ m_k\rangle\ \langle\{n_l\}\backslash\{i,\ j,\ k\}|\{m_l\}\backslash\{i,\ j,\ k\}\rangle$$
$$\langle n_1,\ n_2,\ ...|\ Q_iQ_jQ_k\ |m_1,\ m_2,\ ...\rangle=\langle n_i,\ n_j,\ n_k|\ Q_iQ_jQ_k\ |m_i,\ m_j,\ m_k\rangle\ \langle\{n_l\}\backslash\{i,\ j,\ k\}|\{m_l\}\backslash\{i,\ j,\ k\}\rangle$$

the overlap term is just a big pile of delta functions

$$\langle\{n_l\}\backslash\{i,\ j,\ k\}|\{m_l\}\backslash\{i,\ j,\ k\}\rangle=\prod_{\alpha\notin\{i,\ j\ k\}}\delta_{n_\alpha\ m_\alpha}$$

and so we can clearly only ever change *at most* three indices at once through $H^{(1)}$.

Next we need to consider the three cases for $\{i,\ j,\ k\}$

Case 1: $i=j=k$

This is the simplest case, where we've got the same index three times. From our δ terms, this means we can only change one mode. Constructing our raising/lower representation of the operators we have

$$\begin{aligned}p_iQ_ip_i&=\left(a_i-a_i^\dagger\right)\left(a_i+a_i^\dagger\right)\left(a_i-a_i^\dagger\right)\\&=\left(a_i-a_i^\dagger\right)\left(a_i\ a_i+a_i^\dagger\ a_i-a_i^\dagger\ a_i^\dagger\right)\\&=a_i\ a_i\ a_i-a_i^\dagger\ a_i\ a_i-a_i^\dagger\ a_i^\dagger\ a_i+a_i^\dagger\ a_i^\dagger\ a_i^\dagger\\Q_iQ_iQ_i&=a_i\ a_i\ a_i+a_i^\dagger\ a_i\ a_i+a_i^\dagger\ a_i^\dagger\ a_i+a_i^\dagger\ a_i^\dagger\ a_i^\dagger\end{aligned}$$

which means we can raise/lower our quanta by one or three through this operator

Case 2: $i\neq j=k$

In this case, we can change up to two modes. Constructing our raising/lower representation of the operators we have a few different flavors, based on the different permutations we of the indices we can do, but we'll focus on

$$\begin{aligned}Q_iQ_jQ_j&=\left(a_i+a_i^\dagger\right)\left(a_j+a_j^\dagger\right)\left(a_j+a_j^\dagger\right)\\&=\left(a_i+a_i^\dagger\right)\left(a_j\ a_j+a_j^\dagger\ a_j+a_j^\dagger\ a_j^\dagger\right)\end{aligned}$$

and from this we can change one mode by up to 2 and another mode by one

Case 3: $i \neq j \neq k$

We're just gonna jump ahead to the punch-line and leave working it out to the reader. We can change three modes at a time and each mode can change by one quantum.

$H^{(2)}$

We'll recall

$$H^{(1)}=\nabla_{Q^2} G \odot pQQp+\nabla_{Q^2} F \odot QQQQ$$

this will allow things to change by up to 4 quanta in up to 4 modes.

Tensor Operations

Tensor Addition and Scalar Tensor Multiplication

These two operations will be defined elementwise, that is to say

Given $\boldsymbol{A}$, $\boldsymbol{B}$ tensors, c a scalar,

$$(A+B)_{i_1\, i_2\, i_3\, \ldots i_n}=A_{i_1\, i_2\, i_3\, \ldots i_n}+B_{i_1\, i_2\, i_3\, \ldots i_n}$$

$$(\mathrm{cA})_{i_1\, i_2\, i_3\, \ldots i_n}=c*A_{i_1\, i_2\, i_3\, \ldots i_n}$$

Or extending this a bit, given $\boldsymbol{A}$, $\boldsymbol{B}$ tensors, c, d scalars,

$$(\mathrm{cA+dB})_{i_1\, i_2\, i_3\, \ldots i_n}=\mathrm{cA}_{i_1\, i_2\, i_3\, \ldots i_n}+\mathrm{dB}_{i_1\, i_2\, i_3\, \ldots i_n}$$

This has a few important ramifications, namely

- Tensors can only be added if all of their dimensions are aligned
- Tensor scalar multiplication and tensor addition do not change the overall dimension of the tensor
- Tensor scalar multiplication distributes over tensor addition
- Tensor scalar multiplication and tensor addition can be used to form vector spaces of tensors
- All linear operators (e.g. derivatives) will distribute and act normally with the scalar multiplications

Tensor Transposition

This operations is the analog of the matrix transpose in that it changes the way indexing works in a tensor

Given a tensor $\boldsymbol{A}$ with axes i_1, i_2, ... i_N we define the tensor transpose $\boldsymbol{A}_{T(\{k_m\})}$ such that

$$A_{T(k_1,\ k_2,\ldots,k_N)} \text{ has axes } i_{k_1},\ i_{k_2},\ \ldots\ i_{k_N}$$

$$(A_{T(k_1,\ k_2,\ldots,k_N)})_{j_1\ j_2\ \ldots,j_N}=A_{k_{j_1}k_{j_2}\ \ldots\ k_{j_N}}$$

Note that for the case where N = 2 the traditional transpose is

$$A^{\mathrm{T}}=A_{T(2,\ 1)}$$

Axes Shift

If we are simply shift one axis to a new position, we define an alternate notation for convenience

Given a tensor $\boldsymbol{A}$ with axes i_1, i_2, ... i_N, assuming WLOG that $n>m$

$$A^{n\to m}=A_{T(1,\ 2,\ \ldots,\ m-1,\ n,\ m,\ m+1,\ \ldots,\ n-1,\ n+1,\ \ldots,\ N)}$$

Axes Swap

If we are simply swap two axes, we define an alternate notation for convenience

Given a tensor $\boldsymbol{A}$ with axes i_1, i_2, ... i_N, assuming WLOG that $n>m$

$$A^{n\leftrightarrow m}=A_{T(1,\ 2,\ \ldots,\ m-1,\ n,\ m+1,\ \ldots,\ n-1,\ m,\ n+1,\ \ldots,\ N)}$$

Property 1 (Derivative Relations)

WLOG we will state this only for swaps, but the same ideas hold for standard transpositions

Scalar Derivatives

Given a swap $\boldsymbol{A}^{n\to m}$ and a single cordinate x

$$\frac{\partial}{\partial x}\boldsymbol{A}^{n\to m}=\left(\frac{\partial}{\partial x}\boldsymbol{A}\right)^{n\to m}$$

Proof

$\frac{\partial}{\partial x}$ is an elementwise operator

Vector Derivatives

Given a swap $\boldsymbol{A}^{n\to m}$ and a vector of cordinates X

$$\nabla_{X^k}A^{n\to m}=(\nabla_X\boldsymbol{A})^{n+k\to m+k}$$

Proof

∇_{X^k} increases the rank of the tensor by k but operates elementwise

Property 2 (Composition)

Given a tensor $\boldsymbol{A}$ and two transpositions $k_1, k_2,\ldots,k_N$ and $j_1, j_2,\ldots,j_N$

$$(\mathrm{A}_{T(k_1,\ k_2,\ldots,k_N)})_{T(j_1,\ j_2,\ldots,j_N)}=\mathrm{A}_{T(k_{j_1},\ k_{j_2},\ldots,k_{j_N})}$$

Proof

By definition?

Property 3 (Indexing)

Given a transpositon $\boldsymbol{A}_{T(k_1,\ k_2,\ldots,k_N)}$ and an ordering of the k_n, $\{o_n\}$

$$(A_{T(k_1,\ k_2,\ldots,k_N)})_{i_1\ i_2\ldots\ i_N}=A_{i_{o_1}\ i_{o_2}\ \ldots \mathrm{i}_{o_N}}$$

(By ordering we mean a sequence such that $\forall_n\ k_{o_n}< k_{o_{n+1}}$)

Proof

We won't do a full proof, which would basically have to be done by induction by showing that it works for a single axis swap and then showing that if it works for n swaps it works for $n+1$

Assume that $\boldsymbol{A}$ has axes $\{a_n\} \Rightarrow A_{T(k_1,\ k_2,\ldots,k_N)}$ has axes $\{a_{k_n}\}$

When we ask for element $i_1\ i_2\ldots\ i_N$ we are asking, therefore, for element i_m of axis a_{k_m}

E.g. suppose m = 5, i_5 will be taken from a_{k_5} then suppose k_5 = 2

$\Rightarrow$ we want element i_5 from axis 2 in the original

$\Rightarrow$ we need to find an *ordering* of the $i_1\ i_2\ldots\ i_N$ such that o_2 = 5

$\therefore$ we use the ordering of the $\{k_n\}$

Corr (Shift Indexing)

Given a shift $\boldsymbol{A}^{n\to m}$ WLOG assume $n < m$

$$(A^{n\to m})_{i_1,\ldots,\ i_n,\ldots,\ i_m,\ \ldots\ i_N}=A_{i_1,\ldots,\ i_m,\ i_n,\ \ldots,\ i_{m-1},\ i_{m+1},\ \ldots,\ i_N}$$

or in more prosaic terms, the indices shift in the opposite direction that the axes shift

Property 4 (Shift Compositions)

Given a double shift $(\boldsymbol{A}^{n\to m})^{l\to k}$

if m=l, $(\boldsymbol{A}^{n\to m})^{l\to k} = \boldsymbol{A}^{n\to k}$, otherwise there is no reduction to be had

Table of N=3 cases

Proof

WLOG we will assume $\boldsymbol{A}$ has dimension $N = \max\{n, m, l, k\} - \min\{n, m, l, k\}$

Case 1 : n<m<l< k

If $n<m<l<k$, we have two distinct uncomposable shift operations, $n \to m$ and $l \to k$

Note that in this case and only in this case (I think) the operations commute

Case 2: n<m=l<k

WLOG suppose n=1 and k=N

$$\begin{aligned}\left(A^{1\to m}\right)^{m\to N} &= \left(A_{T(2,\ \ldots,\ m-1,\ 1,\ m,\ \ldots,\ N)}\right)_{T(1,\ \ldots,\ m-1,\ m+1,\ \ldots,\ N,\ m)} \\ &= A_{T(2,\ \ldots,\ N,\ 1)} \\ &= A^{1\to N}\end{aligned}$$

Case 3: n<m & l<k & n=l

WLOG suppose n=1 and k=N

$$\begin{aligned}\left(A^{1\to m}\right)^{1\to N} &= \left(A_{T(2,\ \ldots,\ m-1,\ 1,\ m,\ \ldots,\ N)}\right)_{T(2,\ \ldots,\ N,\ 1)} \\ &= A_{T(3,\ \ldots,\ m-1,\ 1,\ m,\ \ \ldots,\ N,\ 2)}\end{aligned}$$

$\therefore$ Uncomposable in general

Case 4: n<m & l<k & m=k

WLOG suppose n=1 and k=N

$$\begin{aligned}\left(A^{1\to N}\right)^{l\to N} &= \left(A_{T(2,\ \ldots,\ N,\ 1)}\right)_{T(1,\ \ldots,\ k-1,\ k+1,\ \ldots,\ N,\ k)} \\ &= A_{T(2,\ \ldots,\ N,\ 1,\ k)}\end{aligned}$$

$\therefore$ Uncomposable in general

Case 5: n>m & l<k & n=k

WLOG suppose m=1 and k=N

$$\left(A^{n\to 1}\right)^{l\to N} = A_{T(N,\ 1,\ \ldots,\ l-1,\ l+1,\ \ldots,\ N-1,\ l)}$$

$\therefore$ Uncomposable in general

Tensor Derivatives

Given a tensor **A**, and some variable x upon which the elements of **A** may depend we'll define the derivative element-wise such that

$$\left(\frac{\partial}{\partial x}A\right)_{i_1\, i_2 \ldots\, i_n} = \frac{\partial}{\partial x}A_{i_1\, i_2 \ldots\, i_n}$$

We will also define derivative tensors such that given a vector of coordinates $\boldsymbol{x}$ and a scalar function f,

$$\nabla_x f = \begin{pmatrix} \frac{\partial}{\partial x_1}f & \frac{\partial}{\partial x_2}f & \cdots & \frac{\partial}{\partial x_n}f \end{pmatrix}$$

if instead of scalar f we have a tensor $\mathbf{A}$ everything is basically the same but we get the tensor of derivatives

Higher dimensional tensor derivative operators may also be defined, e.g. given another vector of coordinates $\mathbf{y}$ we can have

$$\nabla_{x,y} f = \begin{pmatrix} \frac{\partial^2}{\partial x_1\, \partial y_1}f & \frac{\partial^2}{\partial x_1\, \partial y_2}f & \cdots & \frac{\partial^2}{\partial x_1\, \partial y_m}f \\ \frac{\partial^2}{\partial x_2\, \partial y_1}f & & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial^2}{\partial x_n\, \partial y_1}f & & \cdots & \frac{\partial^2}{\partial x_n\, \partial y_m}f \end{pmatrix}$$

Chain Rule

We'll define the chain rule over a tensor of derivatives in the standard kind of way by supplying a Jacobian for the transformation, e.g. given a vector of coordinates $\boldsymbol{x}$ and a function f and a set of coordinates upon which f depends, $\mathbf{y}$,

$$\nabla_x f = \nabla_x y\, \nabla_y f$$

The same basic form holds for higher dimensional derivative tensors, e.g. given a vector valued function $\mathbf{g}$

$$\nabla_x g = \nabla_x y\, \nabla_y g$$

And for a general derivative tensor for a tensor valued function $\mathbf{A}$

$$\nabla_x A = \nabla_x y \odot \nabla_y A$$

where $\odot$ is the tensor dot operation talked about later

We'll also note that this still works in a slightly different form with univariate derivatives

$$\frac{\partial}{\partial x_i}f = (\nabla_x y)_i\, \nabla_y f$$

Proof

Case 1:

$$\begin{aligned}(\nabla_x f)_i &= \frac{\partial}{\partial x_i} f \\ &= \sum_{j=1}^{N} \frac{\partial y_j}{\partial x_i} \frac{\partial}{\partial y_j} f \\ &= (\nabla_x y)_i \cdot \nabla_y f\end{aligned}$$

Case 2:

$$\begin{aligned}(\nabla_x g)_{ij} &= \left(\frac{\partial}{\partial x_i} g \right)_j \\ &= \frac{\partial}{\partial x_i} g_j \\ &= \sum_k \frac{\partial y_k}{\partial x_i} \frac{\partial}{\partial y_k} g_j \\ &= (\nabla_x y)_i \cdot (\nabla_y g_j)\end{aligned}$$

Case 3:

Left to reader

Tensor Dot

Definition

We'll define the tensor dot $\odot$ to be a binary tensor operation that generalizes the dot product. It will operate recursively such that:

- Given a vector $\mathbf{a}$ of length N and tensor $\mathbf{B}$ with primary dimension N we'll define this such that

$$a \bigodot B = \sum_{i=1}^{N} a_i \, B_i$$

- Given a tensor $\mathbf{A}$ with primary dimension N and innermost dimension M and a tensor $\mathbf{B}$ with primary dimension M we'll define this such that

$$A \bigodot B = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)$$

Property 1 (associativity)

Given $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ tensors,

$$(A \odot B) \odot C = A \odot (B \odot C)$$

Proof

LTR

Property 2 (dot product)

Given vectors $\boldsymbol{a}$, $\boldsymbol{b}$ of length N

$$a \odot b \;=\; a \cdot b$$

where $\cdot$ is the classic dot product

Proof

Basically just by definition

$$a \odot b = \sum_{i=1}^{N} a_i \, b_i = a \cdot b$$

Property 3 (vector matrix)

Given a vector $\boldsymbol{a}$ of length N and matrix $\boldsymbol{B}_{\mathrm{NxM}}$

$$a \odot B \;=\; \mathrm{aB}$$

where **aB** is interpreted in the normal sense of a matrix product

Proof

We start with

$$a \odot B = \sum_{i=1}^{N} a_i \, B_i$$

Given $1 \le i \le N$

$$a_i \, B_i = (a_i \, B_{\mathrm{i1}} \quad a_i \, B_{\mathrm{i2}} \quad \cdots \quad a_i \, B_{\mathrm{iM}})$$

then

$$\sum_{i=1}^{N} a_i \, B_i = \left(\sum_{i=1}^{N} a_i \, B_{\mathrm{i1}} \quad \sum_{i=1}^{N} a_i \, B_{\mathrm{i2}} \quad \cdots \quad \sum_{i=1}^{N} a_i \, B_{\mathrm{iM}} \right)$$
$$= (a \cdot B_{:1} \quad a \cdot B_{:2} \quad \cdots \quad a \cdot B_{:M})$$

which is of course the classic vector matrix product

Property 4 (matrix product)

Given matrices $\boldsymbol{A}_{\mathrm{NxM}}$ and $\boldsymbol{B}_{\mathrm{MxL}}$

$$A \odot B \ = \text{AB}$$

Proof

$$A \odot B = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)$$

Given $1 \le i \le M$, then, by Property 2,

$$(A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B) = (A_1 \ B \quad A_2 \ B \quad \cdots \quad A_N \ B)$$
$$= \begin{pmatrix} A_1 \cdot B_{:1} & A_1 \cdot B_{:2} & \cdots & A_1 \cdot B_{:L} \\ A_2 \cdot B_{:1} & & & \\ \vdots & & \ddots & \vdots \\ A_N \cdot B_{:1} & & \cdots & A_N \cdot B_{:L} \end{pmatrix}$$

which is how **AB** is defined

Property 5 (indexing)

Given $\boldsymbol{A}$, $\boldsymbol{B}$ tensors of dimension N and M, suppose k>0, we have

$$(A \odot B)_{i_1 \, i_2 \ldots \, i_{N-1} \, j_2 \ldots j_M} = A_{i_1 \, i_2 \ldots \, i_{N-1}} \cdot B_{:j_2 \ldots j_M} \tag{1}$$

$$(A \odot B)_{i_1 \, i_2 \ldots \, i_{N-k}} = A_{i_1 \, i_2 \ldots \, i_{N-1}} \odot B \tag{2}$$

$$(A \odot B)_{i_1 \, i_2 \ldots \, i_{N-1} \, j_2 \ldots j_{M-k}} = A_{i_1 \, i_2 \ldots \, i_{N-1}} \odot B_{:j_2 \ldots j_{M-k}:\ldots} \tag{3}$$

where : means to index every element along that axis

Intuitively this means that we simply distribute the indices from left to right until we have run out

Proof

(I proved this for an older statement first so the proof doesn't really directly correspond to what I stated, but is equivalent)

(1)

LTR

(2)

We will show this first for the case that k = N-1

$$(A \odot B)_i = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)_i$$
$$= A_i \odot B$$

Then we'll do this by induction, à la Property 6

Base Case

Suppose N=2, this reduces to (1), therefore it holds

Induction on N:

Suppose this holds for all $\boldsymbol{A}'$ with dimension $N-1$. Next as we have

$$A \odot B = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)$$

and $\forall_i \dim(A_i) = N-1$

$$\begin{aligned}(A \odot B)_{i_1\, i_2 \ldots\, i_{N-1}} &= (A_{i_1} \odot B)_{i_2 \ldots\, i_{N-1}} \\ &= (A_{i_1})_{i_2 \ldots\, i_{N-1}} \odot B \\ &= A_{i_1\, i_2 \ldots\, i_{N-1}} \odot B\end{aligned}$$

(3)

By (1) we have:

$$(A \odot B)_{i_1\, i_2 \ldots\, i_{N-1}\, j_2 \ldots \mathrm{j}_M} = (A_{i_1\, i_2 \ldots\, i_{N-1}} \odot B)_{j_2 \ldots \mathrm{j}_M}$$

Next note that $\dim(A_{i_1\, i_2 \ldots\, i_{N-1}}) = 1$, so

$$\begin{aligned}(A_{i_1\, i_2 \ldots\, i_{N-1}} \odot B)_{j_2 \ldots \mathrm{j}_M} &= \left(\sum_{k=1}^{N} (A_{i_1\, i_2 \ldots\, i_{N-1}})_k \, B_k \right)_{j_2 \ldots \mathrm{j}_M} \\ &= \sum_{k=1}^{N} (A_{i_1\, i_2 \ldots\, i_{N-1} k} \, B_k)_{j_2 \ldots \mathrm{j}_M} \text{ (by distribution of indexing)} \\ &= \sum_{k=1}^{N} A_{i_1\, i_2 \ldots\, i_{N-1} k} \, B_{k\, j_2 \ldots \mathrm{j}_M} \text{(by commutation of indexing and scalar mul.)} \\ &= A_{i_1\, i_2 \ldots\, i_{N-1}} \cdot B_{:\, j_2 \ldots \mathrm{j}_M}\end{aligned}$$

Property 6 (dimension)

Given $\boldsymbol{A}$ and $\boldsymbol{B}$ tensors,

$$\dim(A \odot B) = \dim(A) + \dim(B) - 2$$

Proof

Lemma:

If $\dim(A) = 1$,

$$\dim(A \odot B) = \dim(B) - 1$$

Lemma Proof:

$$A \bigodot B = \sum_{i=1}^{N} A_i \, B_i$$

Given $\boldsymbol{A}'$, $\boldsymbol{B}'$ tensors with the same dimensions, we'll recall that

$$\dim(A' + B') = \dim(A') = \dim(B')$$

therefore

$$\begin{aligned}\dim\left(A \bigodot B\right) &= \dim\left(\sum_{i=1}^{N} A_i\, B_i\right) \\ &= \dim(B_i) \\ &= \dim(B) - 1\end{aligned}$$

QED

Returning to the main problem, we'll use induction on dim(A), taking the Lemma as our base case

Suppose that $\forall A'$ with $\dim(A')=n-1$, $\dim(A'\odot B)=n-1+\dim(B)-2$

Given A with $\dim(A) = n$,

$$A \bigodot B = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)$$

Given $1 \le i \le N$,

$$\begin{aligned}\dim(A_i \odot B) &= n-1+\dim(B)-2 \\ &= n+\dim(B)-3 \\ &= \dim(A)+\dim(B)-3\end{aligned}$$

Then as

$$A \bigodot B = (A_1 \odot B \quad A_2 \odot B \quad \cdots \quad A_N \odot B)$$

since we have added a new outer dimension, we have

$$\begin{aligned}\forall_i \dim(A \odot B) &= \dim(A_i \odot B) + 1 \\ &= \dim(A)+\dim(B)-3+1 \\ &= \dim(A)+\dim(B)-2\end{aligned}$$

And then by induction we have this for all n

Property 7 (scalar multiplication)

Given $\boldsymbol{A}$, $\boldsymbol{B}$, tensors, c a scalar,

$$A \odot (\mathrm{cB}) = c(A \odot B)$$

Proof

The general strategy for this proof would be the same as for Property 6, where we show it holds for tensors of dimension one and then generalize upwards via induction on the dimension of **A** or **C**. In the interest of time, we will simply show that it operates for dim(A)=1

Suppose $\dim(A) = 1$,

$$
\begin{aligned}
A \bigodot (cB) &= \sum_{i=1}^{N} A_i (cB_i) \\
&= c \sum_{i=1}^{N} A_i \, B_i \\
&= c(A \odot B)
\end{aligned}
$$

Property 8 (distributivity)

Given $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ tensors,

$$A \odot (B + C) = A \odot B + A \odot C$$

and

$$(A + B) \odot C = A \odot C + B \odot C$$

Proof

As in Property 7, we will simply show that it operates for tensors of dimension one.

