THE ASTOUNDING VARIATIONAL INFINITE MPS ALGORITHM

GREGORY CROSSWHITE

Contents

Phase 1: Initialization 1
Step 1: Form initial environment 1
Step 2: Normalize 1
Step 3: Build environments 3

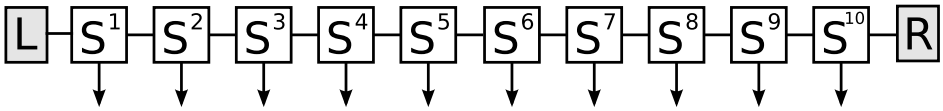
Phase 2: Iteration 4
Step 1: Prepare for optimization 4
Step 2: Perform optimization 5
Step 3: Normalize and absorb 6
Step 4: Increase χ 6

Phase 3: Compute Expectations 7

Phase 1: Initialization

Step 1: Form initial environment

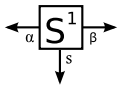
First, we use Frank Verstraete's algorithm to optimize a system with 10 sites and open boundary conditions, with $\chi = 5$.



(The purpose of this step is to enlarge the Hilbert space so that it is much bigger than the degrees of freedom in our matrix product state.)

Step 2: Normalize

On states 1-5 – tensors of the form $S_{s\alpha\beta}$, where the index s corresponds to the obsevable and the indices α and β are connections to the left and right, i.e.,



we impose the normalization condition:

$$\sum_{s\alpha} S_{s\alpha\beta} S^*_{s\alpha\beta'} = \delta_{\beta\beta'} \tag{1}$$

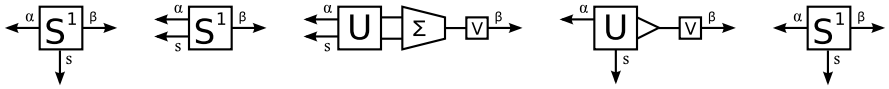
That is, we impose the condition that the right index β on this tensor is orthonormal, given that the other two indices are orthonormal. (We may impose this because the s index on all sites is orthonormal, the left index of S^1 is orthonormal, and each step orthnormalizes the right on S^i which means that the left index on S^{i+1} is orthonormalized.) To do this, we perform a singular value decomposition:

$$S^i_{(s\alpha),\beta} = \sum_{(s'\alpha'),\beta')} U_{(s\alpha)(s'\alpha')} V_{\beta'\beta},$$

where U and V are unitary and Σ is diagonal. We see that S^i obeys condition (1) iff the singular values (i.e., the nonzero elements of the diagonal matrix Σ) are 1. Thus, imposing (1) is equivalent to setting

$$S^i_{(s\alpha),\beta} = \sum_{(s'\alpha')=1}^{\chi} U_{(s\alpha)(s'\alpha')} V_{\beta'\beta}.$$

This process can be represented visually in the following manner:



Note that this is equivalent to left-multiplying S^i by the matrix,

$$V^\dagger S^{-1} V. \tag{2}$$

Ergo, after setting S^i to the above, we right-multiply S^{i+1} by the inverse of (2),

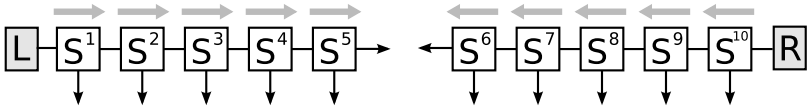
$$V^\dagger S V,$$

so that at the end of the procedure the state has not been changed.

After sites 1-5 have been normalized, we normalize sites 6-10 (in reverse order) to obey the related condition,

$$\sum_{s\beta} S_{s\alpha\beta} S^*_{s\alpha'\beta} = \delta_{\alpha\alpha'}.$$

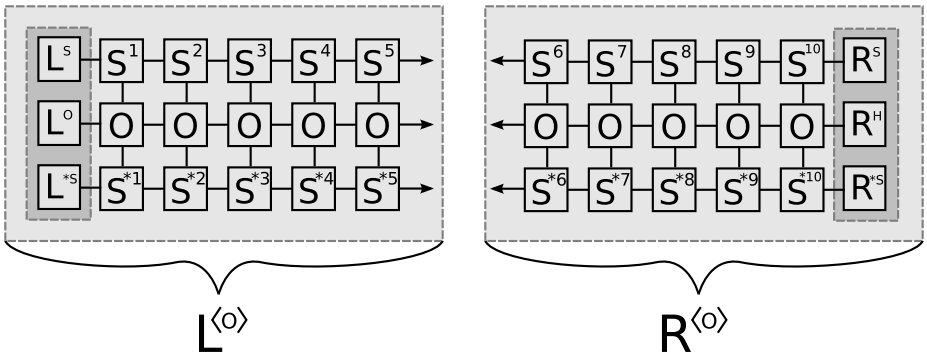
However, we do not apply the inverse normalizer to site 5 when normalizing site 6; this means that we have changed our matrix product state. Thus, at the end of this process, we obtain a state that looks like,



with the light-grey arrows indicating the normalization condition imposed on the tensors.

