Homework 4: Due April 26

Entanglement and Tensor Networks, Spring 2016, Prof. White

1. Define the A[s] tensors in a matrix product state in Julia as an array of 3-index arrays

```
A = [zeros(1,2,1) for i=1:n]
```

Here all bond dimensions are $m = 1$, and the first and last tensors, which ordinarily have only two indices, have 3, just to make all A's look similar. The extra indices on A[1] and A[n] will always have a range of 1:1, so they don't really do anything. We can initialize an up-down-up-down... product state this way:

```
for i=1:n
    A[i][1,iseven(i) ? 2 : 1,1] = 1.0
end
```

Write functions normsq1(A), normsq2(A), and normsq3(A) each to calculate the norm squared of psi, $\langle\psi|\psi\rangle$. The basic algorithms will all be the same, working left to right to contract the tensor network. In normsq1, use for loops to do all the tensor contractions by summing over the indices you want to sum over. In normsq2, use reshape and permutedims to convert each pair of tensors that you want to contract to matrices, and then do the matrix multiply with "C = A * B", then reshape/permute to whatever index order you next need. In normsq3, use the TensorOperations Package in Julia to do each contraction with an index summation convention, e.g.

```
Ai = A[i]
@tensor begin
Op[b,bp] := O[a,ap] * Ai[a,s,b] * Ai[ap,s,bp]
end
```

Test all three versions on the simple product state above to see that they all give 1. Note that normsq1 would be very slow once the tensors got a little bigger.

2. Now let's generalize our MPS to include an orthogonality center. Define an MPS type using

```
type MPS
    A
    oc::Int64
end
```

Write a function "moveto!(psi::MPS, i::Int64)" that moves the OC from wherever it is to site i. The function can assume that the original MPS had a valid OC. It should use the QR algorithm to move the OC one site at a time, either to the left or to the right. Test the function using one of your normsq functions, to see that it stays normalized.

3. Write a function "energybond(psi::MPS, i::Int64)" to calculate $\langle \text{psi}|\vec{S}_i \cdot \vec{S}_{i+1}|\text{psi}\rangle$. It should use the moveto function to move to site i, and do the minimal amount of contractions near that site, using TensorOperations. It can use Htwosite defined in the next problem.

4. Here is code to make a two-site gate that projects towards the ground state

```
sz = Float64[0.5 0; 0 -0.5] ; sp = Float64[0 1; 0 0] ; sm = sp'
Htwosite = Float64[sz[s1,s1p] * sz[s2,s2p] + 0.5 * (sp[s1,s1p] * sm[s2,s2p] +
    sm[s1,s1p] * sp[s2,s2p]) for s1=1:2, s2=1:2, s1p=1:2, s2p=1:2]
tau = 0.001
taugate = reshape(expm(-tau * reshape(Htwosite,4,4)),2,2,2,2)
```

Here is a part of a program to do the TEBD algorithm to find the ground state, starting with our Neel product state

```
for i=1:n-1
    Ai = psi.A[i]
    Ai1 = psi.A[i+1]
    @tensor begin
        AA[a,f,g,e] := Ai[a,b,c] * Ai1[c,d,e] * taugate[b,d,f,g]
    end
    (psi.A[i],psi.A[i+1]) = dosvdtoright(AA,m)
end
```

Write the rest of this TEBD program, including the dosvdtoright function, and a simlar dosvdtoleft function, which splits up the AA tensor using an SVD, putting the D matrix either on the left or the right, to move the OC. Do some tests to show that it goes towards the ground state, not worrying too much about convergence.