Suppose $\dim(A) = 1$,

$$
\begin{aligned}
A \odot (B + C) &= \sum_{i=1}^{N} A_i * (B + C)_i \\
&= \sum_{i=1}^{N} A_i * (B_i + C_i) \\
&= \sum_{i=1}^{N} A_i \, B_i + A_i \, C_i \\
&= \sum_{i=1}^{N} A_i \, B_i + \sum_{i=1}^{N} A_i \, C_i \\
&= A \odot B + A \odot C
\end{aligned}
$$

Further, now suppose $\dim(B) = 1$,

$$
\begin{aligned}
(A + B) \odot C &= \sum_{i=1}^{N} (A + B)_i \, C_i \\
&= \sum_{i=1}^{N} (A_i + B_i) \, C_i \\
&= \sum_{i=1}^{N} A_i \, C_i + \sum_{i=1}^{N} B_i \, C_i \\
&= A \odot C + B \odot C
\end{aligned}
$$

Property 9 (product rule)

9.a: scalar derivatives

Given $\boldsymbol{A}$, $\boldsymbol{B}$ tensors, x some variable upon which the elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ may depend

$$\frac{\partial}{\partial x} A \odot B = \left(\frac{\partial}{\partial x} A\right) \odot B + A \odot \left(\frac{\partial}{\partial x} B\right)$$

Proof

As in Property 7, we will simply show that it operates for tensors of dimension one.

Suppose $\dim(A) = 1$,

$$\begin{aligned} \frac{\partial}{\partial x} A \odot B &= \frac{\partial}{\partial x} \sum_{i=1}^{N} A_i \, B_i \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial x} (A_i \, B_i) \end{aligned}$$

then as the derivative will operate element−wise on the tensor—

is this obvious?—we will have

$$\begin{aligned} \frac{\partial}{\partial x} (A_i \, B_i)_{j_1 \, j_2 \, \ldots \mathrm{j}_n} &= \frac{\partial}{\partial x} A_i (B_i)_{j_1 \, j_2 \, \ldots \mathrm{j}_n} \\ &= \left(\frac{\partial}{\partial x} A_i\right) (B_i)_{j_1 \, j_2 \, \ldots \mathrm{j}_n} + A_i \left(\frac{\partial}{\partial x} (B_i)_{j_1 \, j_2 \, \ldots \mathrm{j}_n}\right) \end{aligned}$$

therefore

$$\begin{aligned} \frac{\partial}{\partial x} A \odot B &= \sum_{i=1}^{N} \frac{\partial}{\partial x} (A_i \, B_i) \\ &= \sum_{i=1}^{N} \left(\frac{\partial}{\partial x} A_i\right) B_i + A_i \left(\frac{\partial}{\partial x} B_i\right) \\ &= \sum_{i=1}^{N} \left(\frac{\partial}{\partial x} A_i\right) B_i + \sum_{i=1}^{N} A_i \left(\frac{\partial}{\partial x} B_i\right) \\ &= \left(\frac{\partial}{\partial x} A\right) \odot B + A \odot \left(\frac{\partial}{\partial x} B\right) \end{aligned}$$

9.b: vector derivatives

For vector derivatives this has a slightly different form

$$\nabla_X (A \odot B) = (\nabla_X A) \odot B + (A \odot (\nabla_{X_i} B))_{i=1\ldots}$$

or, using transposition notation, if A is of rank a and B is of rank b

$$\nabla_X(A\odot B)=(\nabla_X A)\odot B + \left(A\odot(\nabla_X B)^{2\to 1}\right)^{a\to 1}$$

Proof

Property 10 (transposition)

Given $\boldsymbol{A}$ with axes $\{a_n\}_{n=1}^{N+1}$ $\boldsymbol{B}$ with asxes $\{b_m\}_{m=1}^{M+1}$ and some transposition $\{k_l\}$,

if $\forall_i\ i\leq N \Rightarrow k_i \leq N$ and $\forall_j\ j>N \Rightarrow k_j>N$,

$$(A\odot B)_{T(\{k_l\})}=A_{T(\{k_l|\, l\leq N\})}\odot B_{T(\{k_l-N|\, l>N\})}$$

if $\forall_i\ i>M \Rightarrow k_i \leq M$ and $\forall_j\ j\leq M \Rightarrow k_j>M$,

$$(A\odot B)_{T(\{k_l\})}=B_{T(\{k_l|\, l\leq M\})}\odot A_{T(\{k_l-M|\, l>M\})}$$

otherwise $\nexists$ $\{i_l\}$, $\{j_l\}$ such that

$$(A\odot B)_{T(\{k_l\})}=A_{T(\{i_l\})}\odot B_{T(\{j_l\})} \text{ or } (A\odot B)_{T(\{k_l\})}=B_{T(\{j_l\})}\odot A_{T(\{i_l\})}$$

Relation to Einstein Summation Notation (einsum)

Basically a tensor dot is a simpler version of einsum where we only contract along the inner dimensions, e.g.

Given tensors $\boldsymbol{A}_{\text{ijkl}}$ and $\boldsymbol{B}_{\text{l}\alpha\beta\gamma}$ we have

$$A\odot B = A_{\text{ijkl}}\, B_{\text{l}\alpha\beta\gamma}=(\text{AB})_{\text{ijk}\alpha\beta\gamma}$$

Note that when written like this the validity of Properties 2-6 are evident "by notation"

Notations

Since these derivatives involve a large number of $\nabla_{\cdot}\ \cdot$, $x^{\to\cdot}$, and $\cdot\odot\cdot$ operations we'll introduce a convenient shorthand where

$$\nabla_x A \equiv A_x$$
$$A^{a\to b}\equiv A^{a:b}$$
$$A\odot B\equiv A\, B$$

This means the product rule for tensor derivatives can be expressed as

$$\nabla_x(A\odot B)=A_x\, B+\left(A(B_x)^{2:1}\right)^{a:1}$$

Contraction Notation

We'll introduce a convenient notation to cut down on the number of parens we need where we define

$$A\langle n\rangle B = AB^{n:1}$$

to be the tensor dot operation applied along the n^{th} axis of B. Notably

$$A\langle 1\rangle B = \text{AB}$$

we'll also define

$$A\langle m,\ n\rangle B = A^{m:N_A}\ B^{n:1}$$

to be the contraction of A and B along axes m and n.

To cut down on parentheses, these will be further defined to be right-associative

Code

Here's a quick implementation for getting these derivatives automatically

Setup

In[3241]:=
```
BeginPackage["TensorChaining`"];
ClearAll["TensorChaining`*"];

tTens::usage = "";
tBasis::usage = "";
tAxes::usage = "";

tDims::usage = "";
tMul::usage = "";
tPlus::usage = "";
tFlip::usage = "";
tTransp::usage = "";
tDer::usage = "";
tRank::usage = "";

$tReduce::usage = "";

Begin["`Private`"];

tBasis /: Length[tBasis[sym_, nels_]] := nels;
tAxes /: Length[tAxes[index_, nels_]] := nels;

tTens /: NonCommutativeMultiply[a_tTens, b_tTens] :=
  tMul[a, b];
tTens /: Plus[a_tTens, b_tTens] :=
  tMul[a, b];
```

```
tAxes[index_, nels_] /; TrueQ[$tReduce] := Table[Indexed[index, n], {n, nels}];
tDims[tTens[A_, axes_, basis_]] :=
  With[{a = Head[axes]}, a @@ axes];
tDims[tTransp[a_, reorder_]] := tDims[a][[reorder]];
tDims[tDer[a_, v_, n_ : 1]] := Prepend[tDims[a], ConstantArray[Length[v], n]];
tDims[tMul[a_, b_]] := Join[Most[tDims[a]], Rest[tDims[b]]];

tRank[t_] := Length@tDims[t];

tFlip[a_, ax1_ → ax2_] /; TrueQ[$tReduce] :=
  tTransp[
   a,
   Insert[
    Delete[Range[tRank[a]], ax1],
    ax1,
    ax2
   ]
  ];
tTransp[tTransp[A_, o1_], o2_] :=
  tTransp[A, o1[[o2]]];


tDer[t : tTens[A_, axes_, basis_], v_] /; v =!= basis :=
  tMul[tDer[basis, v], tDer[t, basis]];
tDer[tMul[A_, B_], v_, Optional[1, 1]] :=
  tPlus[
   tMul[tDer[A, v], B],
   tFlip[
    tMul[A, tFlip[tDer[B, v], 2 → 1]],
    tRank[A] → 1
   ]
  ];
tDer[t : tMul[A_, B_], v_, n_] :=
  Nest[
   tDer[#, v] &,
   t,
   n
  ];
tDer[tDer[A_, v_, m_ : 1], v_, n_ : 1] :=
  tDer[A, v, n + m];
tDer[tTransp[A_, flips_], v_, n_ : 1] :=
  tTransp[tDer[A, v, n], n + flips];
tDer[tFlip[A_, f1_ → f2_], v_, n_ : 1] :=
  tFlip[tDer[A, v, n], n + f1 → n + f2];
```

```
Format[t : tTens[A_, axes_, basis_]] :=
  Interpretation[
   Style[A, FontWeight → Bold](*Subsuperscript[A, axes, basis]*),
   t
  ];
Format[t : tBasis[A_, nels_]] :=
  Interpretation[
   A,
   t
  ];
Format[t : tTransp[a_, b_]] :=
  Interpretation[
   Superscript[a, Row@Riffle[b, ","]],
   t
  ];
Format[t : tFlip[a_, b_]] :=
  Interpretation[
   Superscript[a, b],
   t
  ];
Format[t : tDer[a_, b_, n_ : 1]] :=
  Interpretation[
   Row@{Subsuperscript["∂", b, If[n > 1, n, ""]], a},
   t
  ];
Format[t : tMul[a_, b_]] :=
  Interpretation[
   Row@{a, " ", b},
   t
  ];
Format[t : tPlus[a_, b_]] :=
  Interpretation[
   Row@{a, "+", b},
   t
  ];

End[];
EndPackage[]
```

Examples

```
In[3151]:= A = tTens["A", tAxes[j, "N"], tBasis[X, "α"]];
           Q = tBasis["Q", "M"];

In[3285]:= $tReduce = True
Out[3285]= True

In[3286]:= J = tTens["J", tAxes[i, 2], tBasis[X, "α"]];
           G = tMul[J, tFlip[J, 2 → 1]]
Out[3287]= J J^{2,1}

In[3289]:= tDer[G, Q]
Out[3289]= ∂_Q X ∂_X J J^{2,1} + J ∂_Q J^{2,3 2,1}

In[2750]:= tDer[
            tDer[tTens[A, {5, 4, 5, 3}, tBasis[X, 5]], tBasis[Y, 5]],
            tBasis[Y, 5]
           ]
Out[2750]= ∂_Y X ∂_Y ∂_X A^{2→1 2→1} + ∂_Y^2 X ∂_X A
```

Potential Derivatives

With these rules in place we can write code to do this differentiation automatically, but for performance reasons and to show the procedure out we'll do up to the fourth derivatives by hand

Derivatives to Target

Basically we'll want

$$\nabla_Q V = \begin{pmatrix} \frac{\partial}{\partial Q_1} & \cdots & \frac{\partial}{\partial Q_m} \end{pmatrix} V$$

$$\nabla_{Q^2} V = \begin{pmatrix} \frac{\partial^2}{\partial Q_1 Q_1} V & \cdots & \frac{\partial^2}{\partial Q_1 Q_m} V \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial Q_1 Q_m} V & \cdots & \frac{\partial^2}{\partial Q_m Q_m} V \end{pmatrix}$$

and higher up to order 4.

We don't actually have these directly, though, so we'll need to build them using the chain rule a bunch from the components we do have.

As a simple initial example, we want

$$\nabla_Q V = \begin{pmatrix} \frac{\partial}{\partial Q_1} & \cdots & \frac{\partial}{\partial Q_m} \end{pmatrix} V$$

and we can look at this element wise as

$$(\nabla_Q V)_i = \frac{\partial}{\partial Q_i} V = \sum_j \frac{\partial x_j}{\partial Q_i} \frac{\partial}{\partial x_j} V$$

General Layout

We have basically two types of terms here:

$$\nabla_Q x = \begin{pmatrix} \frac{\partial x_1}{\partial Q_1} & \cdots & \frac{\partial x_n}{\partial Q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial Q_m} & \cdots & \frac{\partial x_n}{\partial Q_m} \end{pmatrix}$$

which is the Jacobian going from Cartesian coordinates to internal coordinate normal modes. We will also have higher order Jacobian type things.

$$\nabla_x V = \begin{pmatrix} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{pmatrix} V$$

which is the gradient of V with respect to the Cartesian coordinates. We'll also have higher order derivatives here.

I will denote higher derivatives by adding an exponent to the dependent variable, e.g.

$$\nabla_{Q^2} x = \begin{pmatrix} \frac{\partial}{\partial Q_1} \begin{pmatrix} \frac{\partial x_1}{\partial Q_1} & \cdots & \frac{\partial x_n}{\partial Q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial Q_m} & \cdots & \frac{\partial x_n}{\partial Q_m} \end{pmatrix} & \cdots & \frac{\partial}{\partial Q_m} \begin{pmatrix} \frac{\partial x_1}{\partial Q_1} & \cdots & \frac{\partial x_n}{\partial Q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial Q_m} & \cdots & \frac{\partial x_n}{\partial Q_m} \end{pmatrix} \end{pmatrix} =$$

$$\begin{pmatrix} \begin{pmatrix} \frac{\partial^2 x_1}{\partial Q_1 Q_1} & \cdots & \frac{\partial^2 x_n}{\partial Q_1 Q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 x_1}{\partial Q_1 Q_m} & \cdots & \frac{\partial^2 x_n}{\partial Q_1 Q_m} \end{pmatrix} & \cdots & \begin{pmatrix} \frac{\partial^2 x_1}{\partial Q_m Q_1} & \cdots & \frac{\partial^2 x_n}{\partial Q_m Q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 x_1}{\partial Q_m Q_m} & \cdots & \frac{\partial^2 x_n}{\partial Q_m Q_m} \end{pmatrix} \end{pmatrix}$$

is the 3D tensor of second derivatives of Cartesian coordinates with respect to pairs of internal coordinate normal modes.

Derivatives

A worthwhile thing to keep in mind: the number of terms in an expansion grows exponentially as the Bell Numbers. It's a fun little graph proof, if you want to see it.

Firsts

These are unchanged

$$\nabla_Q V = \nabla_Q X \nabla_X V$$

Seconds

These have two transpositions removed and rewritten as tensor contractions

$$\begin{aligned}\nabla_{Q^2} V &= \nabla_{Q^2} X \nabla_X V + (\nabla_Q X \langle 2 \rangle \nabla_Q X \nabla_{X^2} V)^{2:1} \\ &= \nabla_{Q^2} X \nabla_X V + \nabla_Q X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V\end{aligned}$$

Thirds

These still get gross to start, since we've got 5 elements

$$\begin{aligned}\nabla_{Q^3} V = \nabla_{Q^3} X \nabla_X V &+ (\nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \nabla_{X^2} V)^{3:1} \\ &+ \nabla_{X^2} Q \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V \\ &+ (\nabla_Q X \langle 3 \rangle \nabla_{Q^2} X \langle 2 \rangle \nabla_{X^2} V)^{2:1} + \left(\nabla_Q X \langle 3 \rangle (\nabla_Q X \langle 3 \rangle \nabla_Q X \nabla_{X^2} V)^{2:1}\right)^{2:1}\end{aligned}$$

we can simplify this by noting that, since $\nabla_{X^2} V$ is symmetric

$$\begin{aligned}(\nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \nabla_{X^2} V)^{3:1} &= (\nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V)^{3:1} \\ (\nabla_Q X \langle 3 \rangle \nabla_{Q^2} X \langle 2 \rangle \nabla_{X^2} V)^{2:1} &= (\nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V)^{3:2}\end{aligned}$$

and so we really only have one matrix multiplication to do

$$\nabla_{Q^2,Q} \nabla_{x^2} V = \nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V$$

giving us

$$\begin{aligned}\nabla_{Q^3} V = \nabla_{Q^3} X \nabla_X V &+ \nabla_{Q^2,Q} \nabla_{x^2} V \\ &+ (\nabla_{Q^2,Q} \nabla_{x^2} V)^{3:1} \\ &+ (\nabla_{Q^2,Q} \nabla_{x^2} V)^{3:2} + \nabla_Q X \langle 3 \rangle \nabla_Q X \langle 3 \rangle \nabla_Q X \langle 3 \rangle \nabla_{X^3} V\end{aligned}$$

Fourths

We need 15 elements here, but 9 of them are going to come from the $\nabla_{Q^2,Q} \nabla_{x^2} V$ term, so we'll do that first

$$\begin{aligned}\nabla_Q (\nabla_{Q^2,Q} \nabla_{x^2} V) &= \nabla_Q (\nabla_{Q^2} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V) \\ &= \nabla_{Q^3} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V \\ &\quad + (\nabla_{Q^2} X \langle 3 \rangle \nabla_{Q^2} X \langle 2 \rangle \nabla_{X^2} V)^{3:1} \\ &\quad + \left(\nabla_{Q^2} X \langle 3 \rangle (\nabla_Q X \langle 3 \rangle \nabla_Q X \nabla_{X^3} V)^{2:1}\right)^{3:1}\end{aligned}$$

$$
\begin{aligned}
&= \nabla_{Q^3} X\langle 2\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_{X^2} V \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_{Q^2} X\langle 2\rangle \, \nabla_{X^2} V)^{3:1} \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_{X^3} V)^{3:1} \\
&= \nabla_{Q^3} X\langle 2\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_{X^2} V \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_{Q^2} X\langle 2\rangle \, \nabla_{X^2} V)^{3:1} \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_{X^3} V)^{3:1}
\end{aligned}
$$

then when we plug in our transpositions from before we get

$$
\begin{aligned}
&\nabla_Q\left(\nabla_{Q^2,Q}\nabla_{x^2} V + (\nabla_{Q^2,Q}\nabla_{x^2} V)^{3:1} + (\nabla_{Q^2,Q}\nabla_{x^2} V)^{3:2}\right) = \\
&\nabla_Q(\nabla_{Q^2,Q}\nabla_{x^2} V) + \nabla_Q(\nabla_{Q^2,Q}\nabla_{x^2} V)^{4:2} + \nabla_Q(\nabla_{Q^2,Q}\nabla_{x^2} V)^{4:3}
\end{aligned}
$$

which gives us three terms & three transpositions, resulting in nine terms overall

Then we can easily do the first term

$$
\nabla_Q(\nabla_{Q^3} X \, \nabla_X V) = \nabla_{Q^4} X \, \nabla_X V + \nabla_Q X\langle 4\rangle \, \nabla_{Q^3} X\langle 2\rangle \, \nabla_{X^2} V
$$

and with a bit more annoyance we can get the final term

$$
\begin{aligned}
\nabla_Q(\nabla_Q X\langle 3\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_{X^3} V) &= \nabla_{Q^2} X\langle 3\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_{X^3} V \\
&\quad + \nabla_Q X\langle 4\rangle \, (\nabla_{Q^2} X\langle 3\rangle \, \nabla_Q X\langle 3\rangle \, \nabla_{X^3} V)^{2:1} \\
&\quad + \nabla_Q X\langle 4\rangle \, (\nabla_Q X\langle 4\rangle \, \nabla_{Q^2} X\langle 3\rangle \, \nabla_{X^3} V)^{2:1} \\
&\quad + \nabla_Q X\langle 4\rangle \left(\nabla_Q X\langle 4\rangle \, (\nabla_Q X\langle 4\rangle \, \nabla_Q X \nabla_{X^4} V)^{2:1}\right)^{2:1}
\end{aligned}
$$

we'll do some total term gathering in a bit, but we'll first note that by some rearrangements and symmetry in $\nabla_{X^4} V$ we can write

$$
\nabla_Q X\langle 4\rangle \left(\nabla_Q X\langle 4\rangle \, (\nabla_Q X\langle 4\rangle \, \nabla_Q X \nabla_{X^4} V)^{2:1}\right)^{2:1} = \nabla_Q X\langle 4\rangle \, \nabla_Q X\langle 4\rangle \, \nabla_Q X\langle 4\rangle \, \nabla_Q X\langle 4\rangle \, \nabla_{X^4} V
$$