Step 3: Build environments

Next, we form environments that we can use the calculate the expectations of operators later. Specifically, for each operator O, which we assume has been factored into matrix-product form, we contract the following tensor networks,



to give us left and right environments, $L^{\langle O\rangle}$ and $R^{\langle O\rangle}$. We do this twice to compute environments for $\langle H\rangle$ and $\langle I\rangle$ (i.e., the expectation of the Hamiltonian and the normalization.)

We perform the contraction by first multiplying together L^S, L^O and L^{*S} as shown in the diagram above, and then absorbing S^1, O, and S^{*1}, then S^2, O, and S^{*2}, etc. until the network is contracted. (In this way we limit the computational cost to $O(\chi^3)$.) We adopt a similar strategy for the right environment

Phase 2: Iteration

Step 1: Prepare for optimization

At the beginning of this step, we have been given four tensors: $L^{\langle H\rangle}$, $R^{\langle H\rangle}$, $L^{\langle I\rangle}$, and $R^{\langle I\rangle}$. We insert a new site between each of these environments to form a new network as shown:

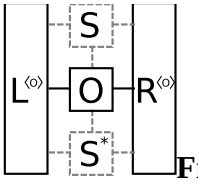
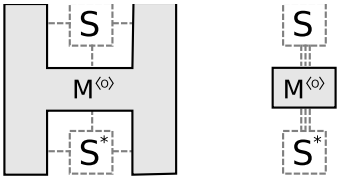


Figure 1

(The site matrix S is grayed in order to indicate that we are deferring our decision of exactly what to put in that spot.)

Contraction of this network is equivalent to computing $\langle O\rangle$. We want our new site to minimize the energy – that is, to minimize the Rayleigh quotient $\langle H\rangle / \langle I\rangle$. Observe that if we were to contract all of the tensors in this network together except for S, then we would obtain a matrix, $M^{\langle O\rangle}$,



such that $\langle O\rangle = S \cdot M^{\langle O\rangle} \cdot S^*$. Thus we see that finding S to minimize our Rayleigh quotient is equivalent to solving the generalized eigenvalue problem,

$$M^{\langle H\rangle} S = \lambda M^{\langle I\rangle} S.$$

We assume that $M^{\langle I\rangle}$ is sufficiently well conditioned that we can rewrite the above in the form,

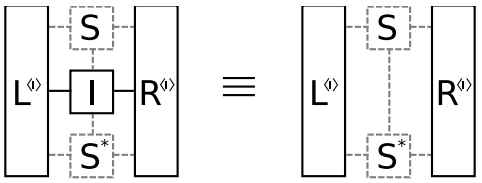
$$\left(M^{\langle I\rangle}\right)^{-1} M^{\langle H\rangle} S = \lambda S. \tag{3}$$

We use the ARPACK eigenvalue solver to find the solution to (3) with the lowest eigenvalue λ. We do not actually calculate $M^{\langle H\rangle}$ or $M^{\langle I\rangle}$, however, as this would be inefficient – a $O(d^2\chi^4)$ operation. Instead, we observe that for ARPACK, all we really need is the ability to compute the action of $\left(M^{\langle I\rangle}\right)^{-1} M^{\langle H\rangle}$ on a vector.

Note that multiplying by a matrix $M^{\langle O\rangle}$ is equivalent to inserting an S (but not S^*) into Figure 1 and then contracting all of the tensors together. Rather than first forming $M^{\langle O\rangle}$, we can instead leave the network in factored form,

and instead absorb first $L^{\langle O\rangle}$, then O, then $R^{\langle O\rangle}$ into S. The time needed to perform these contractions is $O(2cd\chi^3 + c^3d^2\chi^2)$.

This almost gives us all that we need, save that we also need to compute the action of $\left(M^{\langle I\rangle}\right)^{-1}$. To do this, we note that this has a special form: since the identity operator is a tensor product and thus has no "bonds", Figure 1 therefore takes the following form:



That is, we see that $L^{\langle I\rangle}$ and $R^{\langle I\rangle}$ actually appear in a tensor product with respect to each other, and so the inverse is just the tensor product of the inverses. To find these inverses, we take advantage of the fact that we know that both matrices are Hermitian by construction; thus, we use a standard Hermitian eigenvalue solver to decompose

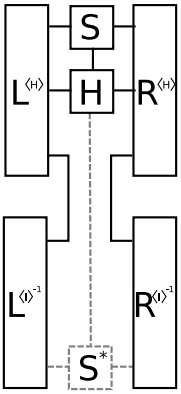
$$L^{\langle I\rangle} = U_L^\dagger D_L U_L, \quad R^{\langle I\rangle} = U_R^\dagger D_R U_R,$$

(an operation which takes $O(\chi^3)$ time) and then obtain the inverse from,

$$L^{\langle I\rangle} = U_L^\dagger D_L^{-1} U_L, \quad R^{\langle I\rangle} = U_R^\dagger D_R^{-1} U_R,$$

which is also an $O(\chi^3)$ operation.

