Error Stabilization Notes

August 28, 2020

1 Program Overview

The core of the program is the stabilize function. It takes a norm matrix and a hamiltonian of the same shape as inputs, with a number of optional inputs that allow the parameters of the stabilization to be changed.

The program begins by calculating a list of norm matrices. The process for calculating these norm matrices will be described in more detail later in these notes. The basic loop is a markov-chain Monte Carlo optimization that maximizes the lowest eigenvalue of the norm matrix. Once it is positive definite, the matrix is accepted into the full list.

A similar process is repeated to gather a list of acceptable H matrices for each N matrix. If the autotune flag is set to True, the program will attempt to find an optimal configuration of step size and acceptance width. The fittness parameter for this selection is the sum of mean and standard deviation for the first 1000 matrices optimized using markov-chain Monte Carlo. The optimization is done with a logarithmic grid search. Once an optimal width and step are selected, the width is reduced to the width specified when the function is called. Matrices are only accepted when the width is at its final, cooled value. If convergence ratio is stuck at a local minima, the width can be increased and 'recooled' back down to try to break it out of the local minima.

Matrices are accepted to a list if their convergence ratio is less than the cutoff and the width is at its final value. If the recooling process has fun many times and there are very few accepted H matrices, then it gives up on the current N matrix.

Once matrices are accepted, the integral of the weigth function is calculated, and used to calculate observables.

2 Norm Matrix Selection

The condition used to select norm matrices is its lowest eigenvalue. Its acceptance guide is given by

$$\frac{1}{\exp(-\lambda/\delta_n)+1}\exp\left(\frac{\sum_{i,j}(N-N_{start})}{2\sigma^2}\right)$$

The first sigmoid ensures that matrices with higher minimal eigenvalues are accepted with higher probability. The second gaussian ensures that the random walk does not wander too far from the expected range σ.

At each step, the norm matrix is stepped in a random direction proportional to stepsize and the expected range. This step is always symmeterized to keep the resulting matrix hermitian.

The guide is calculated for each trial norm matrix and accepted with probability given by w_{new}/w_{old},where w_{old} is the previous acceptance guide that was accepted.

The process repeats this walk. If a norm matrix is accepted and has positive eigenvalues, then it is appended to a list. This process is repeated until there are as many norm matrices as specified by the run parameters.

3 H Matrix Selection

3.1 Autotuning

If autotuning is turned on, then the stepsize and width of the acceptance guide are tuned to give faster convergence. The program executes a rough logarithmic gridsearch and for each combination of width and stepsize runs the Markov Chain Monte Carlo for 1000 steps. It computes the mean of these steps and the standard deviation. The best parameters are chosen as those where the sum of mean and standard deviation are minimal. This is a rough condition, but it is simple and has given good results.

Once starting conditions are selected, a cooling schedule is determined so the width of the acceptance guide decreases to a smaller value. If autotuning is turned off, the cooling schedule is just the initial conditions given.

3.2 H Matrix Steps

The steps for H matrix selection are very similar to those used for norm matrix selection. This process is repeated for each Norm Matrix selected in the previous part.

The condition minimized is the maximal convergence ratio for the last three orders, given by

$$c_i = \frac{E_{i-1} - E_i}{E_{i-2} - E_{i-1}}$$

The eigenvalues calculated here, E, are solutions to the generalized eigenvalue problem

$$Hv = NEv$$

The guide is calculated as

$$\frac{1}{\exp((c - c_{target})/\delta_H) + 1} \exp\left(\frac{\sum_{i,j}(H - H_{start})}{2\sigma^2}\right)$$

With the probability of acceptance again given by w_{new}/w_{old}.

When the convergence is less than the target convergence and the matrix is accepted, it is appended to a list.

If the standard deviation of the last many convergences is very small, the program determines that it is stuck. To resolve this, it creates a new cooling schedule to warm and then recool the acceptance. This greatly improves the convergence and can keep the optimization from getting stuck.

If the recooling process happens too many times, the N matrix is deemed unsuitable and it is rejected. The program then moves on to the next norm matrix.

3.3 Weight Function Integral

For each matrix in the list, the weight function integral is calculated. This is accomplished by first constructing a normalized multivariate gaussian distribution with the same mean and standard deviation.

The form of this gaussian is

$$\frac{1}{\sqrt{(2\pi)^L |C|}} \exp(-\frac{1}{2}(x-\mu)C^{-1}(x-\mu))$$

Where C is the covariance matrix for the distribution,

$$C_{ij} = \left\langle (H_i - \bar{H}_i)(H_j - \bar{H}_j) \right\rangle$$

where i and j loop over all degrees of freedom in the matrices.

This distribution is normalized and therefore its integral is know. The gaussian distribution is then resampled with the same Markov Chain Monte Carlo methods used above, and for each matrix sampled we compute its guide function as previously, as well as its value in the gaussian distribution it was sampled from. We can then calculate the weight function integral as

$$\int w dH = \left\langle \frac{w(H)}{G(H)} \right\rangle$$

where G is the gaussian distribution and w is the weight function used when first sampling H matrices.

4 Calculating Observables

For each norm matrix, we computed the weight function integral and the mean H matrix. We form pairs of norm and H matrices in this way, and for each calculate the observable we are interested in, for example the eigenvectors or lowest eigenvalue. We then weight these observables by their weight function integral to produce our final result.