Finally, we'll gather all the terms we have here, since we only have five distinct types

$$
\nabla_{Q^4}\nabla_X V = \nabla_{Q^4} X \, \nabla_X V
$$

$$
\begin{aligned}
\nabla_{Q^2\,Q^2}\nabla_{X^2} V &= (\nabla_{Q^2} X\langle 3\rangle \, \nabla_{Q^2} X\langle 2\rangle \, \nabla_{X^2} V)^{3:1} \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_{Q^2} X\langle 2\rangle \, \nabla_{X^2} V)^{3:1\,4:2} \\
&\quad + (\nabla_{Q^2} X\langle 3\rangle \, \nabla_{Q^2} X\langle 2\rangle \, \nabla_{X^2} V)^{3:1\,4:3}
\end{aligned}
$$

$$
\begin{aligned}
\nabla_{Q^3\,Q}\nabla_{X^2} V &= \nabla_Q X\langle 4\rangle \, \nabla_{Q^3} X\langle 2\rangle \, \nabla_{X^2} V \\
&\quad + \nabla_{Q^3} X\langle 2\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_{X^2} V \\
&\quad + (\nabla_{Q^3} X\langle 2\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_{X^2} V)^{4:2} \\
&\quad + (\nabla_{Q^3} X\langle 2\rangle \, \nabla_Q X\langle 2\rangle \, \nabla_{X^2} V)^{4:3}
\end{aligned}
$$

$$
\begin{aligned}
&= \nabla_{Q^3} X\langle 2\rangle \nabla_Q X\langle 2\rangle \nabla_{X^2} V \\
&\quad + (\nabla_{Q^3} X\langle 2\rangle \nabla_Q X\langle 2\rangle \nabla_{X^2} V)^{4:1} \\
&\quad + (\nabla_{Q^3} X\langle 2\rangle \nabla_Q X\langle 2\rangle \nabla_{X^2} V)^{4:2} \\
&\quad + (\nabla_{Q^3} X\langle 2\rangle \nabla_Q X\langle 2\rangle \nabla_{X^2} V)^{4:3}
\end{aligned}
$$

$$
\begin{aligned}
\nabla_{Q^2\,\mathrm{QQ}}\nabla_{X^3} V = &\ \nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V \\
&+ (\nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{3:1} \\
&+ (\nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{3:1\,4:2} \\
&+ (\nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{3:1\,4:3} \\
&+ (\nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{1:3} \\
&+ (\nabla_{Q^2} X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{1:3\,1:3}
\end{aligned}
$$

$$
\nabla_{\mathrm{QQQQ}}\nabla_{X^4} V = \nabla_Q X\langle 4\rangle \nabla_Q X\langle 4\rangle \nabla_Q X\langle 4\rangle \nabla_Q X\langle 4\rangle \nabla_{X^4} V
$$

Mixed Derivatives

We don't actually start out with stuff like $\nabla_{\mathrm{X}^4}\mathrm{V}$, instead having things like $\nabla_{q^2}\nabla_{x^2}\mathrm{V}$, and we need to account for that in the derivs. This will enter only in the cubic and quartic force terms, giving us stuff like

$$
\begin{aligned}
\nabla_Q \nabla_{X^2} V &= \nabla_Q q\, \nabla_q \nabla_{X^2} V \\
\nabla_{Q^2} \nabla_{X^2} V &= \nabla_{Q^2} q\, \nabla_q \nabla_{X^2} V + \nabla_Q q\langle 2\rangle \nabla_Q q\langle 2\rangle \nabla_{q^2} \nabla_{X^2} V
\end{aligned}
$$

then we need to back-convert all of our $\nabla_{X^3} V$-like terms into ones involving the mixed derivatives we start with

$$
\begin{aligned}
\nabla_Q \nabla_{X^2} V &= \nabla_Q X\, \nabla_{X^3} V \\
\nabla_{Q^2} \nabla_{X^2} V &= \nabla_{Q^2} X\, \nabla_{X^3} V + \nabla_Q X\langle 2\rangle \nabla_Q X\, \nabla_{X^4} V
\end{aligned}
$$

meaning some of our terms disappear/simplify out

Alternately if we're trying to translate unmixed derivatives into mixed ones we have

$$
\nabla_{Q^2} \nabla_{X^2} V = \nabla_Q X\langle 2\rangle \nabla_Q X\langle 1\rangle \nabla_{X^4} V + \nabla_{Q^2} X\, \nabla_{X^3} V
$$

Firsts & Seconds

Unchanged

Thirds

Here we just need to change up one term, giving us

$$\begin{aligned}\nabla_{Q^3} V = \nabla_{Q^3} X \nabla_X V + \nabla_{Q^2,Q} \nabla_{x^2} V \\ &+ (\nabla_{Q^2,Q} \nabla_{x^2} V)^{3:1} \\ &+ (\nabla_{Q^2,Q} \nabla_{x^2} V)^{3:2} + \nabla_Q X \langle 3 \rangle \nabla_Q X \langle 3 \rangle \nabla_Q \nabla_{X^2} V\end{aligned}$$

Fourths

Diagonal Elements

Original Vs. New Stuff

I wanted to first express things in Cartesian modes and then transform, but things are weird...so we're going to work through the original imp. and the new one to see where things differ.

Coordinate Definitions

We'll note that the two normal mode coordinate sets are equivalent to first order, i.e.

$$\nabla_Q q = I$$

so redoing the fourth derivative stuff we define our OG terms in terms of their Cartesian counterparts

$$\begin{aligned}\nabla_q V &= \nabla_q X \langle 2 \rangle \nabla_X V \\ \nabla_{q^2} V &= \nabla_q X \langle 2 \rangle \nabla_q X \langle 2 \rangle \nabla_{X^2} V \\ \nabla_{q^3} V &= \nabla_q X \langle 3 \rangle \nabla_q X \langle 3 \rangle \nabla_q X \langle 3 \rangle \nabla_{X^3} V \\ \nabla_{q^4} V &= \nabla_q X \langle 4 \rangle \nabla_q X \langle 4 \rangle \nabla_q X \langle 4 \rangle \nabla_q X \langle 4 \rangle \nabla_{X^4} V\end{aligned}$$

and the relationships between the modes

$$\begin{aligned}\nabla_{Q^2} q &= \nabla_Q R \nabla_{R^2} X \nabla_X q \\ &= \nabla_{Q^2} X \nabla_X q \\ \nabla_{Q^3} q &= \nabla_{Q^3} X \nabla_X q\end{aligned}$$

Original

$$\begin{aligned}Q_2 V_2 &= \nabla_{Q^2} X \langle 3 \rangle \nabla_{Q^2} X \langle 2 \rangle \nabla_{X^2} V \\ \nabla_{Q^2\ Q^2} \nabla_{X^2} V &= (Q_2 V_2)^{3:1} + (Q_2 V_2)^{3:1,4:2} + (Q_2 V_2)^{3:1,4:3} \\ Q_3 V_2 &= \nabla_{Q^3} X \langle 2 \rangle \nabla_Q X \langle 2 \rangle \nabla_{X^2} V \\ \nabla_{Q^3\ Q} \nabla_{X^2} V &= Q_3 V_2 + (Q_3 V_2)^{4:1} + (Q_3 V_2)^{4:2} + (Q_3 V_2)^{4:3}\end{aligned}$$

$$\begin{aligned}V_3 &= \nabla_Q X \langle 4 \rangle \nabla_{Q^2} X \langle 3 \rangle \nabla_Q q \langle 1 \rangle \nabla_q \nabla_{X^2} V \\ &= \nabla_Q X \langle 4 \rangle \nabla_{Q^2} X \langle 3 \rangle \nabla_Q X \nabla_{X^3} V \\ \nabla_{Q^2\,\mathrm{QQ}} \nabla_{X^3} V &= V_3^{1:3,4:1} + V_3^{1:3,4:2} + V_3^{4:3} + V_3^{4:2} + V_3^{4:3,1:2}\end{aligned}$$

$$\nabla_{\mathrm{QQQQ}} \nabla_{X^4} V = \nabla_Q X \langle 4 \rangle \nabla_Q X \langle 4 \rangle \nabla_Q X \langle 4 \rangle \nabla_Q X \langle 4 \rangle \nabla_{X^4} V + (\nabla_Q X \langle 4 \rangle \nabla_Q X \langle 4 \rangle \nabla_{Q^2} q \langle 1 \rangle \nabla_q \nabla_{X^2} V)^{1:4,1:4}$$

and then when we have the "V4" term involving the mixed derivatives

$$(\nabla_Q X\langle 4\rangle \nabla_Q X\langle 4\rangle \nabla_{Q^2} q\langle 1\rangle \nabla_q \nabla_{X^2} V)^{1:4,1:4}$$

this is really

$$\begin{aligned}(\nabla_Q X\langle 4\rangle \nabla_Q X\langle 4\rangle \nabla_{Q^2} X \nabla_{X^3} V)^{1:4,1:4} &= (\nabla_Q X\langle 4\rangle \nabla_{Q^2} X\langle 3\rangle \nabla_Q X \nabla_{X^3} V)^{1:4} \\ &= {V_3}^{1:4}\end{aligned}$$

New

now we have

$$\begin{aligned}Q_2 V_2 &= \nabla_{Q^2} q\langle 3\rangle \nabla_{Q^2} q\langle 2\rangle \nabla_{q^2} V \\ &= (\nabla_{Q^2} X \nabla_X q)\langle 3\rangle (\nabla_{Q^2} X \nabla_X q)\langle 2\rangle \nabla_q X\langle 2\rangle \nabla_q X\langle 2\rangle \nabla_{X^2} V \\ &= \nabla_{Q^2} X\langle 3\rangle \nabla_{Q^2} X\langle 3\rangle \nabla_{X^2} V\end{aligned}$$

$$\nabla_{Q^2}\, {}_{Q^2}\nabla_{q^2} V = (Q_2 V_2)^{3:1} + (Q_2 V_2)^{3:1,4:2} + (Q_2 V_2)^{3:1,4:3}$$

$$\begin{aligned}Q_3 V_2 &= \nabla_{Q^3} q\langle 2\rangle \nabla_Q q\langle 2\rangle \nabla_{q^2} V \\ &= (\nabla_{Q^3} X \nabla_X q)\langle 2\rangle \nabla_q X\langle 2\rangle \nabla_q X\langle 2\rangle \nabla_{X^2} V \\ &= \nabla_{Q^3} X\langle 2\rangle \nabla_Q X\langle 2\rangle \nabla_{X^2} V\end{aligned}$$

$$\nabla_{Q^3}\, {}_Q\nabla_{q^2} V = Q_3 V_2 + (Q_3 V_2)^{4:1} + (Q_3 V_2)^{4:2} + (Q_3 V_2)^{4:3}$$

$$\begin{aligned}V_3 &= \nabla_Q q\langle 3\rangle \nabla_{Q^2} q\langle 2\rangle \nabla_Q q\langle 1\rangle \nabla_{q^3} V \\ &= \nabla_{Q^2} q\langle 2\rangle \nabla_q X\langle 3\rangle \nabla_q X\langle 3\rangle \nabla_q X\langle 3\rangle \nabla_{X^3} V \\ &= \nabla_{Q^2} X\langle 2\rangle \nabla_Q X\langle 3\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V \\ &= (\nabla_Q X\langle 3\rangle \nabla_{Q^2} X\langle 2\rangle \nabla_Q X\langle 3\rangle \nabla_{X^3} V)^{3:1} \\ &= ({V_3}_{,\text{old}})^{3:1}\end{aligned}$$

$$\nabla_{Q^2\,\mathrm{QQ}} \nabla_{X^3} V = {V_3}^{4:1} + {V_3}^{4:2} + {V_3}^{3:1,4:3} + {V_3}^{4:2} + {V_3}^{4:3,1:2} + {V_3}^{3:1,1:4}$$

$$\begin{aligned}\nabla_{\mathrm{QQQQ}} \nabla_{q^4} V &= \nabla_Q q\langle 4\rangle \nabla_Q q\langle 4\rangle \nabla_Q q\langle 4\rangle \nabla_Q q\langle 4\rangle \nabla_{q^4} V \\ &= \nabla_{q^4} V\end{aligned}$$

and so in this vein when we had

G-Matrix Derivatives

Helpful Notes

When reading *Wilson, Decius, & Cross* (WDC) the authors introduce some odd terminology at least according to modern notation. First off, they define their internal coordinates by S_i which, like, fine. I'm gonna use r_i, because that's more in line with that we do.

Then given a reference structure which I'll call $\boldsymbol{S}_e$ and whose existence they allow to be implicit, they approximate the internals up to first order by Cartesians by an equation that they write as

$$S_t = \sum_{i=1}^{N} B_{ti}\,\xi_i \;,\; t = 1,\, 2,\, \ldots,\, M$$

where N and M are the number of Cartesian coordinates and number of respective internal coordinates

Here we have ξ_i being the Cartesian displacement. I'd usually call it Δx_i. And the B term is just the derivative of the internal with respect to the Cartesians. This can be much stated more cleanly by saying that we'll *represent* S_t as

$$S_t\colon \left(\frac{\partial S_t}{\partial x_1}\Delta x_1,\; \frac{\partial S_t}{\partial x_2}\Delta x_2,\; \ldots,\; \frac{\partial S_t}{\partial x_{3N}}\Delta x_{3N}\right) = \left(\frac{\partial S_t}{\partial x_i}\Delta x_i\right)_{i=1,\ldots}$$

This is saying that we'll linearly approximate our internal as a Cartesian displacement. We won't actually use this definition to evaluate the our internals, but these derivatives are important as this is how we get the B matrix that is implicitly referred to in the text

$$B = \begin{pmatrix} \frac{\partial S_1}{\partial x_1} & \frac{\partial S_1}{\partial x_2} & \cdots & \frac{\partial S_1}{\partial x_N} \\ \frac{\partial S_2}{\partial x_1} & \frac{\partial S_2}{\partial x_2} & & \\ \vdots & & \ddots & \vdots \\ \frac{\partial S_M}{\partial x_1} & & \cdots & \frac{\partial S_M}{\partial x_N} \end{pmatrix}$$

For anyone who's taken intro calc or ever transformed from Cartesian to polar coordinates to do an integral, this is just the Jacobian matrix and I'm not sure why *WDC* don't refer to it as such.

Returning to the notation that I prefer (and introducing a slightly odd gradient notation), we have

$$\nabla_{\mathrm{X}}\mathrm{R} = \begin{pmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_N} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & & \\ \vdots & & \ddots & \vdots \\ \frac{\partial r_M}{\partial x_1} & & \cdots & \frac{\partial r_M}{\partial x_N} \end{pmatrix}$$

Next, in this notation we're given that the G-matrix is defined as

$$G_{ij} = \sum_{\alpha=1}^{N} \frac{1}{m_\alpha}\,\frac{\partial}{\partial x_\alpha} r_i\,\frac{\partial}{\partial x_\alpha} r_j$$

This is all well and good, but we can write this in a more intuitive way. Letting

$$\nabla_{\mathrm{X}} r_i = \left(\frac{\partial r_i}{\partial x_\alpha}\Delta x_\alpha\right)_{\alpha=1,\ldots,3N}$$

we'll define a mass-weighted form of this by

$$\nabla_{\mathrm{Y}} r_i = \left(\frac{1}{\sqrt{m_\alpha}} \frac{\partial r_i}{\partial x_\alpha} \Delta x_\alpha \right)_{\alpha=1,\ldots,3N}$$

the notation for which is motivated by the mass-weighted Coordinate transformation

$$y_\alpha = \sqrt{m_\alpha}\ x_\alpha$$

taking all of this together we get

$$G_{ij} = \nabla_{\mathrm{Y}} r_i \cdot \nabla_{\mathrm{Y}} r_j$$

This type of definition of a matrix element ($\text{vector}_i \cdot \text{vector}_j$) tells us that our matrix can be constructed by a matrix multiplication. In this case we have

$$G = (\nabla_{\mathrm{Y}} \mathrm{R})^{\mathrm{T}} (\nabla_{\mathrm{Y}} \mathrm{R})$$

where we can build $\nabla_{\mathrm{Y}} \mathrm{R}$ by

$$\nabla_{\mathrm{Y}} \mathrm{R} = (\nabla_{\mathrm{X}} \mathrm{R})\, \mathrm{M}^{-\frac{1}{2}}$$

with

$$\mathrm{M} = \begin{pmatrix} m_1 & & & \\ & m_2 & & \\ & & \ddots & \\ & & & m_{3N} \end{pmatrix}$$

GF Analysis

one fun thing to note is that in our usual construction, we find normal modes and frequencies by finding the eigenvalue and eigenvectors of GF, where F is the Hessian. Since both F and G and real-symmetric, this is equivalent to finding the *generalized* eigenvectors of F and G^{-1}, i.e. solving

$$\mathrm{F}\, v = \lambda \mathrm{G}^{-1}\, v$$

which can be done nicely by **scipy.linalg.eigh**

Note that this means that

$$\mathrm{G}\mathrm{V}^{-1}\, \mathrm{F}\mathrm{V} = \Lambda$$

where V are the normal modes

Derivatives

Since we use it so often, we let the mass-weighted Jacobian be

$$\mathrm{J} = \nabla_{\mathrm{Y}} \mathrm{R}$$

With this, the G matrix is given

$$G = J^T J$$

Next let's say we want the derivatives of the G matrix with respect to a new set of coordinates, $\{Q_n\}$.

Going back to prior notes on tensor derivates, we had

$$\nabla_Q V = \nabla_Q x \, \nabla_x V$$

where V was our potential function in Cartesian coordinates, Q was our internal coordinate set, and $\odot$ represents a generalized matrix multiplication to tensors (implemented in **numpy** via **tensordot**).

Correspondingly since J is defined in our R coordinates, this means that

$$\nabla_Q J = \nabla_Q R \, \nabla_R J$$

the $\nabla_Q R$ term is the Jacobian for going from the Q coordinate set to the R coordinate set. In the special case the Q_i are defined as linear combinations of the R coordinates defined by some transformation matrix L, this reduces to being simply

$$\nabla_Q R = L^{-1}$$

Next we have the $\nabla_R J$ term, a three dimensional tensor whose elements look like

$$(\nabla_R J)_{ijk} = \frac{\partial}{\partial r_i}\left(\frac{\partial r_j}{\partial x_k}\right) = \frac{\partial r_j^2}{\partial r_i \, \partial x_k} = \frac{\partial}{\partial x_k}\left(\frac{\partial r_j}{\partial r_i}\right) = \frac{\partial r_j}{\partial x_k}\,\delta_{ij}$$

which means this is just a nice block-diagonal tensor. This fact can be convenient when implementing this.