At the end of the day, we see that computing $\left(M^{\langle I\rangle}\right)^{-1} M^{\langle H\rangle} S$ is equivalent to contracting the tensor network,



which, as before, we perform by first absorbing $L^{\langle H\rangle}$, then $\left(L^{\langle I\rangle}\right)^{-1}$, etc., so that the entire operation has time complexity $O(cd\chi^3 + c^3d^2\chi^2)$.

Step 2: Perform optimization

We run ARPACK with the following information:

- A routine to perform matrix-vector multiplication, as described in the previous section
- The result from the last run of ARPACK to be used as an initial guess (if available)
- Relative tolerance is 10^{-8}.

The result is a site matrix S which should minimize the energy.

(Sanity checks are performed to make sure that the solution is valid – i.e., that the solution when plugged into the tensor network gives the same energy as that reported by the eigenvalue solver, that the energy should not have a significant ($> 10^{-10}$) imaginary part, etc.)

Step 3: Normalize and absorb

Now we want to take our new site and absorb it into the environment. At each step the site is absorbed into a different side – i.e., if in the last iteration step we absorbed it into the left then in the next we should absorb it into the right. Before absorbtion, we normalize it; if we are absorbing into the left environment, we have to normalize it to the right (i.e., so that it satisfied condition (1)) as described in section ; if we are absorbing into the right environment, then we have to normalize to the left (i.e., using the other condition).

To absorb, we perform the same process as in section , save that we only absorb one site into each of the environments.

Normalization is an $O(d^2\chi^3)$ operation (the complexity required to perform the singular value decomposition) and absorbtion is $O(c^2d^2\chi^2 + cd\chi^3)$.

We call the normalized site $\tilde{S}$ to distinguish it from the unnormalized site S which was obtained from ARPACK. We feed the unnormalized S into ARPACK in the next iteration, and not $\tilde{S}$.

Also, when normalizing the site matrix one needs to be careful not to accidently introduce a random phase; the procedure described in does this, but if one adopts another procedure one must take this into account.

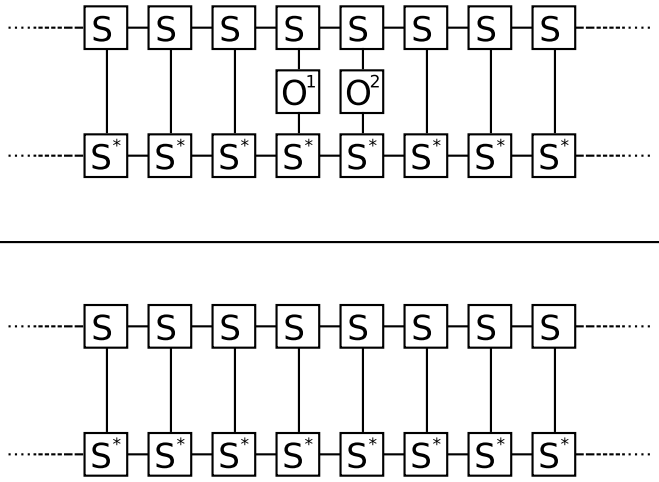
Step 4: Increase χ

At this point, we might want to increase χ. (We could do this, say, once every thousand iterations, or every time some convergence criteria is met.) To do this, we effectively just absorb a random isometric tensor U and its conjugate into our environments to increase their χ dimension.

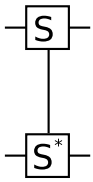
You also need to conjugate the (unnormalized) site matrix S obtained in this iteration about U so that we have something useful to feed into the ARPACK in the next iteration.

Phase 3: Compute Expectations

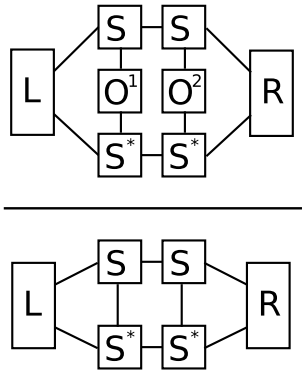
At some point, we will want to take our site matrix and use to compute the expectation of an operator. Assuming that we have converged to a translationally invariant site matrix, which we get from the normalized $\tilde{S}$ in our iteration. The expectation value $\left\langle O^1 O^2 \right\rangle$ where O^1 and O^2 are local is given by the ratio of the following infinite tensor contractions:



Assuming finite boundary effects, the boundary on the matrix product state will not matter in the finite limit. In particular, as a result it should be the case that the repeated action of the matrix



on any (left or right) vector will converge. This gives us the (left and right) effective infinite environment; that is, given left and right eigenvectors of this matrix corresponding to the greatest eigenvalue, we can rewrite our quotient in the form,



Finding the L and R eigenvectors is an $O(d^2\chi^3)$ operation as long as we supply ARPACK with a matrix-vector product routine.