Applying iTEBD to the tranverse-field Ising model. In this tutorial we explain the application of the infinite time evolving block decimation (iTEBD) algorithm to the transverse-field Ising model.

Generate a random matrix $G$ (i.e. $\Gamma$) with dimension $2$ (odd and even sites, $A$ or $B$) $\times d$ ($=2$, spin) $\times \chi \times \chi$ (block dimension). $l$ (i.e. $\Lambda$) is a diagonal matrix for $A$ and $B$ sites represented by a random vector.

```python
import numpy as np
from scipy import integrate

# First define the parameters of the model / simulation
J=1.0; g=0.5; chi=5; d=2; delta=0.005; N=3000;
G = np.random.rand(2,d,chi,chi);l = np.random.rand(2,chi)
```

The Hamiltonian describes a transverse field Ising model on two sites. The diagonal term is $J\sigma_j^z\sigma_{j+1}^z$, while the off-diagonal terms are $-(g/2)(\sigma_j^x + \sigma_{j+1}^x)$. The basis is $\{|\uparrow\uparrow\rangle,|\uparrow\downarrow\rangle,|\downarrow\downarrow\rangle\}$.

```python
# Generate the two-site time evolution operator
H = np.array( [[J,-g/2,-g/2,0], [-g/2,-J,0,-g/2], [-g/2,0,-J,-g/2], [0,-g/2,-g/2,J]] )
w,v = np.linalg.eig(H)
U = np.reshape(np.dot(np.dot(v,np.diag(np.exp(-delta*(w)))),np.transpose(v)),(2,2,2,2))
```

To obtain the ground state energy, we perform the imaginary-time evolution of the randomly selected state.

```python
# Perform the imaginary time evolution alternating on A and B bonds
for step in range(0, N):
    A = np.mod(step,2); B = np.mod(step+1,2)
```

FIG. 1: iTEBD step (i): Construct $\Theta$.

```python
# Construct theta
theta = np.tensordot(np.diag(l[B,:]),G[A,:,:,:],axes=(1,1))
theta = np.tensordot(theta,np.diag(l[A,:]),axes=(2,0))
theta = np.tensordot(theta,G[B,:,:,:],axes=(2,1))
theta = np.tensordot(theta,np.diag(l[B,:]),axes=(3,0))
```

FIG. 2: iTEBD step (ii): Apply time evolution.

```python
# Apply imaginary-time evolution operator $U$
theta = np.tensordot(theta,U,axes=((1,2),(0,1)))
```
Now we decompose the two-site tensor back to tensors for single sites. First we perform a singular-value decomposition to decompose the two-site tensor. Then, we insert $\Lambda_B$ and bring the tensors back to the standard format. This way, we have updated $\Gamma_A$, $\Lambda_A$, $\Gamma_B$, but not $\Lambda_B$. In the next iteration, we swap $A$ and $B$.

# Perform singular-value decomposition
theta = np.reshape(np.transpose(theta,(2,0,3,1)),(d*chi,d*chi));
X, Y, Z = np.linalg.svd(theta); Z = Z.T

# Truncate the bond dimension back to chi and normalize the state
l[0:chi]=Y[0:chi]/np.sqrt(sum(Y[0:chi]**2))
X=np.reshape(X[0:d*chi,0:chi],(d,chi,chi))
G[0:chi,:,:,0]=np.transpose(np.tensordot(np.diag(l**(1/chi)),X,axes=(1,1)),(1,0,2));
Z=np.transpose(np.reshape(Z[0:d*chi,0:chi],(d,chi,chi)),(0,2,1))
G[0:chi,:,:,1]=np.tensordot(Z,np.diag(l**(1/chi)),axes=(2,0));

After sufficiently many iterations, the tensors converge to the values corresponding to the ground state of the system. We can compare the ground state energy to the exact solution.

print "E_iTEBD =", -np.log(np.sum(theta**2))/delta/2
f = lambda k,g : -2*np.sqrt(1+g**2-2*g*np.cos(k))/np.pi/2.
E0_exact = integrate.quad(f, 0, np.pi, args=(g,))[0]
print "E_exact =", E0_exact

The python code was kindly provided by Frank Pollmann (MPI-PKS). You are encouraged to install the Anaconda python package to run the program. A nice review of the iTEBD algorithm, as well as its comparison with the infinite-size density matrix renormalization group (iDMRG) algorithm, can be found in arXiv:1212.6255 by Pollmann and his collaborators.

Midterm project: Understand the MPS formalism and the iTEBD algorithm. Adapt the python code to solve Heisenberg antiferromagnetic chains, for either spin-1/2s or spin-1s. Solve the entanglement spectrum (i.e., the set of values at the diagonal of $\Lambda$) of the spin-1 chain and show that there is a gap in the spectrum separating two degenerate levels from the rest. Compare the results with the spin-1 AKLT chain.