G-matrix Derivatives

First

We'll try to rearrange

$$\begin{aligned}\nabla_Q (J^T J) &= (\nabla_Q J^T)\, J + \left(J^T (\nabla_Q J)^{2:1}\right)^{2:1} \\ &= \nabla_Q Y \, \nabla_Y J^T \, J + \nabla_Q Y \, \langle 3 \rangle \, J^T \, \nabla_Y J\end{aligned}$$

and just expanding to find the most stable way to make these we expand that first term

$$\begin{aligned}\nabla_Y J &= \nabla_Y (\nabla_Y Q) \\ &= \nabla_{Y^2} Q \\ &= \nabla_Y (\nabla_Y R \, \nabla_R Q) \\ &= \nabla_{Y^2} R \, \nabla_R Q + \nabla_Y R \, \nabla_Y R \, \nabla_{R^2} Q \\ &= \nabla_{Y^2} R \, \nabla_R Q \\ \nabla_R Q &= \nabla_R Y \, \nabla_Y Q \\ \nabla_Q Y \, \nabla_Y J^T \, J &= \nabla_Q Y \, \langle 2 \rangle \, (\nabla_{Y^2} R \, \nabla_R Y \, \nabla_Y Q) \, \nabla_Y Q\end{aligned}$$

and since we never make use of something that looks like $\nabla_{\mathrm{Y}} \mathrm{Q}\, \nabla_{\mathrm{Q}} \mathrm{Y}$ we should be fine. For the next term we have

$$\nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \mathrm{J}^{\mathrm{T}}\, \nabla_{\mathrm{Y}} \mathrm{J} = \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \nabla_{\mathrm{Y}} \mathrm{Q}\, (\nabla_{\mathrm{Y}^2} \mathrm{R}\, \nabla_R \mathrm{Y}\, \nabla_{\mathrm{Y}} \mathrm{Q})$$

and again this also should be fine

Second

Then for the second derivatives

$$\begin{aligned}
\nabla_{\mathrm{Q}} Y\, \nabla_{\mathrm{Y}} \mathrm{J}^{\mathrm{T}}\, J + \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \mathrm{J}^{\mathrm{T}}\, \nabla_{\mathrm{Y}} J) &= \nabla_{\mathrm{Q}} (\nabla_{\mathrm{Q}} Y\, \nabla_{\mathrm{Y}} J^{\mathrm{T}}\, J) + \nabla_{\mathrm{Q}} (\nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \mathrm{J}^{\mathrm{T}}\, \nabla_{\mathrm{Y}} J) \\
&= \nabla_{\mathrm{Q}^2} Y\, \nabla_{\mathrm{Y}} J^{\mathrm{T}}\, J + \nabla_{\mathrm{Q}} Y\, \langle 2 \rangle\, \nabla_{\mathrm{Q}} Y\, \nabla_{\mathrm{Y}^2} J^{\mathrm{T}}\, J + \nabla_{\mathrm{Q}} Y\, \langle 2 \rangle\, \nabla_{\mathrm{Q}} Y\, \langle 3 \rangle\, \nabla_{\mathrm{Y}} J^{\mathrm{T}}\, \nabla_{\mathrm{Y}} \\
&+ \nabla_{\mathrm{Q}^2} \mathrm{Y}\, \langle 3 \rangle\, \mathrm{J}^{\mathrm{T}}\, \nabla_{\mathrm{Y}} J + \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 2 \rangle\, \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \nabla_{\mathrm{Y}} \mathrm{J}^{\mathrm{T}}\, \nabla_{\mathrm{Y}} J + \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \nabla_{\mathrm{Q}} \mathrm{Y}\, \langle 3 \rangle\, \mathrm{J}^{\mathrm{T}}\, \nabla
\end{aligned}$$

most things are straightforward, but worth noting that

$$\begin{aligned}
\nabla_{\mathrm{Q}}\, Y &= \nabla_Q R\, \nabla_R\, Y \\
\nabla_{Q^2}\, Y &= \nabla_{Q^2} R\, \nabla_R\, Y + \nabla_Q R\, \langle 2 \rangle\, \nabla_Q R\, \nabla_{R^2}\, Y \\
&= \nabla_Q R\, \langle 2 \rangle\, \nabla_Q R\, \nabla_{R^2}\, Y \\
\nabla_{\mathrm{Y}^2} J &= \nabla_Y (\nabla_Y J) \\
&= \nabla_Y (\nabla_{Y^2} R\, \nabla_R Q) \\
&= \nabla_{Y^3} R\, \nabla_R Q + \nabla_Y R\, \nabla_{R^2} Q \\
&= \nabla_{Y^3} R\, \nabla_R Q
\end{aligned}$$

i.e.

$$\frac{\partial y_j}{\partial Q_i} = \sum_\alpha \frac{\partial R_\alpha}{\partial Q_i}\, \frac{\partial R_\alpha}{\partial Q_i}$$

Dipole Derivatives

We need to transform from Cartesian coords to internals again...but this time we've got different mixed derivatives to deal with.

The first derivatives are straightforward (being wholly in Cartesian coordinates)

$$\nabla_Q \mu_\alpha = \nabla_Q X\, \nabla_X \mu_\alpha$$

The second derivatives are not bad, even though we start with $\nabla_q \nabla_X \mu_\alpha$

$$\begin{aligned}
\nabla_Q \nabla_X \mu_\alpha &= \nabla_Q q\, \nabla_q \nabla_X \mu_\alpha \\
\nabla_{Q^2} \mu_\alpha &= \nabla_Q X\, \langle 2 \rangle\, \nabla_Q \nabla_X \mu_\alpha
\end{aligned}$$

note that this can be easily proved by the fact that $\nabla_Q \nabla_X \mu_\alpha = (\nabla_X \nabla_Q \mu_\alpha)^{\mathrm{T}}$ *by Clairaut's theorem or whatever*

The third derivatives require a bit of care, as we start with $\nabla_{q^2} \nabla_X \mu_\alpha$ and so we have

$$\nabla_{Q^2} \nabla_X \mu_\alpha = \nabla_Q (\nabla_Q q\, \nabla_q \nabla_X \mu_\alpha)$$

$$= \nabla_{Q^2} q \, \nabla_q \nabla_X \mu_\alpha + \nabla_Q q \, \langle 2 \rangle \, \nabla_Q q \, \nabla_{q^2} \nabla_X \mu_\alpha$$

and then finally

$$\nabla_{Q^3} \mu_\alpha = \nabla_Q X \, \langle 3 \rangle \, \nabla_{Q^2} \nabla_X \mu_\alpha$$

Coriolis Zeta Constants

To be able to compare to pure Cartesian coordinate calculations, it is useful to be able to add the effects of Coriolis coupling to our calculations, even though we'd prefer to work purely in internal coordinates. To make this work, we note that the according to Barone and friends, the Coriolis term comes in at second order

$$H^{(2)} = \frac{1}{24} \nabla_{Q^4} V \; \mathrm{QQQQ} + \sum_{\tau \in \{x \; y, \; z\}} B_e^\tau \, \zeta_{ij}^\tau \, \zeta_{kl}^\tau \left(\frac{\omega_j \, \omega_l}{\omega_i \, \omega_k} \right) q_i \, p_j \, q_k \, p_l$$

$$\hat{H}^{(2)} = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} \frac{1}{24} k_{ijkl} q_i q_j q_k q_l$$

$$+ \sum_{\tau=x,y,z} B_e^\tau \zeta_{ij}^\tau \zeta_{kl}^\tau \left(\frac{\omega_j \omega_l}{\omega_i \omega_k} \right) q_i \hat{p}_j q_k \hat{p}_l, \qquad (1)$$

and then the question is simply how to evaluate the ζ_{ij}^τ terms. Those are defined as

$$\zeta_{ij}^z = \sum_n \frac{\partial x_n}{\partial Q_i} \frac{\partial y_n}{\partial Q_j} - \frac{\partial x_n}{\partial Q_j} \frac{\partial y_n}{\partial Q_i}$$

which can be rewritten as

$$\zeta_{ij}^z = \sum_n \left(\frac{\partial x_n}{\partial Q_i} + \frac{\partial y_n}{\partial Q_i} \right) \left(- \frac{\partial x_n}{\partial Q_j} + \frac{\partial y_n}{\partial Q_j} \right) - \left(\frac{\partial y_n}{\partial Q_i} \frac{\partial y_n}{\partial Q_j} - \frac{\partial x_n}{\partial Q_i} \frac{\partial x_n}{\partial Q_j} \right)$$

Why is this nice? Well it makes the operator form of this clearer. Letting $\tau \in \{a, b, c\}$ and defining two transpositions of the Jacobian like

$$J_{ni\tau} = \frac{\partial \tau_n}{\partial_{Q_i}}$$

$$J^{\mathrm{T}}{}_{n\tau i} = \frac{\partial \tau_n}{\partial_{Q_i}}$$

we can write

$$\zeta_{ij}^\tau = \sum_n J_{ni} \, \varepsilon^{(\alpha\beta)} \, J^{\mathrm{T}}{}_{jn}$$

and $\varepsilon^{(\alpha\beta)}$ is the 2D Levi-Civita tensor (antisymmetric tensor) defined for $\alpha, \beta \in \{a, b, c\} \backslash \{\tau\}$. And so then we can build the entirety of ζ in one go by

$$\zeta = \sum_n J_n \, \varepsilon \, J^T{}_n$$

where ε is now the proper 3D Levi-Civita tensor.

Unit-wise, things are kinda annoying. We note that we expect J to be the matrix of derivatives of the mass-weighted Cartesian coordinates with respect to the dimensionless normal modes. If this is the case, we know that

$$J^{-1\,T} \, J^{-1} = G$$

and so all of these products should have units of energy. If everything is internally consistent and in atomic units, specifically this energy unit is E_h. That means we expect ζ to have units of $1 / E_h$...but that doesn't work with the expression

$$\sum_\tau B_e^\tau \, \zeta_{ij}^\tau \, \zeta_{kl}^\tau \left(\frac{\omega_j \, \omega_l}{\omega_i \, \omega_k} \right) p_i \, q_j \, p_l \, q_k$$

which is supposed to be the Coriolis correction in $H^{(2)}$...because B_e^τ should *also* have units of energy and this would end up giving something with units of $1 / E_h$.

Potential-Like Terms

Come out of the how the inertia tensor transforms in internal coordinates. Add a brief overview.

The Watson Term

Add some notes about how you need the U term to get a ZPE that agrees with Gaussian in Cartesian displacements (linearized internals).

The V' Term

Somewhat annoying to show, but this is basically comes from

$$V'(Q)_{ij} = -\frac{\hbar^2}{2} \left(-\frac{1}{4} |g|^{-\frac{1}{4}} \frac{\partial}{\partial q_i} \left(|g|^{-\frac{3}{4}} \, G_{ij} \, \frac{\partial |g|}{\partial q_j} \right) \right)$$

where

$$g = \nabla_Q Y (\nabla_Q Y)^T$$

which of course looks terrible...but using Jacobi's formula we can write

$$\frac{\partial}{\partial q_i} |g| = \mathrm{Tr}\left(\mathrm{Adj}(g) \, \frac{\partial g}{\partial q_i} \right)$$

which shows the path one can take to getting these elements.

From Pickett, this can be re-expressed to fourth order using a Taylor series to give

$$V' = \frac{\hbar^2}{8} \sum_{ij} \frac{\partial G_{ij}}{\partial q_i} \frac{\partial \gamma}{\partial q_j} + G_{ij} \left(\frac{\partial^2 \gamma}{\partial q_i q_j} + \frac{1}{4} \frac{\partial \gamma}{\partial q_i} \frac{\partial \gamma}{\partial q_j} \right)$$

where, calling the mass-weighted coordinates Y

$$G = \nabla_Q Y (\nabla_Q Y)^T$$
$$\gamma = \ln \frac{|I_0|}{|G|}$$

where I^0 is the moment of inertia tensor. Ignoring the difficulty of getting the γ derivatives for the moment (we'll come back to it), we'll note that we can partially factorize this as

$$\frac{\partial G_{ij}}{\partial q_i} \frac{\partial \gamma}{\partial q_j} + G_{ij} \left(\frac{\partial^2 \gamma}{\partial q_i q_j} + \frac{1}{4} \frac{\partial \gamma}{\partial q_i} \frac{\partial \gamma}{\partial q_j} \right) = \left(\frac{\partial G_{ij}}{\partial q_i} + \frac{1}{4} G_{ij} \frac{\partial \gamma}{\partial q_i} \right) \frac{\partial \gamma}{\partial q_j} + G_{ij} \frac{\partial^2 \gamma}{\partial q_i q_j}$$

which is different from the product-rule type factorization that we could also try

$$\frac{\partial G_{ij}}{\partial q_i} \frac{\partial \gamma}{\partial q_j} + G_{ij} \left(\frac{\partial^2 \gamma}{\partial q_i q_j} + \frac{1}{4} \frac{\partial \gamma}{\partial q_i} \frac{\partial \gamma}{\partial q_j} \right) = \frac{\partial}{\partial q_i} \left(G_{ij} \frac{\partial \gamma}{\partial q_j} \right) = \frac{\partial G_{ij}}{\partial q_i} \frac{\partial \gamma}{\partial q_j} + G_{ij} \frac{1}{4} \frac{\partial \gamma}{\partial q_i} \frac{\partial \gamma}{\partial q_j}$$

on the other hand, that first form is potentially more useful, though, because assuming we have the tensor $\nabla_Q G$ (which we need for the perturbation theory anyway) we can write the total sum as

$$\sum_{ij} \frac{\partial G_{ij}}{\partial q_i} \frac{\partial \gamma}{\partial q_j} + G_{ij} \left(\frac{\partial^2 \gamma}{\partial q_i q_j} + \frac{1}{4} \frac{\partial \gamma}{\partial q_i} \frac{\partial \gamma}{\partial q_j} \right) = \sum_i (\nabla_Q G)_{ii} \nabla_Q \gamma + G \odot_2 \nabla_{Q^2} \gamma + \frac{1}{4} (G \nabla_Q \gamma)^{\mathrm{T}} \nabla_Q \gamma$$

and now all that's left is figuring out how to evaluate

$$(\nabla_Q \gamma)_i = \frac{\partial}{\partial q_i} \ln \frac{|I_0|}{|G|}$$
$$(\nabla_{Q^2} \gamma)_{ij} = \frac{\partial^2}{\partial q_i \partial q_j} \ln \frac{|I_0|}{|G|}$$

and now the Jacobi formula will be super useful.

$$\begin{aligned}
\frac{\partial}{\partial q_i} \ln \frac{|I_0|}{|G|} &= \frac{\partial}{\partial q_i} \ln |I_0| - \frac{\partial}{\partial q_i} \ln |G| \\
&= \frac{1}{|I_0|} \frac{\partial}{\partial q_i} |I_0| - \frac{1}{|G|} \frac{\partial}{\partial q_i} |G| \\
\frac{\partial^2}{\partial q_j \partial q_i} \ln \frac{|I_0|}{|G|} &= \frac{\partial}{\partial q_j} \left(\frac{1}{|I_0|} \frac{\partial}{\partial q_i} |I_0| - \frac{1}{|G|} \frac{\partial}{\partial q_i} |G| \right) \\
&= -\frac{1}{|I_0|^2} \frac{\partial |I_0|}{\partial q_i} \frac{\partial |I_0|}{\partial q_j} + \frac{1}{|I_0|} \frac{\partial^2 |I_0|}{\partial q_j \partial q_i} - \left(-\frac{1}{|G|^2} \frac{\partial |G|}{\partial q_i} \frac{\partial |G|}{\partial q_j} + \frac{1}{|G|} \frac{\partial^2 |G|}{\partial q_j \partial q_i} \right)
\end{aligned}$$

We'll focus on the inertia tensor for now, but note that everything will work the same for G. For the first derivatives we have

$$\frac{\partial}{\partial q_i}\left|I^0\right| = \mathrm{Tr}\left(\mathrm{Adj}(I_0)\ \frac{\partial I_0}{\partial q_i}\right)$$

where Adj is the *adjugate matrix*. In general, it's the transpose of the *cofactor matrix*, but since I^0 is invertible we have

$$\mathrm{Adj}(I_0) = I_0^{-1}\ |I_0|$$

and looking towards the second derivatives we note that

$$\begin{aligned}\frac{\partial}{\partial q_j}\mathrm{Tr}(A) &= \frac{\partial}{\partial q_j}\sum_k A_{kk} \\ &= \sum_k \frac{\partial}{\partial q_j} A_{kk}\end{aligned}$$

and so we get

$$\begin{aligned}\frac{\partial^2}{\partial q_j\,\partial q_i}|I_0| &= \sum_k \frac{\partial}{\partial q_j}\left(\mathrm{Adj}(I_0)\ \frac{\partial I_0}{\partial q_i}\right)_{kk} \\ &= \sum_k \frac{\partial}{\partial q_j}\left(\mathrm{Adj}(I_0)_k\ \frac{\partial (I_0)_k}{\partial q_i}\right) \\ &= \sum_k \frac{\partial \mathrm{Adj}(I_0)_k}{\partial q_j}\ \frac{\partial (I_0)_k}{\partial q_i} + \mathrm{Adj}\left(I^0\right)_k\ \frac{\partial^2 (I_0)_k}{\partial q_i\,\partial q_j} \\ &= (\nabla_Q I_0)_i \odot_2 (\nabla_Q \mathrm{Adj}(I_0))_j + \mathrm{Adj}\left(I^0\right) \odot_2 (\nabla_{Q^2} I_0)_{ij}\end{aligned}$$

then

$$\begin{aligned}\frac{\partial \mathrm{Adj}(I_0)_k}{\partial q_j} &= \frac{\partial}{\partial q_j}\left(I_0^{-1}\right)_k |I_0| \\ &= |I_0|\ \frac{\partial}{\partial q_j}\left(I_0^{-1}\right)_k + \left(I_0^{-1}\right)_k\ \frac{\partial}{\partial q_j}|I_0| \\ &= |I_0|\ \left(\nabla_Q I_0^{-1}\right)_{jk} + (\nabla_Q |I_0|)_j \left(I_0^{-1}\right)_k \\ \nabla_Q \mathrm{Adj}(I_0) &= |I_0|\ \nabla_Q I_0^{-1} + (\nabla_Q |I_0|)\ I_0^{-1}\end{aligned}$$

(where we made use of the fact that since I_0 is symmetric its inverse is too). Finally when worrying about how to do the inertia tensor derivatives we have

$$\begin{aligned}\frac{\partial I_0}{\partial q_i} &= (\nabla_Q Y)_i\ \nabla_Y I_0 \\ \frac{\partial^2 I_0}{\partial q_i\,\partial q_j} &= \nabla_Q((\nabla_Q Y)_j\ \nabla_Y I_0)_i \\ &= \nabla_Q Y\ ((\nabla_Q Y)_j\ \nabla_Y I_0)\end{aligned}$$

$$= (\nabla_{Q^2} Y)_{ij} \nabla_Y I_0 + (\nabla_Q Y)_j (\nabla_Q Y)_i \nabla_{Y^2} I_0$$

then getting the Cartesian derivatives of the inertia tensor, for every mass-weighted atomic position vector we get a derivative tensor like

$$\begin{aligned}\frac{\partial}{\partial y_n} I_0 &= \frac{\partial}{\partial y_n} \sum_m \mathbb{I}_3 (y_m \cdot y_m) - y_m \otimes y_m \\ &= \frac{\partial}{\partial y_n} (\mathbb{I}_3 \; y_n \cdot y_n - y_n \otimes y_n) \\ &= 2 \; y_n (\mathbb{I}_3 \otimes \mathbb{I}_3) - (y_n \otimes \mathbb{I}_3 + \mathbb{I}_3 \otimes y_n)\end{aligned}$$

where we've got one non-standard notation. The term $y_n \otimes \mathbb{I}_3$ should be read to mean the vector $(y_n \otimes (\mathbb{I}_3)_i)_i$. I chose to write it this way because this is easy to implement in modern numerical programming languages. Your milage may vary on this, though. Worth also noting that if your language supports high-dimensional transpositions, we have

$$y_n \otimes \mathbb{I}_3 = \mathbb{I}_3 \otimes y_n^{(1,\,3,\,2)}$$

where the superscripted (1, 3, 2) indicates that we transposed the final two axes of the tensor.

With this out of the way, the second derivative tensor is actually pretty straightforward and actually has no dependence on the coordinates themselves (this because the products & moments of inertia are quadratic terms)

$$\frac{\partial}{\partial y_n \, \partial y_m} I_0 = \Big(2 \; (\mathbb{I}_3 \otimes \mathbb{I}_3) - \Big(\; (\mathbb{I}_3 \otimes (\mathbb{I}_3)_i)_i + \;\; (\mathbb{I}_3 \otimes (\mathbb{I}_3)_i)_i^{(1,2,4,3)} \Big) \Big) \, \delta_{nm}$$

where again, totally weird notation, but $(\mathbb{I}_3 \otimes (\mathbb{I}_3)_i)_i$ is the same kind of vector of outer products as before and the (1,2,4,3) exponent just means transpose the final two axes in the tensor. Not gonna deny that it's not intuitive notation, but it's easy to implement and that's what really matters.

