11/9/04 - routines to calculate clebsch-gordon coefficients. simple iterative procedure following development in Phys 811.

- Intro

cg[ja_, jb_] - Build a list of representations for $j = j_{\text{max}}$, $j_{\text{min}}$ with $j_{\text{max}} = j_1 + j_2$ and $j_{\text{min}} = |j_1 - j_2|$. Each representation is a list of states with $-j \leq m \leq j$. Each $|jm\rangle$ state is a linear combination of $|m_1 m_2\rangle$ states, weighted by the appropriate clebsch gordon coefficients. Beginning with the $j_{\text{max}}$ representation, the program finds the highest $m$ state for the current $j$-representation, and builds the full $j$-rep by repeated applications of $J_\pm$. After completing the ep, $j$ is lowered. The highest $m$-state for that next $j$-rep is the state which is orthogonal to all the other $m$-states from higher $j$ representations.

The result is given as a list of representations from $j = j_{\text{max}}$ to $j = j_{\text{min}}$. Each representation is given as a list of $m$-states,

rep = {{j, j, {m1 m2 list}}, {j, j - 1, {m1 m2 list}}, ... {j, j, {m1 m2 list}}}. Each state $|j, m\rangle$ is defined by the cg-coefficients which weight the $|m_1 m_2\rangle$ states from which it is built, $|m_1 m_2\rangle = \{|m_1_{\text{min}}, m - m_{1 \text{min}}\}, c_{j m m_1 m_2}^j, \{|m_1_{\text{min}} - 1, m - m_{1 \text{min}} + 1\}, c_{j m m_1 m_2}^{j-1}, ... \{|m_{1 \text{max}}, m - m_{1 \text{max}}\}, c_{j m m_1 m_2}^{j_{\text{max}}}$

```math
CG[ja_, jb_] := Block[{},
j1 = Max[ja, jb];
j2 = Min[ja, jb];
jmax = j1 + j2;
jmin = j1 - j2;
replist = {};
j = jmax;
head = {{{j1, j2}, 1}};
While[j > jmin,
newrep = buildjrep[j1, j2, j, head];
AppendTo[replist, newrep];
j = j - 1;
ids = getmid[#, j, @replist;]
states = Table[replist[[i]][[ids[[i]]]][[3]], {i, jmax - j}];
ms = First @@ states[[1]];
coeffs = (Last @@ #) & @@ states;
newcs = getorthogonalstate[coeffs];
head = Transpose[{ms, newcs}];
newrep = buildjrep[j1, j2, j, head];
AppendTo[replist, newrep];
replist]
```
Given \( j_1 \) \( j_2 \) find coefficients for \( |jm\rangle = c_{jm,m_1,m_2}^{j_1 j_2} |m_1 \ m_2\rangle \)

In[1]:= cminus[j_, m_] := If[Abs[m] <= j, Sqrt[(j + m) (j - m + 1)], 0]

getmid[rep, m] - finds the index of m-state within the j-rep

getmid[rep_, m_] := select[rep, #[[2]] == m &][[1]]

Loads GramSchmidt orthogonalization routine

In[77]:= << LinearAlgebra`Orthogonalization`

gethogonalstate[coeffs] - coeffs holds n-1 vectors each n elements long. getorthogonalstate finds an nth-state orthogonal to the other n.

gethortalogonalstate[coeffs_] := Block[{}, n = Length[coeffs]; newc = Prepend[0 Range[n], 1]; Simplify[Last[GramSchmidt[Append[coeffs, newc]]]]]

buildjrep - given maximal weight and initial state, build remaining states by application of jminus

In[25]:= buildjrep[j1_, j2_, j_, jmlist_] := Block[{}, m = j; jmlista = {j, m, jmlist}; jlist = {jmlista}; While[m > -j, jmlista = applyjminus[j1, j2, jmlista]; AppendTo[jlist, jmlista]; m = m - 1]; jlist]

applyjminus- given \( j_1, j_2, j, m \), and the list of coefficients for the state stored in jmlist, apply \( J_- \) to produce new jmlist for state \( j_1, j_2, j, m-1 \). elements are stored in order of increasing m1. format is \( \{\{m1,m2\},c\},\{\{m1,m2\},c\},\{\{m1,m2\},c\}...\} \). Depending on the value of \( m \), the new state may have one more, the same, or one less component than the original state.
In[114]:= applyjminus[j1_, j2_, {j_, m_, jmlist_}] := Block[{},
cjm = cminus[j, m];
n = Length[jmlist];
jmlist1 = cminus[j1, #][[1, 1]] & /@ jmlist;
AppendTo[jmlist1, 0];
jmlist2 = cminus[j2, #][[1, 2]] & /@ jmlist;
PrependTo[jmlist2, 0];
jmlistc = Simplify[(jmlist1 + jmlist2) / cjm];
m1m2list = (# - (1, 0)) & /@ jmlist[[All, 1]];
AppendTo[m1m2list, Last[m1m2list] + {1, -1}];
newjmlist = Transpose[{m1m2list, jmlistc}];
mlmin = First[jmlist][[1, 1]];
m2min = Last[jmlist][[1, 2]];
newjmlist = If[mlmin == -j1, Drop[newjmlist, 1], newjmlist];
newjmlist = If[m2min == -j2, Drop[newjmlist, -1], newjmlist];
{j, m - 1, newjmlist}]

Tests

Test applyjminus by building the $j = 2, m = 1$ state from the $j = 2, m = 2$ state in the case where $j_1 = j_2 = 1$.

In[10]:= applyjminus[1, 1, {2, 2, {{1, 1}, 1}}]

Out[10]= {2, 1, {{0, 1}, 1/Sqrt[2]}, {1, 0}, 1/Sqrt[2]}}

Repeat to find the $j = 2, m = 0$ state.

In[11]:= applyjminus[1, 1, %]

Out[11]= {2, 0, {{-1, 1}, 1/Sqrt[6]}, {0, 0}, {2/3}, {1, -1}, 1/Sqrt[6]}}

Test buildjrep by building the $j = 2$ rep in the product where $j_1 = j_2 = 1$. 
test cg to make product of $1 \times 1$

entries in table - each row is a different jrep, with sequential m-states. For each m-state, a list of m1,m2 pairs with the cg coefficient underneath.

Here is the same result in numerical form.
Here is the results for \(5 \times \frac{7}{2}\)

I've trimmed the output to just give the values of \(j,m\) and the list of \(cg\)'s, with the \(m1,m2\) values understood.
TableForm[{{1, 2}, Last/@{{3}}}] &/@@ &/cetable

Again the same result with real coefficients

cgtable = cg[5., 7/2];
TableForm[
  Table[
    Table[
      If[
        Plus @@ Abs[1 - #] < 1, #, 0]
      , {3, Last[TableForm]}]
    , {3, TableForm}]
]