Outline of typical quantum simulation

Start from the script "coherentSimMP.py" (coherent Simulation on Multiple Processors). This script initializes and simulates the following coherent state

$$|\vec{z}\rangle = \bigotimes_{i=1}^{N} \exp\left[-\frac{|z_i|^2}{2}\right] \sum_{n_i} \frac{z_i^{n_i}}{\sqrt{n_i!}} |n_i\rangle \,. \tag{1}$$

where N represent our number of allowed momentum modes. The vector $\vec{z}$ is the given using the vectors IC and ϕ defined in the configuration parameters. $\vec{z}$ is given

$$\vec{z}_j = \exp(i\phi_j)\sqrt{IC_j} \,. \tag{2}$$

The sum in equation 1 is taken such that the expectation of each mode j is $0.999\, IC_j$.
The simulated Hamiltonian is

$$\hat{H} = \sum_{j}^{M} \omega_j \hat{a}_j^\dagger \hat{a}_j + \sum_{ijkl}^{M} \frac{\Lambda_{kl}^{ij}}{2} \hat{a}_k^\dagger \hat{a}_l^\dagger \hat{a}_i \hat{a}_j \,. \tag{3}$$

With the following coupling constants

$$\Lambda_{pl}^{ij} = \left(\frac{C}{2(p_p - p_i)^2} + \frac{C}{2(p_p - p_j)^2} + \Lambda_0 \right) \delta_{pl}^{ij} \,. \tag{4}$$

and kinetic term defined as either

$$\omega_j = j\omega_0 \,, \text{ or} \tag{5}$$

$$\omega_j = \frac{j^2}{2}\omega_0 \,, \tag{6}$$

depending on whether or not the "quad" variable in the configuration parameters is set to True or False.

1. **Pick the initial conditions.** Start from the "main" function. The first step is to find all the terms in the sum in equation 1. For historical reasons this is done rather circuitously by first finding the terms expressed as differences from the expectation values, this is done in "GetDNs". We then loop over the differences and construct a list "terms" which contains all the number eigenstates in the sum in equation 1.

2. **Find the special Hilbert spaces.** Next we need to find and tag all the special Hilbet spaces in the problem. This is done in a for loop in "main". Each special Hilbet spaces is given a "signature" which contains a tuple of total particle number and its net momentum, (n, p), which uniquely identify the special Hilbet space. In this step we construct a dictionary "H_{sp}". The keys for the dictionaries are the signatures and the values are the terms in "terms" that correspond to that special Hilbert space. We also construct a list of all the special Hilbert space signatures "tags" which will be used in the data interpretation step.

3. **Perform the simulation.** We now want to actually run the simulation. We do this done by simulating each special Hilbert space in parallel using a multiprocessing pool in "main". Each special Hilbert space is simulated using "RunTerm". Here we create a simulation object "SimObj", a class defined in "SimObj.py", and use it to store our various simulation configuration parameters. Next we want to initialize a quantum solver "QuantObj" a class in "FullQuantumObjRetry.py".

   (a) **Find the states in the special Hilbert space.** The first step is to find the states in special Hilbert space. This is done in "InspectStatesR" in the "QuantObj". Note that this function assumes that the modes are naturally ordered and linear with index. The number of states in a given special Hilbert space will be denoted N_s.

   (b) **Project the Hamiltonian into the special space.** Next we compute the Hamiltonian and squared Hamiltonian in the special Hilbert space. The Hamiltonian is computed by looping over the allowed states and then computing the action of each term in the sums in equation 3 on that allowed state. What results is a 2-d matrix $H \in \mathbb{R}^{N_s \times N_s}$. We then simply square this Hamiltonian and save this as another 2-d matrix.

   (c) **Project the wavefunction into the special space.** We now need to project our wavefunction $|\vec{z}\rangle$ into the special Hilbert space. This done in the "QuantObj" method "SetPsiHS".

(d) **Run the simulation.** We now want to run the simulation. This is done by calling the "Run" method in our "SimObj" which loops through each solver and call each one's "Update" method.

(e) **Output data.** At the end of each time step in "Run" we output data using each solver's "DataDrop" method.

4. **Interpret data.** We now want to interpret the output data. Generally there are a few interesting things to look at. Generally its best if you run the following steps in order.

   (a) **Calculate mode occupations.** We first want to find how the expectation values of the occupations change over time. This is done by running "di_CS_Num.py". The data directory needs to be specified at the top.

   (b) **Calculate M.** We can also get the principal eigenvector and eigenvalues of the $\hat{M}_{ij} = \langle \hat{a}_i \hat{a}_j \rangle$ operator using "di_calcMSmart.py".