The final component that we need to think about how to handle is

$$\frac{\partial}{\partial q_j} \Big(\big(I_0^{-1} \big) \Big)_j = \Big(\nabla_Q \big(I_0^{-1} \big) \Big)_{jj}$$

Happily, using the formula for the derivative of a matrix inverse from The Matrix Cookbook we can actually formulate this without too much trouble. They give use

$$\frac{\partial \, Y^{-1}}{\partial x} = -\big(Y^{-1} \big) \, \frac{\partial \, Y}{\partial x} \; Y^{-1}$$

or explicitly

$$\frac{\partial \, I_0^{-1}}{\partial y_n} = -\big(I_0^{-1} \big) \, \frac{\partial I_0}{\partial y_n} \; I_0^{-1}$$

and so generalizing to the gradient with respect to the normal modes we have

$$\nabla_Q\left(I_0^{-1}\right)=-\left(I_0^{-1}\right)\langle 2\rangle\,\nabla_Q I_0\, I_0^{-1}$$

which justifies all of the effort that we put into writing down those expressions for $\frac{\partial}{\partial y_n} I_0$ and friends

Tests

```
In[1548]:= Outer[Times, IdentityMatrix[3], IdentityMatrix[3]]
```

```
Out[1548]= {{{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}},
    {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}, {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}},
   {{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}, {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}},
    {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}}, {{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},
    {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}, {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}}}
```

```
In[1551]:= Outer[Times, IdentityMatrix[3], IdentityMatrix[3]] // MatrixForm
```

Out[1551]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

```
In[1561]:= Outer[Times, IdentityMatrix[3], IdentityMatrix[3]] // MatrixForm
```

Out[1561]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

```
In[1563]:= Transpose[ArrayReshape[IdentityMatrix[9], {3, 3, 3, 3}], {2, 3, 1, 4}] //
 MatrixForm
```

Out[1563]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

In[1565]:= `ArrayReshape[IdentityMatrix[9], {3, 3, 3, 3}] // MatrixForm`

Out[1565]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

In[1559]:= `Transpose[Outer[Times, IdentityMatrix[3], IdentityMatrix[3]], {2, 3, 1, 4}] // MatrixForm`

Out[1559]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

`Transpose[Outer[Times, IdentityMatrix[3], IdentityMatrix[3]], {2, 3, 1, 4}] // MatrixForm`

In[1558]:= `ArrayReshape[IdentityMatrix[9], {3, 3, 3, 3}] // MatrixForm`

Out[1558]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{pmatrix}$$

In[1550]:= `inertDeriv2[{{x1, y1, z1}, {x2, y2, z2}}][[1]] // MatrixForm`

Out[1550]//MatrixForm=

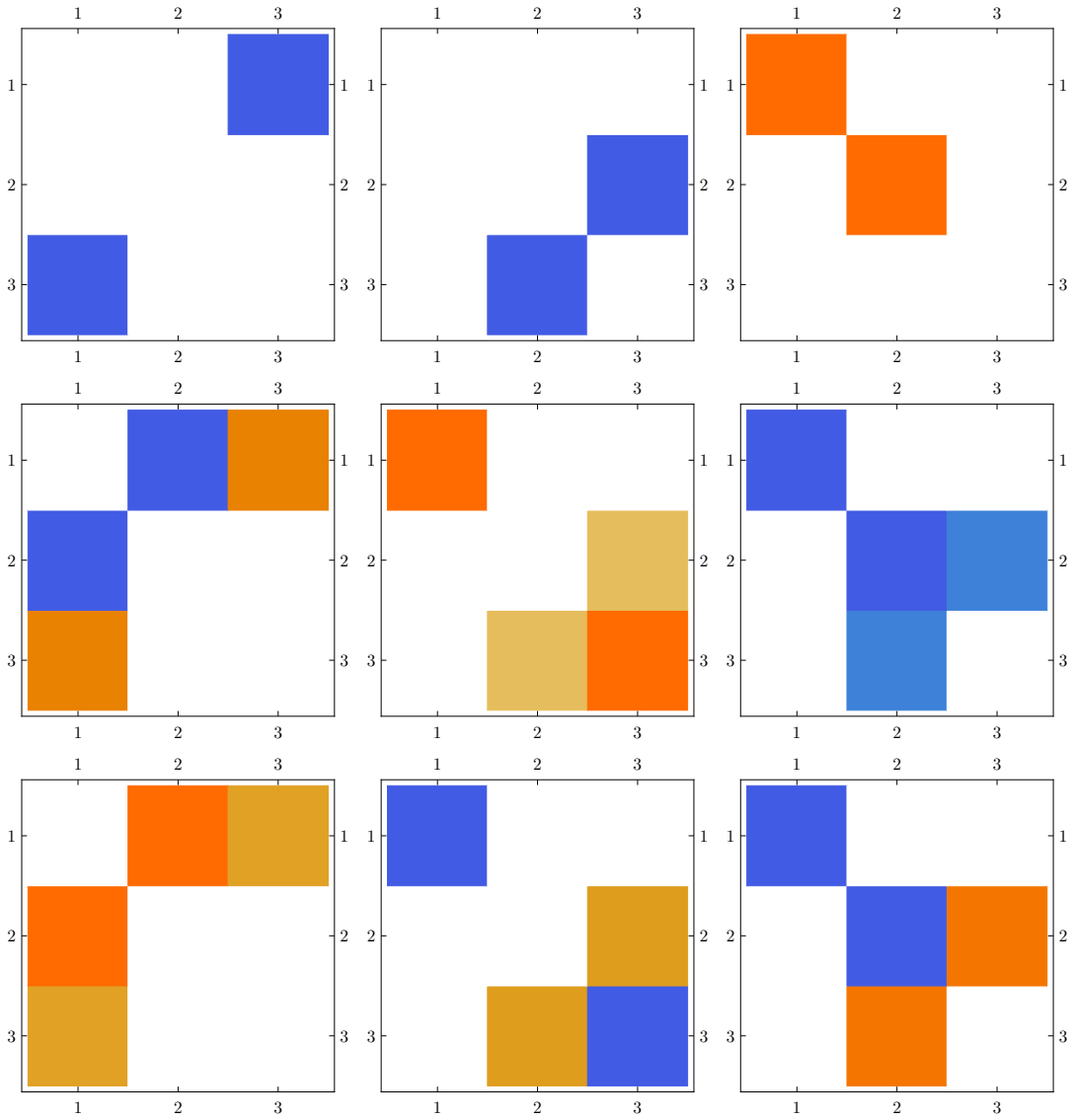
$$\begin{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{pmatrix}$$

In[1625]:=
```
inertDeriv[{{4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01},
    {-8.45304271*^-30, 6.15988973*^+01, -3.85889502*^+01},
    {-1.06642895*^-14, -8.70805320*^+01, -5.45520530*^+01}}] // Abs // Max
```

Out[1625]= 174.161

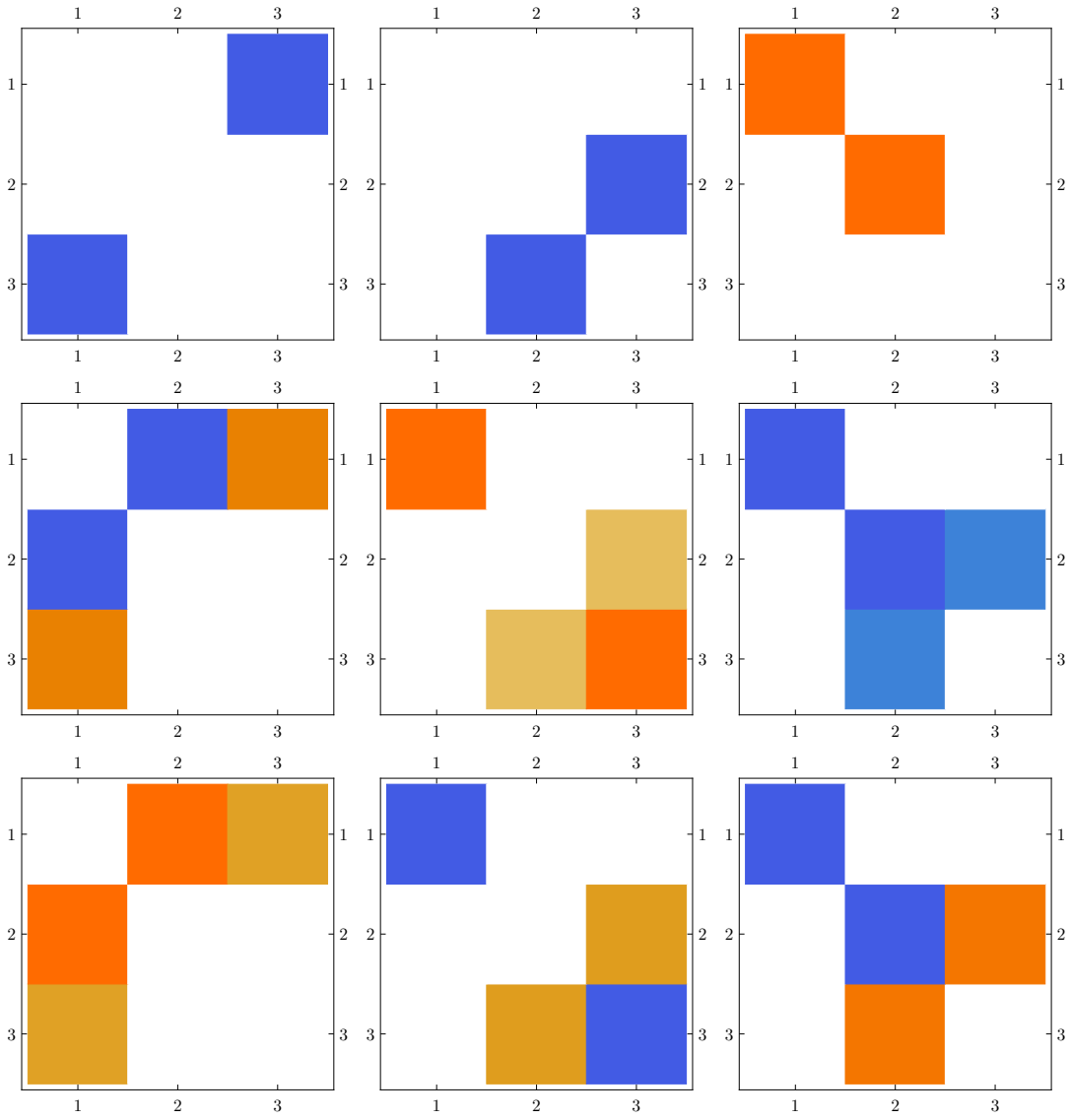
In[1623]:=
```
inertDeriv[{{4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01},
     {-8.45304271*^-30, 6.15988973*^+01, -3.85889502*^+01},
     {-1.06642895*^-14, -8.70805320*^+01, -5.45520530*^+01}}] //
   Threshold // Map[MatrixPlot, #, {2}] & // Grid
```

Out[1623]=



In[1624]:=
```
Round[
  inertDerivFD[{{4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01},
    {-8.45304271*^-30, 6.15988973*^+01, -3.85889502*^+01},
    {-1.06642895*^-14, -8.70805320*^+01, -5.45520530*^+01}}],
  10^-10
 ] // Threshold // Map[MatrixPlot, #, {2}] & // Grid
```

Out[1624]=



In[1609]:=
```
inert[pos_] := Sum[IdentityMatrix[3] * Dot[p, p] - Outer[Times, p, p], {p, pos}];
inertDeriv[pos_] := Table[
   2 * p.Outer[Times, IdentityMatrix[3], IdentityMatrix[3]] - (
     (Outer[Times, #, p] & /@ IdentityMatrix[3]) +
      ( Outer[Times, p, #] & /@ IdentityMatrix[3])
    ),
   {p, pos}
  ];
inertDerivFD[pos_] :=
  Table[
   Table[
    NDSolve`FiniteDifferenceDerivative[
      1,
      Table[dx, {dx, -.1, .1, .05}],
      Table[
       inert[
         pos + ReplacePart[
           ConstantArray[0, Dimensions[pos]],
           {n, a} → dx
          ]
        ][[i, j]],
       {dx, -.1, .1, .05}
      ]
     ][[3]],
    {i, 3},
    {j, 3}
   ],
   {n, Length[pos]},
   {a, 3}
  ];
inertDerivP2[pos_] := Table[
   (Outer[Times, #, p] & /@ IdentityMatrix[3]),
   {p, pos}
  ];
inertDeriv2[pos_] := Table[
   2 * Outer[Times, IdentityMatrix[3], IdentityMatrix[3]] - (
     # + Transpose[#, {1, 2, 4, 3}] &@(
       Outer[Times, IdentityMatrix[3], #] & /@ IdentityMatrix[3]
      )
    ),
   {p, pos}
  ];
```

In[1582]:=
```
inertDerivP2[{
        {4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01}
      }][[1]] // Threshold // Map[MatrixForm] // Row
```

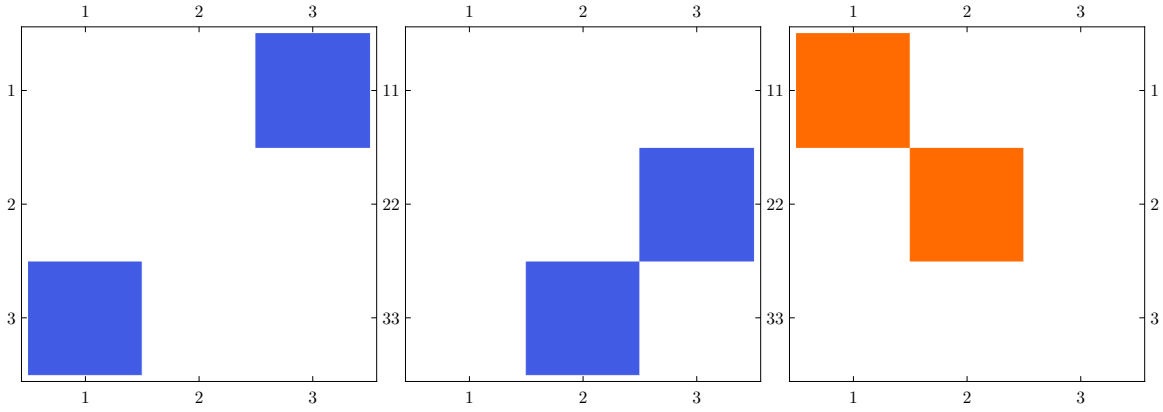
Out[1582]=
$$\begin{pmatrix} 0. & 0. & 38.4327 \\ 0. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}\begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 38.4327 \\ 0. & 0. & 0. \end{pmatrix}\begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 38.4327 \end{pmatrix}$$

In[1588]:=
```
inertDeriv[{
        {4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01}
      }][[1]] // Threshold // Map[MatrixForm] // Row
```

Out[1588]=
$$\begin{pmatrix} 0. & 0. & -38.4327 \\ 0. & 0. & 0. \\ -38.4327 & 0. & 0. \end{pmatrix}\begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & -38.4327 \\ 0. & -38.4327 & 0. \end{pmatrix}\begin{pmatrix} 76.8655 & 0. & 0. \\ 0. & 76.8655 & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

In[1574]:=
```
inertDeriv[{
        {4.20941234*^-30, -2.36968848*^-15, 3.84327414*^+01}
      }][[1]] // Threshold // Map[MatrixPlot[#, ImageSize → 200] &] // Row
```

Out[1574]=



In[1567]:=
```
inertDeriv[{
        {x1, y1, z1},
        {x2, y2, zz}
      }][[1]] // Map[MatrixForm] // Row
```

Out[1567]=
$$\begin{pmatrix} 0 & -y1 & -z1 \\ -y1 & 2\,x1 & 0 \\ -z1 & 0 & 2\,x1 \end{pmatrix}\begin{pmatrix} 2\,y1 & -x1 & 0 \\ -x1 & 0 & -z1 \\ 0 & -z1 & 2\,y1 \end{pmatrix}\begin{pmatrix} 2\,z1 & 0 & -x1 \\ 0 & 2\,z1 & -y1 \\ -x1 & -y1 & 0 \end{pmatrix}$$

In[1585]:=
```
inertDeriv2[{
    {x1, y1, z1},
    {x2, y2, zz}
   }][[2]] // MatrixForm
```

Out[1585]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{pmatrix}$$

In[1425]:=
```
inertDeriv[
    {
     {x1, y1, z1},
     {x2, y2, zz}
    }
   ][[1]] // Map[MatrixForm] // MatrixForm
```

Out[1425]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 0 & -y1 & -z1 \\ -y1 & 2\,x1 & 0 \\ -z1 & 0 & 2\,x1 \end{pmatrix} \\ \begin{pmatrix} 2\,y1 & -x1 & 0 \\ -x1 & 0 & -z1 \\ 0 & -z1 & 2\,y1 \end{pmatrix} \\ \begin{pmatrix} 2\,z1 & 0 & -x1 \\ 0 & 2\,z1 & -y1 \\ -x1 & -y1 & 0 \end{pmatrix} \end{pmatrix}$$

In[1431]:=
```
D[
  inert[
   {
    {x1, y1, z1},
    {x2, y2, zz}
   }
  ],
  x1,
  y1
 ] // MatrixForm
```

Out[1431]//MatrixForm=

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In[1587]:=
```
Table[
  With[{x = x, y = y},
   D[
    inert[
     {
      {x1, y1, z1},
      {x2, y2, zz}
     }
    ],
    x, y
   ]
  ],
  {x, {x1, y1, z1, x2, y2, zz}},
  {y, {x1, y1, z1, x2, y2, zz}}
 ] // MatrixForm
```

Out[1587]//MatrixForm=

$$\begin{pmatrix}
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
\begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}
\end{pmatrix}$$

In[1426]:=
```
{
 D[
  inert[
   {
    {x1, y1, z1},
    {x2, y2, zz}
   }
  ],
  x1
 ],
 D[
  inert[
   {
    {x1, y1, z1},
    {x2, y2, zz}
   }
  ],
  y1
 ],
 D[
  inert[
   {
    {x1, y1, z1},
    {x2, y2, zz}
   }
  ],
  z1
 ]
} // Map[MatrixForm] // MatrixForm
```

Out[1426]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 0 & -y1 & -z1 \\ -y1 & 2\,x1 & 0 \\ -z1 & 0 & 2\,x1 \end{pmatrix} \\ \begin{pmatrix} 2\,y1 & -x1 & 0 \\ -x1 & 0 & -z1 \\ 0 & -z1 & 2\,y1 \end{pmatrix} \\ \begin{pmatrix} 2\,z1 & 0 & -x1 \\ 0 & 2\,z1 & -y1 \\ -x1 & -y1 & 0 \end{pmatrix} \end{pmatrix}$$

Towards Analytic Expressions

We would like to take this very elegant way of doing perturbation theory based on the iterative

corrections given by

$$E_n^{(k)} = \left\langle n^{(0)} \right| H^{(k)} \left| n^{(0)} \right\rangle + \sum_{i=1}^{k-1} \left\langle n^{(0)} \right| H^{(k-i)} \left| n^{(i)} \right\rangle - E_n^{(k-i)} \left\langle n^{(0)} \middle| n^{(i)} \right\rangle$$

and use it to derive the next order of correction when we're using a set of harmonic zero-order states, assuming we have lower-order expressions in terms of the "classical actions" (i.e. $n_k + \frac{1}{2}$). We're not aiming for any flavor of explicit formula. Instead our goal will be to generate the coefficient matrix through the tensor operations, rather than to write them down by hand.

To see if this is even feasible, we'll start by trying to rewrite the second-order energy as

$$\begin{aligned} E_n^{(2)} &= X \odot a^2 \\ &= \sum_{ij} x_{ij} \left(n_i + \frac{1}{2} \right) \left(n_j + \frac{1}{2} \right) \end{aligned}$$

Targeted Representations of $H^{(1)}$ and $H^{(2)}$

First, write out the regular second-order energy correction as

$$E_n^{(2)} = \left\langle n^{(0)} \right| H^{(2)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \left| n^{(1)} \right\rangle$$

then we'll focus on what happens with $H^{(2)}$. When we operate with it we get something like

$$H^{(2)} \left| n^{(0)} \right\rangle = \sum_m c_m \left| m^{(0)} \right\rangle$$

but after projecting, we just get c_n. This c_n, then, should be partitioned into its parts coming from the various modes. To that effect, we note that for us (in rectilinear normal modes)

$$\begin{aligned} \left\langle n^{(0)} \right| H^{(2)} \left| n^{(0)} \right\rangle &= \left\langle n^{(0)} \right| \nabla_{q^4} V \odot qqqq \left| n^{(0)} \right\rangle \\ &= \nabla_{q^4} V \odot \left\langle n^{(0)} \right| qqqq \left| n^{(0)} \right\rangle \end{aligned}$$

and of course we established earlier how to compute representations of $\left\langle n^{(0)} \right| qqqq \left| n^{(0)} \right\rangle$, but what's important here is that only two flavors of permutations have the ability to properly couple, and those are

$$\left\langle n^{(0)} \right| q_i q_i q_i q_i \left| n^{(0)} \right\rangle$$
$$\left\langle n^{(0)} \right| q_i q_i q_j q_j \left| n^{(0)} \right\rangle$$

since no other flavor of these operators can leave the quantum numbers unchanged. This is incredibly convenient, and is the reason the classic x_{ij} expressions only contain only semi-diagonal elements in the quartic force constants. We'll deal with the coefficients later. For now, we need to figure out what to do about $\left| n^{(1)} \right\rangle$.

First off, we note that

$$\left|n^{(1)}\right\rangle = \Pi_n H^{(1)} \left|n^{(0)}\right\rangle$$

but the perturbation operator is actually incredibly simple in the harmonic basis, giving us

$$\begin{aligned}(\Pi_n)_{mm} &= \left(\left(H^{(0)} - E^{(0)}\right)^{-1}\right)_{mm} \\ &= \left(\sum_i \omega_i \left(m_i + \frac{1}{2}\right) - \omega_i \left(n_i + \frac{1}{2}\right)\right)^{-1} \\ &= \left(\sum_i \omega_i (m_i - n_i)\right)^{-1}\end{aligned}$$

assuming, of course, $n \neq m$ since we already projected that out. The most important feature of this, for what we're doing here, is that it's diagonal. For the ease of notation, we'll write $\Pi_n = \Pi$ and let the n be implicit.

Next, we'll write

$$H^{(1)} \left|n^{(0)}\right\rangle = \sum_m k_m^{(n)} \left|m^{(0)}\right\rangle$$

where we actually know explicitly which $k_m^{(n)}$ terms will be non-zero and how they relate to $\left|n^{(0)}\right\rangle$. For the ease of putting things down, though, we'll write it out like that (which is formally equivalent to obtaining a full representation of $H^{(1)}$ in an infinite basis).

Next, because the perturbation operator is diagonal, we get

$$\Pi H^{(1)} \left|n^{(0)}\right\rangle = \sum_m \Pi_{mm} k_m^{(n)} \left|m^{(0)}\right\rangle$$

so then

$$\begin{aligned}H^{(1)} \Pi H^{(1)} \left|n^{(0)}\right\rangle &= \sum_m \Pi_{mm} k_m^{(n)} \left|m^{(0)}\right\rangle \\ &= \sum_m \Pi_{mm} k_m^{(n)} \sum_l k_l^{(m)} \left|l^{(0)}\right\rangle \\ &= \sum_{m,l} \Pi_{mm} k_m^{(n)} k_l^{(m)} \left|l^{(0)}\right\rangle\end{aligned}$$

and finally

$$\begin{aligned}H^{(1)} \Pi H^{(1)} \left|n^{(0)}\right\rangle &= \sum_m \Pi_{mm} k_m^{(n)} \left|m^{(0)}\right\rangle \\ &= \sum_m \Pi_{mm} k_m^{(n)} \sum_l k_l^{(m)} \left|l^{(0)}\right\rangle \\ &= \sum_{m,l} \Pi_{mm} k_m^{(n)} k_l^{(m)} \left|l^{(0)}\right\rangle\end{aligned}$$

and then projecting with $\left\langle n^{(0)}\right|$ we recover the well-known expression

$$\left\langle n^{(0)}\right| H^{(1)} \Pi H^{(1)} \left|n^{(0)}\right\rangle = \sum_{m,l} \Pi_{mm} k_m^{(n)} k_n^{(m)}$$

Trying a different tack, writing

$$H^{(1)}=\nabla_{q^3} V\odot qqq$$

we'll note that

$$\left\langle n^{(0)}\right| H^{(1)}\,\Pi H^{(1)}\left|n^{(0)}\right\rangle=\sum_{\alpha\beta\gamma}\sum_{ijk}\left(V^{(3)}\right)_{\alpha\beta\gamma}\left(V^{(3)}\right)_{ijk}\left\langle n^{(0)}\right| q_\alpha\ q_\beta\ q_\gamma\,\Pi q_i\ q_j\ q_k\left|n^{(0)}\right\rangle$$

and now writing down (from raising/lowering operators, will explain in more detail below)

$$\begin{aligned} q^k\left|n_i\right\rangle &= \sum_{j=-k}^{k}\left(\sum_{g\in P_a^\alpha}\prod_{l\in g}\sqrt{n_i+l}\right)\left|n_i+j\right\rangle \\ &= \sum_{j=-k}^{k}\rho_j^k(n_i)\left|n_i+j\right\rangle \end{aligned}$$

we get

$$\Pi\, q_i^a\left|n_i\right\rangle\ q_j^b\left|n_j\right\rangle\ q_k^c\left|n_k\right\rangle=\sum_{\alpha=-a}^{a}\sum_{\beta=-b}^{b}\sum_{\gamma=-c}^{c}\frac{\rho_\alpha^a(n_i)\,\rho_\beta^b(n_j)\,\rho_\gamma^c(n_k)}{\alpha\omega_i+\beta\omega_j+\gamma\omega_k}\left|n_i+\alpha\right\rangle\left|n_j+\beta\right\rangle\left|n_k+\gamma\right\rangle\left(1-\delta_{\alpha,\beta,\gamma,0}\right)$$

where when, e.g. $b=0$ this is actually totally clean, since β is always 0 and $\left|n_j+\beta\right\rangle=\left|n_j\right\rangle$ which means it just comes directly out of the sum. One thing worth noting is that we have a 4D Kronecker-delta-esque thing here which is just a way of writing

$$\left(1-\delta_{\alpha,\beta,\gamma,0}\right)=\begin{cases}0 & \alpha=\beta=\gamma=0\\ 1 & \text{else}\end{cases}$$

Now we *could* just go straight to writing out an explicit form for

$$q_\alpha\ q_\beta\ q_\gamma\,\Pi q_i\ q_j\ q_k\left|n^{(0)}\right\rangle$$

but instead we should note that we have

$$\left\langle n^{(0)}\right| q_\alpha\ q_\beta\ q_\gamma\,\Pi q_i\ q_j\ q_k\left|n^{(0)}\right\rangle$$

and so only the operations of $q_\alpha\ q_\beta\ q_\gamma$ that will reverse the operation of $q_i\ q_j\ q_k\left|n^{(0)}\right\rangle$ or which already lead to a non-zero integral in $\left\langle n^{(0)}\right| q_\alpha\ q_\beta\ q_\gamma\left|n^{(0)}\right\rangle$ can contribute.

At the end of the day, then, this means we'll get things that look like

$$\left\langle n_i\right| q_i^{a'}\left\langle n_j\right|q_j^{b'}\left\langle n_k\right|q_k^{c'}\,\Pi\, q_i^a\left|n_i\right\rangle\ q_j^b\left|n_j\right\rangle\ q_k^c\left|n_k\right\rangle=$$

$$\sum_{\alpha=-a}^{a}\sum_{\beta=-b}^{b}\sum_{\gamma=-c}^{c}\frac{\rho_\alpha^a(n_i)\,\rho_{-\alpha}^{a'}(n_i+\alpha)\,\rho_\beta^b(n_j)\,\rho_{-\beta}^{b'}(n_j+\beta)\,\rho_\gamma^c(n_k)\,\rho_{-\gamma}^{c'}(n_k+\beta)}{\alpha\omega_i+\beta\omega_j+\gamma\omega_k}\left(1-\delta_{\alpha,\beta,\gamma,0}\right)$$

we'll ignore, just for the moment, how the derivatives play into all of this, but we'll note that from $q_i^a\ q_j^b\ q_k^c$ we can work back to its corresponding derivative tensor element.

Let's look at some concrete examples

$$\langle n_i|\, {q_i}^3 \Pi {q_i}^3\, |n_i\rangle = \sum_{\alpha=-3}^{3} \frac{\rho_\alpha^3(n_i)\,\rho_{-\alpha}^3(n_i+\alpha)}{\alpha\omega_i}\,(1-\delta_{\alpha 0})$$

$$= \frac{\rho_{-3}^3(n_i)\,\rho_3^3(n_i-3)}{-3\,\omega_i} + \frac{\rho_{-1}^3(n_i)\,\rho_1^3(n_i-1)}{-\omega_i} + \frac{\rho_1^3(n_i)\,\rho_{-1}^3(n_i+1)}{\omega_i} + \frac{\rho_3^3(n_i)\,\rho_{-3}^3(n_i+3)}{3\,\omega_i}$$

$$= \frac{1}{3\,\omega_i}\left(\rho_3^3(n_i)\,\rho_{-3}^3(n_i+3) - \rho_{-3}^3(n_i)\,\rho_3^3(n_i-3)\right) + \frac{1}{\omega_i}\left(\rho_1^3(n_i)\,\rho_{-1}^3(n_i+1) - \rho_{-1}^3(n_i)\,\rho_1^3(n_i-1)\right)$$

$$\langle n_i|\, {q_i}^2\, q_j \Pi {q_i}^3\, |n_i\rangle = \sum_{\alpha=-3}^{3} \frac{\rho_\alpha^3(n_i)\,\rho_{-\alpha}^2(n_i+\alpha)\,\rho_0^1(n_j)\,\rho_0^0(n_i)}{\alpha\omega_i}\,(1-\delta_{\alpha 0})$$

$$= 0$$

$$\langle n_j|\, \langle n_i|\, q_i\, {q_j}^2 \Pi {q_i}^3\, |n_i\rangle\, |n_j\rangle = \sum_{\alpha=-3}^{3} \frac{\rho_\alpha^3(n_i)\,\rho_{-\alpha}^1(n_i+\alpha)\,\rho_0^0(n_j)\,\rho_0^2(n_j)}{\alpha\omega_i}\,(1-\delta_{\alpha 0})$$

$$= \frac{1}{\omega_i}\left(\rho_1^3(n_i)\,\rho_{-1}^1(n_i+1) - \rho_{-1}^3(n_i)\,\rho_1^1(n_i-1)\right)\rho_0^0(n_j)\,\rho_0^2(n_j)$$

$$\langle n_i|\, q_i\, q_j\, q_k \Pi q_i\, q_j\, q_k\, |n_i\rangle\, |n_j\rangle\, |n_k\rangle = \sum_{\alpha=-1}^{1}\sum_{\beta=-1}^{1}\sum_{\gamma=-1}^{1} \frac{\rho_\alpha^1(n_i)\,\rho_{-\alpha}^1(n_i+\alpha)\,\rho_\beta^1(n_j)\,\rho_{-\beta}^1(n_i+\beta)\,\rho_\gamma^1(n_k)\,\rho_{-\gamma}^1(n_k+\gamma)}{\alpha\omega_i+\beta\omega_j+\gamma\omega_k}\,(1-$$

$$= \prod_z \frac{1}{\omega_z}\left(\rho_1^1(n_z)\,\rho_{-1}^1(n_z+1) - \rho_{-1}^1(n_z)\,\rho_1^1(n_z-1)\right)$$

And now the hard part begins, because our goal setting out has always been to coerce this into a form that looks like

$$\sum_{ij} x_{ij}\left(n_i+\frac{1}{2}\right)\left(n_j+\frac{1}{2}\right)$$

and so here we need to note two things: a) this form is honestly pretty clunky for the equations we have, given that ours have 3 indices and this has 2, and b) we're going to need to figure out what the deal is with $\rho_\alpha^a(n_i)$.

To focus on that second part, we'll note that

$$\rho_N^\alpha(n_i) = \sum_{p\in P_N^\alpha} \prod_{l\in p} \sqrt{\frac{n_i+l}{2}}$$

Notes on generating the paths are given below. This means that for a path of length N we get a leading term of $1 \Big/ \sqrt{2}^{\,N}$, and so for even N we have a power of 1/2.

Next, we're unfortunately going to need to tabulate the relevant cases

$${q_i}^3 \Pi {q_i}^3 = \{\{3,\ 0,\ 0\},\ \{3,\ 0,\ 0\}\}$$
$${q_i}^3 \Pi {q_i}^1\ {q_j}^2 = \{\{3,\ 0,\ 0\},\ \{1,\ 2,\ 0\}\}$$
$${q_i}^2\ q_j \Pi {q_j}^3 = \{\{2,\ 1,\ 0\},\ \{0,\ 3,\ 0\}\}$$
$${q_i}^2\ q_j \Pi {q_i}^2\ q_j = \{\{2,\ 1,\ 0\},\ \{2,\ 1,\ 0\}\}$$
$${q_i}^2\ q_j \Pi q_j\ {q_k}^2 = \{\{2,\ 1,\ 0\},\ \{0,\ 1,\ 2\}\}$$

$$q_i\ q_j\ q_k \Pi q_i\ q_j\ q_k = \{\{1,\ 1,\ 1\},\ \{1,\ 1,\ 1\}\}$$

and it's clear that we need the even-ness of the indices to match up on the left and right. So in the future we can programmatically generate these cases.

Now let's see what the matrix elements for these cases look like

$$q_i^3 \Pi q_i^3 = \frac{7 + 60\left(\frac{1}{2} + n_i\right)^2}{16\ \omega_i}$$

$$q_i^3 \Pi q_i^1\ q_j^2 = \frac{3\left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_j\right)}{2\ \omega_i}$$

$$q_i^2\ q_j \Pi q_j^3 = \frac{3\left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_j\right)}{2\ \omega_j}$$

$$q_i^2\ q_j \Pi q_i^2\ q_j = \frac{-32\left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_j\right)\omega_i\ \omega_j + 3\ \omega_j^2 - 4\left(\frac{1}{2} + n_i\right)^2\left(8\ \omega_i^2 - 3\ \omega_j^2\right)}{16\left(-4\ \omega_i^2\ \omega_j + \omega_j^3\right)}$$

$$q_i^2\ q_j \Pi q_j\ q_k^2 = \frac{\left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_k\right)}{2\ \omega_j}$$

$$q_i\ q_j\ q_k \Pi q_i\ q_j\ q_k = \frac{1}{32\ (\omega_i - \omega_j - \omega_k)} - \frac{1}{32\ (\omega_i + \omega_j - \omega_k)} - \frac{1}{32\ (\omega_i - \omega_j + \omega_k)} + \frac{1}{32\ (\omega_i + \omega_j + \omega_k)} +$$
$$\frac{1}{8}\left(\frac{1}{2} + n_j\right)\left(\frac{1}{2} + n_k\right)\left(\frac{1}{\omega_i - \omega_j - \omega_k} + \frac{1}{\omega_i + \omega_j - \omega_k} + \frac{1}{\omega_i - \omega_j + \omega_k} + \frac{1}{\omega_i + \omega_j + \omega_k}\right)$$
$$- \frac{\left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_j\right)\omega_k\left(\omega_i^2 + \omega_j^2 - \omega_k^2\right) + \left(\frac{1}{2} + n_i\right)\left(\frac{1}{2} + n_k\right)\omega_j\left(\omega_i^2 - \omega_j^2 + \omega_k^2\right)}{2\ (\omega_i - \omega_j - \omega_k)\ (\omega_i + \omega_j - \omega_k)\ (\omega_i - \omega_j + \omega_k)\ (\omega_i + \omega_j + \omega_k)}$$

And we see they are broadly factorizable in terms of expansions in $n + \frac{1}{2}$!

These can then potentially be factored into their contributions from the different actions, which would allow us to build something like an x-matrix, or even better, could be used to generate energies which than can be *fit* to an x-matrix form.

2 to 4

Next, we'll try to go from 2nd order to 4th order in the energy. The 3rd order is correction rigorously zero. So note that our 4th order correction can be written as

$$E_n^{(4)} = \left\langle n^{(0)} \right| H^{(4)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(3)} \left| n^{(1)} \right\rangle - E_n^{(3)} \left\langle n^{(0)} \middle| n^{(1)} \right\rangle +$$
$$\left\langle n^{(0)} \right| H^{(2)} \left| n^{(2)} \right\rangle - E_n^{(2)} \left\langle n^{(0)} \middle| n^{(2)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \left| n^{(3)} \right\rangle - E_n^{(1)} \left\langle n^{(0)} \middle| n^{(3)} \right\rangle$$

dropping out the zero terms we have

$$E_n^{(4)} = \left\langle n^{(0)} \right| H^{(4)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(3)} \left| n^{(1)} \right\rangle + \left\langle n^{(0)} \right| H^{(2)} \left| n^{(2)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \left| n^{(3)} \right\rangle - E_n^{(2)} \left\langle n^{(0)} \middle| n^{(2)} \right\rangle$$

then expanding those we get

$$\left\langle n^{(0)} \right| H^{(4)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(3)} \left| n^{(1)} \right\rangle + \left\langle n^{(0)} \right| H^{(2)} \left| n^{(2)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \left| n^{(3)} \right\rangle - E_n^{(2)} \left\langle n^{(0)} \middle| n^{(2)} \right\rangle$$

$$\left\langle n^{(0)} \right| H^{(3)} \Pi_n H^{(1)} \left| n^{(0)} \right\rangle$$

$$\left\langle n^{(0)} \right| H^{(2)} \Pi_n H^{(2)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(2)} \Pi_n H^{(1)} \Pi_n H^{(1)} \left| n^{(0)} \right\rangle - \frac{1}{2} \left\langle n^{(0)} \right| H^{(1)} \Pi_n \Pi_n H^{(1)} \left| n^{(0)} \right\rangle \left\langle n^{(0)} \right| H^{(2)} \left| n^{(0} \right.$$

$$\left\langle n^{(0)} \right| H^{(1)} \Pi_n H^{(3)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \Pi_n H^{(2)} \left| n^{(1)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \Pi_n H^{(1)} \left| n^{(2)} \right\rangle - E_n^{(2)} \left\langle n^{(0)} \right| H^{(1)} \Pi_n \Pi_n H^{(1)} \left| \right.$$

$$\left\langle n^{(0)} \right| H^{(1)} \Pi_n H^{(1)} \Pi_n H^{(2)} \left| n^{(0)} \right\rangle + \left\langle n^{(0)} \right| H^{(1)} \Pi_n H^{(1)} \Pi_n H^{(1)} \Pi_n H^{(1)} \left| n^{(0)} \right\rangle - \frac{1}{2} \left\langle n^{(0)} \right| H^{(1)} \Pi_n \Pi_n H^{(1)} \left| n^{(0)} \right\rangle$$

and ooooh fuck I don't want to expand any further. We will clearly need a programmatic strategy for dealing with these kinds of terms.

Implementation Notes

General Raising/Lowering Representation of q^n

Inclusion of Momenta

Let's look back at

$$Q = (a + a^+)$$
$$p = (-i)\,(a - a^+)$$

we see that the terms we get come from generating the Cartesian product of pairs like $(1,\ 1)$ for Q and $(1,\ -1)$ for p. Our permutations are constructed in the same way, but in that case all of the pairs are $(-1,\ 1)$ because as written the lowering operator comes first. This means we can write

$$\prod_r o_r \left| n_i \right\rangle = \sum_{\alpha=-N}^{N} \left(\sum_{\{\phi,\ p\} \in P_{o_r}^{\alpha}} \phi \prod_{l \in p} \sqrt{\frac{n_i + l}{2}} \right) \left| n_i + j \right\rangle$$
$$= \sum_{\alpha=-N}^{N} \rho_{o_r}^{\alpha}(n_i) \left| n_i + j \right\rangle$$

where o_r is a sequence of N operators (where each is Q or p), $P_{o_r}^{\alpha}$ is the same as P_N^{α} but now also tracking phase information in the permutations, as given by ϕ. This will include

Now we'd like to see if the phase information encoded in o_r can be deduced if we're given a permutation. To do that, we'll start with a function to generate $P_{o_r}^{\alpha}$ (defined below).

Here are the paths in pQQp that change quanta by +2 with associated phase

In[1103]:=
```
pathStepsWithPhases[2, {p, q, q, p}] // Thread // Column
```

Out[1103]=
```
{-1, {-1, 1, 1, 1}}
{1, {1, -1, 1, 1}}
{1, {1, 1, -1, 1}}
{-1, {1, 1, 1, -1}}
```

here's the same for ppQQ

In[1109]:=
```
pathStepsWithPhases[2, {p, p, q, q}] // Thread // Column
```

Out[1109]=
```
{-1, {-1, 1, 1, 1}}
{-1, {1, -1, 1, 1}}
{1, {1, 1, -1, 1}}
{1, {1, 1, 1, -1}}
```

or pQpQ

In[1106]:=
```
pathStepsWithPhases[2, {p, q, p, q}] // Thread // Column
```

Out[1106]=
```
{-1, {-1, 1, 1, 1}}
{1, {1, -1, 1, 1}}
{-1, {1, 1, -1, 1}}
{1, {1, 1, 1, -1}}
```

and it certainly seems we only get the negative signs when $o_r == p$, assuming lexical permutation ordering. Of course, that only worked because we had the same number of permutations as operators. What happens when we consider $\alpha = 0$?

In[1117]:=
```
Grid@{
  Row /@ {
    {p, q, q, p},
    {p, p, q, q},
    {p, q, p, q}
   },
  {
   pathStepsWithPhases[0, {p, q, q, p}] // Thread // Column,
   pathStepsWithPhases[0, {p, p, q, q}] // Thread // Column,
   pathStepsWithPhases[0, {p, q, p, q}] // Thread // Column
  }
 }
```

Out[1117]=

pqqp	ppqq	pqpq
{-1, {-1, -1, 1, 1}}	{1, {-1, -1, 1, 1}}	{-1, {-1, -1, 1, 1}}
{-1, {-1, 1, -1, 1}}	{-1, {-1, 1, -1, 1}}	{1, {-1, 1, -1, 1}}
{1, {-1, 1, 1, -1}}	{-1, {-1, 1, 1, -1}}	{-1, {-1, 1, 1, -1}}
{1, {1, -1, -1, 1}}	{-1, {1, -1, -1, 1}}	{-1, {1, -1, -1, 1}}
{-1, {1, -1, 1, -1}}	{-1, {1, -1, 1, -1}}	{1, {1, -1, 1, -1}}
{-1, {1, 1, -1, -1}}	{1, {1, 1, -1, -1}}	{-1, {1, 1, -1, -1}}

Or $\alpha = -2$?

In[1118]:=
```
Grid@{
  Row /@ {
    {p, q, q, p},
    {p, p, q, q},
    {p, q, p, q}
   },
  {
   pathStepsWithPhases[-2, {p, q, q, p}] // Thread // Column,
   pathStepsWithPhases[-2, {p, p, q, q}] // Thread // Column,
   pathStepsWithPhases[-2, {p, q, p, q}] // Thread // Column
  }
 }
```

Out[1118]=

pqqp	ppqq	pqpq
{-1, {-1, -1, -1, 1}}	{1, {-1, -1, -1, 1}}	{1, {-1, -1, -1, 1}}
{1, {-1, -1, 1, -1}}	{1, {-1, -1, 1, -1}}	{-1, {-1, -1, 1, -1}}
{1, {-1, 1, -1, -1}}	{-1, {-1, 1, -1, -1}}	{1, {-1, 1, -1, -1}}
{-1, {1, -1, -1, -1}}	{-1, {1, -1, -1, -1}}	{-1, {1, -1, -1, -1}}

and all of this tells us what we could basically have guessed: *we have a negative phase when we raise and odd number of times using* p. In algorithmic terms, that means if we have a permutation/sequence of raising lowering operations, e.g.

$$(a_r)=\{..., 1, -1, ..., \}$$

and we have an associate sequence $\{o_r\}$

$$(o_r)=\{..., p, q, ...\}$$

the phase for the sequence is given by

$$\phi=\prod_{r\in\{r\mid o_r \text{ is } p\}} -a_r$$

we can see this works because lowering operations have $-(-1) = 1$. If we want to include the $(-i)$ on each p term we need to multiply that into each term, giving us

$$\phi=\prod_{r\in\{r\mid o_r \text{ is } p\}} ia_r$$

Implementation

In[1112]:=
```
pathStepsWithPhases // Clear
pathStepsWithPhases[target_, seq_, p_ : p, q_ : q] :=
  Block[
   {
    phaseElems = If[# === p, {1, -1}, {1, 1}] & /@ seq,
    pathElems = ConstantArray[{-1, 1}, Length[seq]],
    sel,
    phases,
    perms,
    states,
    paths
   },
   perms = Tuples[pathElems];
   phases = Tuples[phaseElems];
   sel = Total[#] == target & /@ perms;
   phases = Pick[phases, sel];
   perms = Pick[perms, sel];
   {Times @@@ phases, perms}
  ];
```

In[694]:=
```
Clear[phasedPathsNi, phasedRhoNi];
phasedPathsNi[seq_, target_, p_ : p, q_ : q] :=
  Block[
   {
    perms =
     Join @@ Permutations /@ IntegerPartitions[target, {Length[seq]}, {-1, 1}],
    sel = Pick[Range[Length[seq]], # === p & /@ seq],
    states,
    paths,
    phases
   },
   states = Accumulate /@ perms;
   paths  = states + Unitize[perms - 1];
   phases = Times @@@ perms[[ ;; , sel]];
   {I^Length[sel] * phases, paths, perms}
  ];
phasedRhoNi[seq_, target_, ni_, p_ : p, q_ : q] :=
 With[{pathStuff = phasedPathsNi[seq, target, p, q]},
  Total@
   MapThread[
    Function[{ϕ, pa},
     ϕ / Sqrt[2^Length[seq]] *
      Sqrt[Times @@ (If[NumericQ[ni], UnitStep[#], 1] * #) &[
        ni + pa
       ]
      ]
    ],
    pathStuff[[ ;; 2]]
   ]
 ]
```

In[1206]:=
```
phasedPathsNi[{p, q, p, q}, 2]
```

Out[1206]=
```
{{-1, 1, -1, 1}, {{1, 2, 3, 3}, {1, 2, 2, 2}, {1, 1, 1, 2}, {0, 0, 1, 2}},
 {{1, 1, 1, -1}, {1, 1, -1, 1}, {1, -1, 1, 1}, {-1, 1, 1, 1}}}
```

In[1256]:=
```
phasedRhoNi[{p, q, p, q}, 2, 10]
```

Out[1256]=
$$-\sqrt{33}$$

In[1266]:=
```
rhoNi[2, 2, 10]
```

Out[1266]=
$$\sqrt{33}$$

In[1274]:=
```
phasedRhoNi[ConstantArray[q, 2], 2, ni] + phasedRhoNi[ConstantArray[p, 2], 2, ni]
```

Out[1274]=
```
0
```

In[1284]:=
```
phasedRhoNi[ConstantArray[q, 2], -2, ni] +
  phasedRhoNi[ConstantArray[p, 2], -2, ni] //
 Assuming[ni ∈ Integers && ni > 0, Simplify[#]] &
```

Out[1284]= 0

In[1285]:=
```
phasedRhoNi[{q, p, p, q}, -2, ni]
```

Out[1285]=
$$-\frac{1}{4}\sqrt{(-2+ni)^2\,(-1+ni)\,ni}+\frac{1}{4}\sqrt{(-1+ni)^3\,ni}+\frac{1}{4}\sqrt{(-1+ni)\,ni^3}-\frac{1}{4}\sqrt{(-1+ni)\,ni\,(1+ni)^2}$$

In[1286]:=
```
phasedRhoNi[{q, q, q, q}, -2, ni]
```

Out[1286]=
$$\frac{1}{4}\sqrt{(-2+ni)^2\,(-1+ni)\,ni}+\frac{1}{4}\sqrt{(-1+ni)^3\,ni}+\frac{1}{4}\sqrt{(-1+ni)\,ni^3}+\frac{1}{4}\sqrt{(-1+ni)\,ni\,(1+ni)^2}$$

In[1287]:=
```
phasedRhoNi[ConstantArray[q, 10], 7, ni]
```

Out[1287]= 0

In[1310]:=
```
op = ConstantArray[q, 4] ~Join~ ConstantArray[p, 4];
Row@{Bra[8], Row[op], Ket[Subscript[n, i]]} == phasedRhoNi[op, 8, ni]
```

Out[1311]=
$$\langle 8|\mathrm{qqqqpppp}|n_i\rangle == \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)\,(7+ni)\,(8+ni)}$$

In[1318]:=
```
op = ConstantArray[{q, p}, 4] // Flatten;
Row@{Bra[8], Row[op], Ket[Subscript[n, i]]} == phasedRhoNi[op, 8, ni]
```

Out[1319]=
$$\langle 8|\mathrm{qpqpqpqp}|n_i\rangle == \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)\,(7+ni)\,(8+ni)}$$

In[1322]:=
```
With[{op = ConstantArray[q, 4] ~Join~ ConstantArray[p, 4], m = 6},
 Row@{Bra[m], Row[op], Ket[Subscript[n, i]]} == phasedRhoNi[op, m, ni]
]
```

Out[1322]=
$$\langle 6|\mathrm{qqqqpppp}|n_i\rangle == \frac{1}{16}\sqrt{ni^2\,(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)}+\frac{1}{16}\sqrt{(1+ni)^3\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)}+\frac{1}{16}\sqrt{(1+ni)\,(2+ni)^3\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)}+\frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)^3\,(4+ni)\,(5+ni)\,(6+ni)}-\frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)^3\,(5+ni)\,(6+ni)}-\frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)^3\,(6+ni)}-\frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)^3}-\frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)\,(7+ni)^2}$$

In[1321]:=
```
With[{op = ConstantArray[{q, p}, 4] // Flatten, m = 6},
 Row@{Bra[m], Row[op], Ket[Subscript[n, i]]} == phasedRhoNi[op, m, ni]
]
```

Out[1321]=
$$\langle 6 | \text{qpqpqpqp} | n_i \rangle == \frac{1}{16}\sqrt{ni^2\,(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)} - \frac{1}{16}\sqrt{(1+ni)^3\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)} + \frac{1}{16}\sqrt{(1+ni)\,(2+ni)^3\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)} - \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)^3\,(4+ni)\,(5+ni)\,(6+ni)} + \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)^3\,(5+ni)\,(6+ni)} - \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)^3\,(6+ni)} + \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)^3} - \frac{1}{16}\sqrt{(1+ni)\,(2+ni)\,(3+ni)\,(4+ni)\,(5+ni)\,(6+ni)\,(7+ni)^2}$$

In[1302]:=
```
ConstantArray[{q, p}, 4] // Flatten // Row
phasedRhoNi[ConstantArray[{q, p}, 4] // Flatten, 8, ni]
```

In[1366]:=
```
xMat[n_] :=
  SparseArray[{
    Band[{1, 2}] → Sqrt[Range[n]] / Sqrt[2],
    Band[{2, 1}] → Sqrt[Range[n]] / Sqrt[2]
   }];
pMat[n_] :=
  SparseArray[{
    Band[{1, 2}] → I * Sqrt[Range[n]] / Sqrt[2],
    Band[{2, 1}] → -I * Sqrt[Range[n]] / Sqrt[2]
   }];
```

In[1368]:=
```
pMat[10].xMat[10].pMat[10] // Simplify // MatrixForm
```

Out[1368]//MatrixForm=

$$\begin{pmatrix}
0 & \frac{1}{2\sqrt{2}} & 0 & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2\sqrt{2}} & 0 & 1 & 0 & -\sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \frac{3\sqrt{\frac{3}{2}}}{2} & 0 & -\sqrt{\frac{15}{2}} & 0 & 0 & 0 & 0 & 0 \\
-\frac{\sqrt{3}}{2} & 0 & \frac{3\sqrt{\frac{3}{2}}}{2} & 0 & 2\sqrt{2} & 0 & -\sqrt{15} & 0 & 0 & 0 & 0 \\
0 & -\sqrt{3} & 0 & 2\sqrt{2} & 0 & \frac{5\sqrt{\frac{5}{2}}}{2} & 0 & -\frac{\sqrt{105}}{2} & 0 & 0 & 0 \\
0 & 0 & -\sqrt{\frac{15}{2}} & 0 & \frac{5\sqrt{\frac{5}{2}}}{2} & 0 & 3\sqrt{3} & 0 & -\sqrt{42} & 0 & 0 \\
0 & 0 & 0 & -\sqrt{15} & 0 & 3\sqrt{3} & 0 & \frac{7\sqrt{\frac{7}{2}}}{2} & 0 & -3\sqrt{7} & 0 \\
0 & 0 & 0 & 0 & -\frac{\sqrt{105}}{2} & 0 & \frac{7\sqrt{\frac{7}{2}}}{2} & 0 & 8 & 0 & -3\sqrt{10} \\
0 & 0 & 0 & 0 & 0 & -\sqrt{42} & 0 & 8 & 0 & \frac{27}{2\sqrt{2}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -3\sqrt{7} & 0 & \frac{27}{2\sqrt{2}} & 0 & -\frac{\sqrt{5}}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -3\sqrt{10} & 0 & -\frac{\sqrt{5}}{2} & 0
\end{pmatrix}$$

In[1347]:=
```
Table[phasedRhoNi[{p, q}, j - i, i], {i, 0, 10}, {j, 0, 10}] // MatrixForm
```

Out[1347]//MatrixForm=

$$\begin{pmatrix}
\frac{i}{2} & 0 & \frac{i}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{i}{2} & 0 & i\sqrt{\frac{3}{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{i}{\sqrt{2}} & 0 & \frac{i}{2} & 0 & i\sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -i\sqrt{\frac{3}{2}} & 0 & \frac{i}{2} & 0 & i\sqrt{5} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -i\sqrt{3} & 0 & \frac{i}{2} & 0 & i\sqrt{\frac{15}{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -i\sqrt{5} & 0 & \frac{i}{2} & 0 & i\sqrt{\frac{21}{2}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -i\sqrt{\frac{15}{2}} & 0 & \frac{i}{2} & 0 & i\sqrt{14} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -i\sqrt{\frac{21}{2}} & 0 & \frac{i}{2} & 0 & 3i\sqrt{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -i\sqrt{14} & 0 & \frac{i}{2} & 0 & 3i\sqrt{\frac{5}{2}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -3i\sqrt{2} & 0 & \frac{i}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3i\sqrt{\frac{5}{2}} & 0 & \frac{i}{2}
\end{pmatrix}$$

Tensor Element Permutations

Explicit Operator Evaluations

General-Purpose Term-Generator

The goal is to create a function that can build the expressions used to evaluate any term appearing in

$$\left\langle n^{(0)}\right| H^{(k_1)}\,\Pi H^{(k_1)}\,\Pi\ldots\Pi H^{(k_M)}\left|n^{(0)}\right\rangle$$

where each $H^{(k_1)}$ is going to really be expressed as a set of coefficients attached to an operator built out of p and Q terms.

The key to this is just quantum-number counting. To do that we figure out the total-quanta changes that each H term can effect and then figure out the total set of paths to 0, then add up the effects.

In[1407]:=
```
ladderUpAndDownPaths[changes : {__Integer}] :=
 Block[
  {
   maxDim = Max[changes],
   displacements
  },
  displacements =
   Function[{dim},
     Select[
      Tuples[Range[-dim, dim], maxDim],
      Total[Abs[#]] == dim &
     ]
    ] /@ changes;
  Select[Tuples[displacements], AllTrue[Total[#], EqualTo[0]] &]
 ]
```

What effect does this have in practice? We'll recall that we could write

$$\left\langle n_i\right|\left\langle n_j\right|\left\langle n_k\right| q_i^{a'}\,q_j^{b'}\,q_k^{c'}\,\Pi\,q_i^{a}\;q_j^{b}\;q_k^{c}\left|n_i\right\rangle\left|n_j\right\rangle\left|n_k\right\rangle=$$

$$\sum_{\alpha=-a}^{a}\sum_{\beta=-b}^{b}\sum_{\gamma=-c}^{c}\frac{\rho_{\alpha}^{a}(n_i)\,\rho_{-\alpha}^{a'}(n_i+\alpha)\,\rho_{\beta}^{b}(n_j)\,\rho_{-\beta}^{b'}(n_j+\beta)\,\rho_{\gamma}^{c}(n_k)\,\rho_{-\gamma}^{c'}\,(n_k+\beta)}{\alpha\omega_i+\beta\omega_j+\gamma\omega_k}\,(1-\delta_{\alpha,\beta,\gamma,0})$$

and now we can adapt this idea to get a recursive way to express these things.

Let's assume we have operators A and B that can change our quanta by M and N respectively. Let $m = \min(M,\ N)$. If A and B are just products of Q, p, and Π, this means that up to m modes can be excited/de-excited. We need to be careful about how we think about this partitioning, though, since if we just think about the terms pQp and QQQ, we note that if we want to put excite/de-excite mode j there is only one way with QQQ, namely

$$Q_jQ_jQ_k$$

but with the not-totally symmetric one there are multiple ways

$$p_j Q_j p_k,\ p_j\, p_j Q_k,\ p_i\, Q_j\, p_j$$

which is all just to say, A and B will contain information about which modes they're able to excite/de-excite. Let's call that info Δ, which will be a sequence of integers detailing which modes it can excite and by how many quanta. That allows us to write

$$\left\langle n^{(0)}\right| A \Pi B \left|n^{(0)}\right\rangle = \sum_{d_1=\Delta_1^B}^{\Delta_1^B} \cdots \sum_{d_m=\Delta_m^B}^{\Delta_m^B} \pi(\boldsymbol{d}) \prod_i \chi_A\left(n_i + d_i,\ -d_i,\ \Delta_i^A\right) \chi_B\left(n_i,\ d_i,\ \Delta_i^B\right)$$

where $\pi(\boldsymbol{d})$ is just tracking the effect of the perturbation operator and χ_A and χ_B just track the coefficients associated with the excitations/de-excitations in mode i.

Let's look at a concrete case to make this clearer. We'll look at one of the terms in one of the the internal coordinate versions of $H^{(1)}$, $p_i Q_j p_j \Pi q_j q_j q_i$ (we'll note that by index math $p_i Q_j p_j \Pi q_j q_j q_k$ must be zero, since q_k has to excite n_k but there's nothing to bring it back down in $p_i Q_j p_j$). So for this we have $\{\Delta_i = 1,\ \Delta_j = 2\}$ and while this isn't required in general, $p_i Q_j p_j$ has the same possible excitations/de-excitations.

We can then write this term out as

$$\left\langle n^{(0)}\right| p_i Q_j p_j \Pi q_j q_j q_i \left|n^{(0)}\right\rangle =$$

$$\sum_{d_i=-1}^{1} \sum_{d_j=-2}^{2} \frac{\left(1-\delta_{d_i,d_j,0}\right)}{d_i\omega_i + d_j\omega_j} \rho_{Qp}(n_j + d_j\ ;\ -d_j,\ 2)\, \rho_{qq}(n_j\ ;\ d_j,\ 2)\, \rho_p(n_i + d_i\ ;\ -d_i,\ 1)\, \rho_q(n_i\ ;\ d_i,\ 1)$$

what if we wanted to toss in an extra term $q_k q_k$ term (separated by a perturbation operator)? By our recursive formulation we'll say

$$B = p_i Q_j p_j \Pi q_j q_j q_i$$
$$\Delta^B = \left\{\Delta_i^B = 2,\ \Delta_j^B = 4\right\}$$
$$\Delta^A = \left\{\Delta_k^A = 2\right\}$$

and

$$\left\langle n^{(0)}\right| q_k q_k \Pi p_i Q_j p_j \Pi q_j q_j q_i \left|n^{(0)}\right\rangle =$$

$$\sum_{d_i=-2}^{2} \sum_{d_j=-4}^{4} \pi(\boldsymbol{d})\, \rho_{q_k q_k}\left(n_i + d_i,\ -d_i,\ \Delta_i^A\right) \chi_B\left(n_i,\ d_i,\ \Delta_i^B\right) \rho_{q_k q_k}\left(n_j + d_j,\ -d_j,\ \Delta_j^A\right) \chi_B\left(n_j,\ d_j,\ \Delta_j^B\right)$$

and this seems tough even just to evaluate χ_B, until you think about the fact that $\Delta_i^A = \Delta_j^A = 0$, which means that the only time this is non-zero is when $d_i = d_j = 0$.

Then going back to previous expressions, to look at

$$\chi_B(n_j,\ 0,\ 4) = \frac{(1-\delta_{000})}{d_j\omega_j} \rho_{Qp}(n_j + d_j\ ;\ 0,\ 2)\, \rho_{qq}(n_j\ ;\ 0,\ 2)$$

$$\chi_B(n_i,\,0,\,2)=\frac{(1-\delta_{000})}{d_i\omega_i}\,\rho_p(n_i+d_i\,;\,0,\,1)\,\rho_q(n_i\,;\,0,\,1)$$

we see, even totally ignoring everything on the right, that the perturbation operator term has made this zero. We could also have applied the same logic to the $\pi(\boldsymbol{d})$ term in the earlier summation. Therefore,

$$\left\langle n^{(0)}\right|\,q_k q_k\,\Pi p_i Q_j p_j \Pi\, q_j q_j q_i\,\left|n^{(0)}\right\rangle=0$$

and that gives us a key simplification, in

$$\left\langle n^{(0)}\right|\,A\Pi B\,\left|n^{(0)}\right\rangle=\sum_{d_1=\Delta_1^B}^{\Delta_1^B}\cdots\sum_{d_m=\Delta_m^B}^{\Delta_m^B}\pi(\boldsymbol{d})\prod_i\chi_A\left(n_i+d_i,\,-d_i,\,\Delta_i^A\right)\chi_B\left(n_i,\,d_i,\,\Delta_i^B\right)$$

we only get non-zero terms if Δ^A and Δ^B are able to act on at least one of the same modes. Furthermore, since we're working with raising/lowering terms, we also know that for every shared mode, i, Δ_i^A and Δ_i^B must both be even or odd and for every *unshared* mode, j, the Δ_j^B or Δ_j^A must be even. This gives us pruning rules for terms coming from stuff like

$$\left\langle n^{(0)}\right|\,H^{(1)}\,\Pi_n\,H^{(1)}\,\Pi_n\,H^{(2)}\,\left|n^{(0)}\right\rangle$$

which naively would have a very, very large number of possible elements. Just taking of of the combinations of terms represented in it, we have

$$\left\langle n^{(0)}\right|\left(\frac{1}{2}\,\nabla_Q G\odot_3 pQp\right)\Pi_n\left(\frac{1}{6}\,\nabla_{Q^3}\,V\odot_3 QQQ\right)\Pi_n\left(\frac{1}{4}\,\nabla_{Q^2}\,G\odot_4 pQQp\right)\left|n^{(0)}\right\rangle$$

and so we'd expect basically our numbers of terms to grow like a 10D tensor. But in reality, focussing for now on just the elements of

$$\left\langle n^{(0)}\right|\,pQp\Pi_n\,QQQ\Pi_n\,pQQp\left|n^{(0)}\right\rangle$$

we have a much smaller set of allowed bits. We know this because starting from the left, we note that pQp can change one mode by ±3, ±2, or ±1, one mode by ±2 and another mode by ±1, or three modes by ±1.

How do we know this? Well we can look at the choices of 3 indices and just keep the unique ones

In[1564]:=
```
popTerms[opDim_] :=
  Sort@Tally[#][[ ;; , 2]] & /@ Tuples[Range[opDim], opDim] // DeleteDuplicates;
pQpPerms = popTerms[3]
```

Out[1565]=
```
{{3}, {1, 2}, {1, 1, 1}}
```

and then we can expand each of those in terms of their # quanta changes

In[1572]:=
```
popChanges // Clear
popChanges[n_Integer] :=
  PlusMinus /@ Range[Mod[n, 2], n, 2];
popChanges ~ SetAttributes ~ Listable;
popTups[changes_] :=
  Tuples /@ changes // Apply[Join] // DeleteCases[{PlusMinus[0] ..}];
popTups@popChanges@pQpPerms
```

Out[1576]=
```
{{±1}, {±3}, {±1, ±0}, {±1, ±2}, {±1, ±1, ±1}}
```

this tells us where we can be after applying $QQQ\Pi_n\, pQQp$. So to work backwards, we'll then see what QQQ can do (spoiler alert: it has the same effect as pQp)

In[1578]:=
```
popTups@popChanges@popTerms[3]
```

Out[1578]=
```
{{±1}, {±3}, {±1, ±0}, {±1, ±2}, {±1, ±1, ±1}}
```

so now this gives us restrictions possible space for $pQQp$ since we know that after applying QQQ we need to be in a state like

$$\left\{\left|n_i\pm1\right\rangle,\ \left|n_i\pm3\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle\right\}$$

being lazy for now and just taking the Cartesian product of operations pQp and QQQ, and eliminating duplicates & the ones forbidden by the perturbation operator (at least one mode must change), we have

$$\begin{gathered}\left\{\left|n_i\pm1\right\rangle,\ \left|n_i\pm3\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle\right\}\otimes\\ \left\{\left|n_i\pm1\right\rangle,\ \left|n_i\pm3\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm1\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle\right\}=\\ \left\{\left|n_i\pm2\right\rangle,\ \left|n_i\pm4\right\rangle,\ \left|n_i\pm2\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm2\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle,\ \left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle,\right.\\ \left|n_i\pm6\right\rangle,\ \left|n_i\pm4\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm4\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle,\\ \left.\left|n_i\pm2\right\rangle\left|n_j\pm4\right\rangle,\ \left|n_i\pm2\right\rangle\left|n_j\pm3\right\rangle\left|n_k\pm1\right\rangle\right\}\end{gathered}$$

now we know that pQQp can only do up to 4 quanta changes, so we can immediately pare the possibilities down to

$$\left\{\left|n_i\pm2\right\rangle,\ \left|n_i\pm4\right\rangle,\ \left|n_i\pm2\right\rangle\left|n_j\pm2\right\rangle,\ \left|n_i\pm2\right\rangle\left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle,\ \left|n_j\pm1\right\rangle\left|n_k\pm1\right\rangle\right\}$$

then looking at what $pQQp$ can do

In[1588]:=
```
popTups@popChanges@popTerms[4]
```

Out[1588]=
```
{{±2}, {±4}, {±1, ±1}, {±1, ±3}, {±0, ±2}, {±2, ±0},
 {±2, ±2}, {±1, ±1, ±0}, {±1, ±1, ±2}, {±1, ±1, ±1, ±1}}
```

we see that we can't eliminate any of the existing possibilities, but at least we *can* eliminate some of what $pQQp$ can do.

So that's all well and good, but how can we do this programmatically? Well basically we can write an algorithm to do what we just did, but programmatically. It will start from the left-most

operator, determine which transformations it can do, then take a reduced Cartesian product with the next operator where two conditions are satisfied

1. the identity transform is forbidden
2. the total number of quanta changed is less than or
   equal to the sum of the dimension of the remaining transforms

we do this until the second to last operator, then we figure out which of these transformations have inverses in the ones supported by the final operator

```
pertPopTerms // Clear
pertPopTerms[opDims : {_Integer, __Integer}] :=
 Block[
  {
   getTerms, getChanges, getTransfs,
   possibleTransfs, reduceTransfs,
   checkTerm,
   reducedProduct,
   terms, finale
  },
  getTerms[n_Integer] :=
   getTerms[n] =
    DeleteDuplicates[Sort@Tally[#][[ ;; , 2]] & /@ Tuples[Range[n], n]];
  getTerms[n_Integer, dim_] :=
   PadRight[#, dim, 0] & /@ getTerms[n];
  getChanges ~ SetAttributes ~ Listable;
  getChanges[n_Integer] :=
   getChanges[n] = Range[-n, n, 2];
  getTransfs[changes_] := (* we keep all the permutations
    in memory because this will allow for better parallelization *)
   Developer`ToPackedArray@Apply[Join, Tuples /@ changes];
  possibleTransfs[dim_] :=
   getTransfs@getChanges@getTerms[#, #] &@@dim;
  reduceTransfs[transfs_] :=
   Sort@DeleteCases[#, 0] & /@ transfs // DeleteDuplicates // DeleteCases[{}];
  checkTerm[a_, b_, dim_] :=
   Block[
    {
     d = Min[{Length[a], Length[b]}],
     mD = Max[{Length[a], Length[b]}],
     shared,
     sdim,
     rest,
     rdim
```

```
    },
    shared = a[[ ;; d]] + b[[ ;; d]];
    rest = If[Length[a] > d, a[[d + 1 ;;]], If[Length[b] > d, b[[d + 1 ;;]], {}]];
    sdim = Total[Abs@shared];
    rdim = Total[Abs@rest];
    If[rdim + sdim > dim || rdim + sdim == 0,
     Table[dim + 1, {i, mD}],
     Join[shared, rest]
    ]
   ];
  reducedProduct[termsA_, termsB_, dim_] :=
   With[{mD = Max@{Length[termsA[[1]]], Length[termsB[[1]]]}},
    Block[{set = <|Table[dim + 1, {i, mD}] → 1|>},
     Do[set[checkTerm[a, b, dim]] = 1, {a, termsA}, {b, termsB}];
     Rest@Keys[set](* drops the null term *)
    ]
   ];
  terms = possibleTransfs[opDims[[1]]];
  Do[
   (*Echo[Total@opDims[[i+1;;]], i];*)
   terms =
    reducedProduct[
     terms,
     possibleTransfs[opDims[[i]]],
     Total@opDims[[i + 1 ;;]]
    ],
   {i, 2, Length[opDims] - 1}
  ];
  finale = getTransfs@getChanges@getTerms[opDims[[-1]]];
  Intersection[ (* assuming totally symmetric transformations this works... *)
   reduceTransfs[terms],
   reduceTransfs@finale
  ]
 ]
```

In[1727]:=
```
pertPopTerms[{3, 3}]
```

Out[1727]=
```
{{-3}, {-1}, {1}, {3}, {-2, -1}, {-2, 1}, {-1, 2},
 {1, 2}, {-1, -1, -1}, {-1, -1, 1}, {-1, 1, 1}, {1, 1, 1}}
```

In[1731]:=
```
pertPopTerms[{6, 3, 4}]
```

Out[1731]=
```
{}
```

```
In[1732]:= pertPopTerms[{3, 3, 4}]
Out[1732]= {{-4}, {-2}, {2}, {4}, {-3, -1}, {-3, 1}, {-2, -2}, {-2, 2},
  {-1, -1}, {-1, 1}, {-1, 3}, {1, 1}, {1, 3}, {2, 2}, {-2, -1, -1},
  {-2, -1, 1}, {-2, 1, 1}, {-1, -1, 2}, {-1, 1, 2}, {1, 1, 2}}

In[1733]:= pertPopTerms[{5, 3, 4}]
Out[1733]= {{-4}, {-2}, {2}, {4}, {-3, -1}, {-3, 1}, {-2, -2}, {-2, 2}, {-1, -1},
  {-1, 1}, {-1, 3}, {1, 1}, {1, 3}, {2, 2}, {-2, -1, -1}, {-2, -1, 1},
  {-2, 1, 1}, {-1, -1, 2}, {-1, 1, 2}, {1, 1, 2}, {-1, -1, -1, -1},
  {-1, -1, -1, 1}, {-1, -1, 1, 1}, {-1, 1, 1, 1}, {1, 1, 1, 1}}

In[1734]:= pertPopTerms[{3, 3, 3, 3, 4}]
Out[1734]= {{-4}, {-2}, {2}, {4}, {-3, -1}, {-3, 1}, {-2, -2}, {-2, 2},
  {-1, -1}, {-1, 1}, {-1, 3}, {1, 1}, {1, 3}, {2, 2}, {-2, -1, -1},
  {-2, -1, 1}, {-2, 1, 1}, {-1, -1, 2}, {-1, 1, 2}, {1, 1, 2}}

In[1735]:= pertPopTerms[{3, 3, 3, 3}]
Out[1735]= {{-3}, {-1}, {1}, {3}, {-2, -1}, {-2, 1}, {-1, 2},
  {1, 2}, {-1, -1, -1}, {-1, -1, 1}, {-1, 1, 1}, {1, 1, 1}}
```

By doing this, we're able to get the kinds of total transformations that are supported...which isn't actually quite the question we wanted to answer initially (but it still useful as it gives a sense for the number of index combos we need to consider). What we really want is to know which operators contribute...to do that, we actually need to keep more information as we consider our cases. In particular, when we do our paring of the Cartesian products of terms, we need to track which set of indices the terms came from, which will allow us to track what changes we have overall.

Inverting The X-Matrix

An alternate approach to get things like the X matrix and Y tensor is to simply generate enough perturbation theory results that we can invert to get them.

We'll write out the energy in general as

$$E_{\boldsymbol{n}}^{(o)} = \sum_{k=0}^{m} \epsilon_k^{(o)} \odot_k \left(\boldsymbol{n} + \frac{1}{2} \right)^k$$

where $\epsilon_k^{(o)}$ is the k^{th} order tensor correction to the energy at the o order of PT. The exact relationship between the order of PT we use and the max m isn't totally obvious, but at the very least we know that we get an extra order of correction for every 2 orders of PT.

We can string this all out into one rather long and involved summation as

$$E_{\boldsymbol{n}}^{(o)} = \sum_{k=0}^{m} \sum_{p \in \text{inds}(k)} \left(\epsilon_k^{(o)}\right)_p \prod_{i \in p} \left(n_i + \frac{1}{2}\right)$$

where $\text{inds}(k)$ is just the set of indices appropriate for the k^{th} energy correction tensor.

Why is this nice? Well if we have energies for enough sets of $\boldsymbol{n}$, we can string them together and use

$$C_{\boldsymbol{n}p} = \prod_{i \in p} \left(n_i + \frac{1}{2}\right)$$

to get a big ol' set of linear equations that we can directly invert. This will give us a vector representation of $\left(\epsilon_k^{(o)}\right)_p$ that we'll need to reshape into its proper tensors.

We'll make a replacement where we say

$$a_i = n_i + \frac{1}{2}$$

which is nice because we can then write

$$E_{\boldsymbol{n}}^{(o)} = \sum_{k=0}^{m} \epsilon_k^{(o)} \odot_k \boldsymbol{a}^k$$

which is just notationally simpler. It should be noted that the $\boldsymbol{a}^k$ is the iterated outer product of the $\boldsymbol{a}$ vector with itself.

First-Order Correction

We'd like to be able to back-out what the $\epsilon_k^{(o)}$ terms look like from the energies, assuming this ansatz. We'll start with just the first-order terms and see if we can work out what's going on there.

To do that, we'll start with

$$\begin{aligned} E_{n_i}^{(o)} - E_0^{(o)} &= \sum_{k=0}^{m} \epsilon_k^{(o)} \odot_k \left(\left(n_i \delta_i + \frac{1}{2} \right)^k - \left(\frac{1}{2} \right)^k \right) \\ &= \sum_{k=1}^{m} \sum_{p \in \text{inds}(k)} \left(\epsilon_k^{(o)}\right)_p \left(\left(n_i + \frac{1}{2} \right)^{\text{count}(p,\, i)} \left(\frac{1}{2} \right)^{\frac{1}{2}(k - \text{count}(p,\, i))} - \right) \\ &= \end{aligned}$$

and to make sense of this, we note that we can write

$$\boldsymbol{a}^k = \boldsymbol{a} \otimes \boldsymbol{a}^{k-1}$$

where $\otimes$ is a Kronecker product. Then if we have

$$\boldsymbol{b} = \boldsymbol{a} + \boldsymbol{\Delta}$$

we get

$$\begin{aligned}\boldsymbol{b}^k&=(\boldsymbol{a}+\boldsymbol{\Delta})\otimes\boldsymbol{b}^{k-1}\\&=\boldsymbol{a}\otimes\boldsymbol{b}^{k-1}+\boldsymbol{\Delta}\otimes\boldsymbol{b}^{k-1}\end{aligned}$$

or in other words, building up combinatorially

$$\begin{aligned}\boldsymbol{b}&=\boldsymbol{a}+\boldsymbol{\Delta}\\\boldsymbol{b}^2&=(\boldsymbol{a}+\boldsymbol{\Delta})\otimes(\boldsymbol{a}+\boldsymbol{\Delta})\\&=\boldsymbol{a}^2+\boldsymbol{a}\bigotimes\boldsymbol{\Delta}+\boldsymbol{\Delta}\bigotimes\boldsymbol{a}+\boldsymbol{\Delta}^2\\\boldsymbol{b}^3&=(\boldsymbol{a}+\boldsymbol{\Delta})\otimes(\boldsymbol{a}+\boldsymbol{\Delta})\otimes(\boldsymbol{a}+\boldsymbol{\Delta})\\&\dots\end{aligned}$$

Not Quite Right Stuff (But Useful Ideas)

State Indexing

We need to be able to do efficient indexing of states, which can be expressed as vectors of quanta of excitation in our different modes. The basic tack we'll take is to order by number of quanta of excitation, then lexicographically based on the partition of the total number of quanta, and finally lexicographically based for the permutation of that partition.

This has many nice features, for one, the ground state is always the first state, the states are ordered in terms of quanta of excitation, and then states are ordered by how few modes they touch. On the other hand, these numbers will still grow quite quickly.

For a given number of total quanta of excitation, there will be some number of integer partitionings. For each of those partitionings, the number of distinct permutations is given by

$$\text{perms}(p)=\frac{D!}{(D\text{-}|p|)!\prod_i \#p_i}$$

where $\#p_\text{i}$ gives the number of times p_i appears in the partition.

We can write a function that counts these like

In[889]:=
```
numPerms[p_, dim_] :=
 dim! / Apply[Times,
   Factorial[
    Append[
     Values[Counts[p]],
     dim - Length[p]
    ]
   ]
  ]
numStates[nq_Integer, dim_] :=
 Sum[
  numPerms[p, dim],
  {p, IntegerPartitions[nq]}
 ]
```

which allows us to figure out, say, how many 10 quantum states there are in a system of dimension 100

In[918]:=
```
2^64 - Sum[numStates[i, 377], {i, 10}]
```

Out[918]= −21 640 509 651 853 999

In[909]:=
```
Sum[numStates[i, 350], {i, 10}]
```

Out[909]= 8 881 600 973 913 295 055

In[907]:=
```
350 / 3.
```

Out[907]= 116.667

In[908]:=
```
(120 * 3) - 6
```

Out[908]= 354

In[905]:=
```
Sum[numStates[i, 350], {i, 10}]
```

Out[905]= 8 881 600 973 913 295 055

or how many ≤10 quanta states there are at dimension 100

In[1687]:=
```
Sum[numStates[i, 100], {i, 10}]
```

Out[1687]= 46 897 636 623 980

(we can see that it's really dominated by those 10 quantum states)

Overall, this is a plus, as it means we can index up to large-ish numbers of quanta in a "global" fashion.

Generating Indices

The next issue we have is actually generating the indices we'll need. To do this, let's consider the

concrete case of the state (1, 4, 3, 1, 0, 0, 0, 1, 0) which is in 9D space, so maybe it's like for methane or something. Our first step will be to get the total number of quanta in the state

$$\begin{aligned}\#\text{quanta} &= 1{+}4{+}3{+}1{+}1 \\ &= 10\end{aligned}$$

then we reverse sort the list (alternately, sort its negation)

$$\text{revsorted}(s){=}(4,\ 3,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0)$$

and find how many integer partitions came before it

```
In[1698]:= TakeWhile[IntegerPartitions[10], # ≠ {4, 3, 1, 1, 1} &]
Out[1698]= {{10}, {9, 1}, {8, 2}, {8, 1, 1}, {7, 3}, {7, 2, 1}, {7, 1, 1, 1},
  {6, 4}, {6, 3, 1}, {6, 2, 2}, {6, 2, 1, 1}, {6, 1, 1, 1, 1}, {5, 5},
  {5, 4, 1}, {5, 3, 2}, {5, 3, 1, 1}, {5, 2, 2, 1}, {5, 2, 1, 1, 1},
  {5, 1, 1, 1, 1, 1}, {4, 4, 2}, {4, 4, 1, 1}, {4, 3, 3}, {4, 3, 2, 1}}
```

and then determine how many states they represent

```
In[1706]:= Total[numPerms[#, 9] & /@ TakeWhile[IntegerPartitions[10], # ≠ {4, 3, 1, 1, 1} &]]
Out[1706]= 15 831
```

adding on the number of states with fewer than 10 quanta

```
In[1712]:= Sum[numStates[i, 9], {i, 9}] +
  Total[numPerms[#, 9] & /@ TakeWhile[IntegerPartitions[10], # ≠ {4, 3, 1, 1, 1} &]]
Out[1712]= 64 450
```

and now we have our *baseline* for getting the index. That was the easy part.

Next when do our reverse sorting, we can capture a permutation that orders the partition

```
In[1713]:= Ordering[{1, 4, 3, 1, 0, 0, 0, 1, 0}] // Reverse
Out[1713]= {2, 3, 8, 4, 1, 9, 7, 6, 5}
```

we now need to provide a way to index permutations.

A classic algorithm is the *Lehmer code* which computes a bit string representation of an index, or alternately a representation in a factorial base. This is constructed iteratively by recognizing that given a permutation like

$$p{=}(p_1,\ p_2,\ \ldots\ )$$

at each step there are

The issue with this is that it doesn't account for the lack of uniqueness of the various terms. OTOH, it's a good starting point for what we're going. So first, we calculate the Lehmer code, which will

Tree Viz

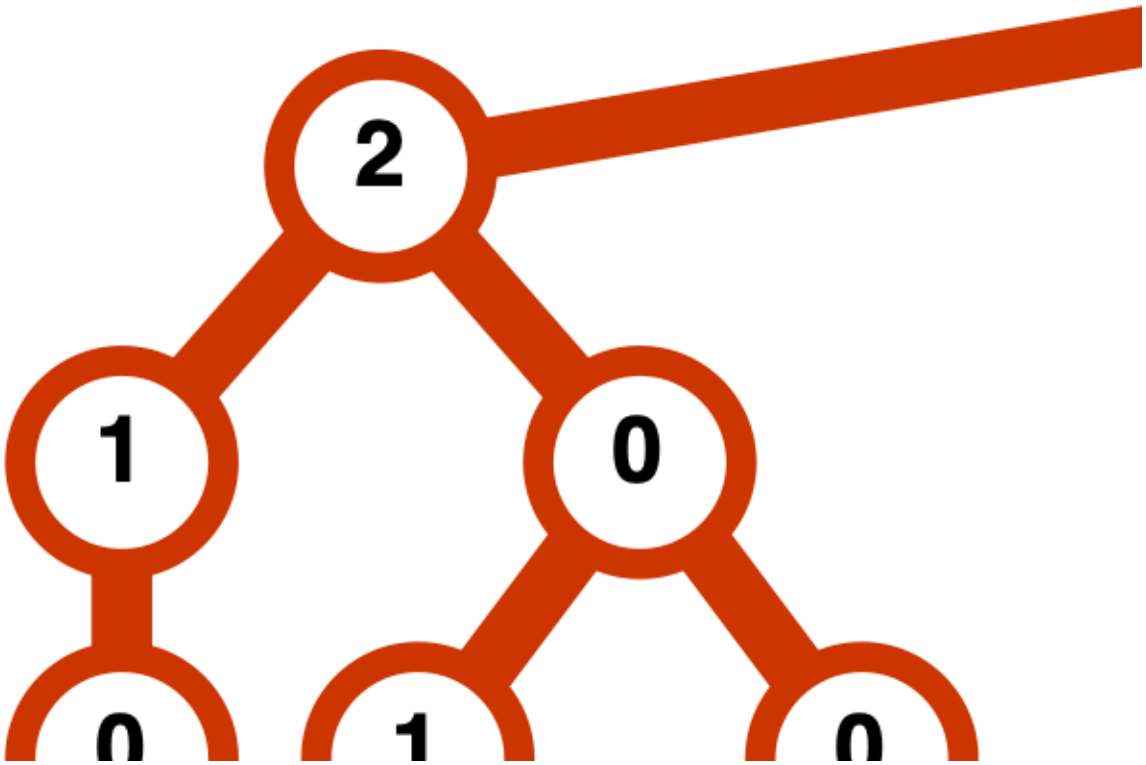
In[978]:=
```
baseTree = GroupBy[
    Permutations[{2, 1, 0, 0, 0}],
    First → Rest,
    Nest[
     Function[{g}, GroupBy[#, First → Rest, g] &],
     GroupBy[First → Rest],
     2
    ]
   ];
normTree = Nest[Normal, baseTree, 5];
simpTree = List[normTree //. Rule[a_, b_] :> a[b] //. List[x__] :> x];
```

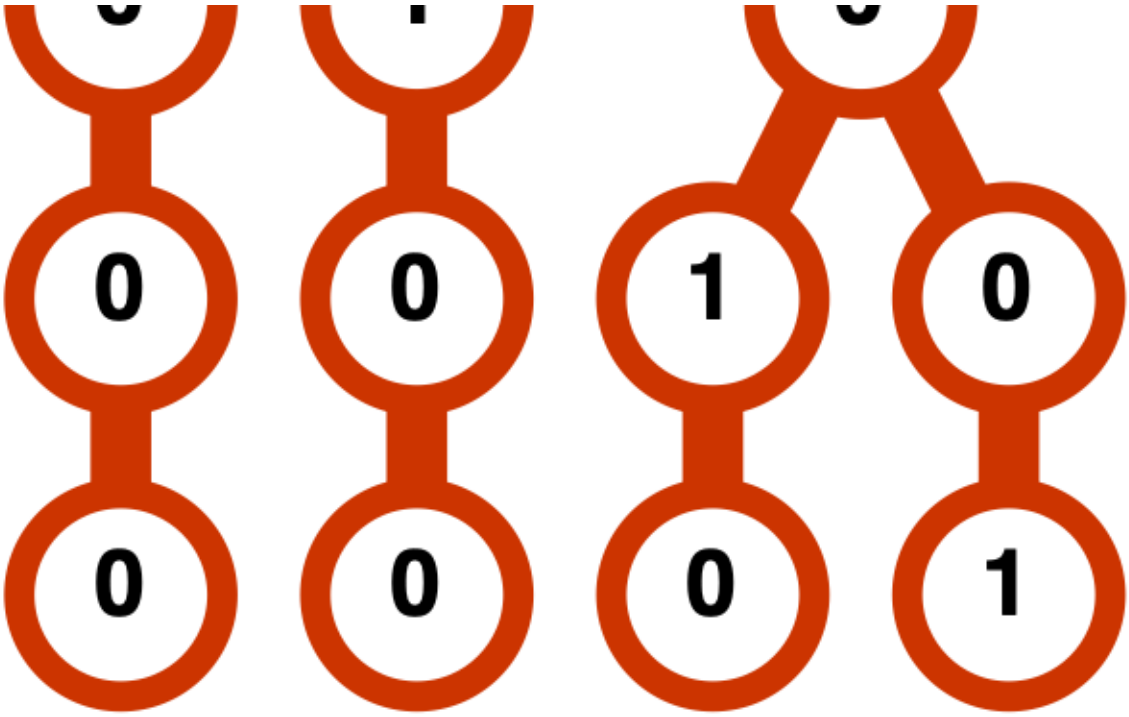
In[962]:=
```
makeTree[nodes_, renderFunc_] := Module[{counter = 0},
  traverse[h_[childs___]] := With[{id = counter}, {UndirectedEdge[id, ++counter],
       counter → renderFunc[#], traverse[#]} & /@ {childs}];
  traverse[_] := Sequence[];
  TreeGraph[#1, VertexShape → #2] & @@
   Transpose[Partition[Flatten[traverse[nodes]], 2]]]
```

In[1029]:=
```
labStyle =
  Style[
    #,
    FontSize → 24,
    FontWeight → Bold,
     FontColor → Black, FontFamily → "Helvetica"] &;
initTree =
  makeTree[
   simpTree,
   labStyle@If[Head[#] === Integer, #, Head[#]] &
  ];
valMap =
  PropertyValue[
   initTree,
   VertexShape
  ];
```

In[1032]:=
```
TreePlot[
  initTree,
  PlotTheme → "Web",
  VertexShapeFunction →
   (
    {
      {White, Disk[#, .2]},
      {
       Black,
       Inset[Lookup[valMap, #2, labStyle@"{}"], #]
      }
     } &
   ),
  ImageSize → {Automatic, 500}
 ] // imageify
```

Out[1032]=




0
0
0
0
1
0
0
1