For information of how to send data in vector using MPI, see:

<https://mpitutorial.com/tutorials/point-to-point-communication-application-random-walk/>

We still use dirow, dicol, dmat to store matrix.

Because we broadcast dmat0, dmat1, vmode0, vmode1 to all process, we can just do binary insert now.

We also have to construct a int \* rowptr store information about first position of row-index in our form.

Then we could use row based method to evolve our detector wave function.

How to communicate vector (x,y) between different process:

How do binary search for dicol, and record different dicol to for:

remote\_vector\_index and

still we need

1 remote\_vector\_ index ( index of remote vector we want to get from other process ( this is index in global matrix (not in local matrix))  
2 remote\_vector\_Count: remote vector number for each process they want to get

3 remote\_vector\_ptr: remote vector pointer ( displacement) for each process we want to get.

Then when we do binary search in remote\_vector\_index ( ordered list) to find relative position of index in ghost cells.

Attach ghost cell at end of local x,y vectors.

construct new array local\_dicol to record element in local array. ( if element is not in proces, do binary search in remote\_vector\_index and find the location offset, record the offset in local vector for computation.)

Now we have construct detector matrix, finish code for evolve detector wave function, read and load detector Hamiltonian and wave function.

We can now move to construct full matrix of photon + detector system.

construct diagonal part of full\_matrix.

For diagonal part of full matrix, we have to go through double loop for all detector state. You can split it for each process.

We have finished the code compute diagonal part of full\_system matrix.

Now we parallelize the code insert\_sys\_quotient\_state and detector\_quotient state.

// technical difficulty:

We have to know how to construct ordered list between different process and do binary search there.

1 For sys\_quotient\_state, since each process has their own full\_system matrix element, we can make an ordered list there for each of them.d

For sys\_quotient state ,because when we order the list of sys\_quotient state, we first compare detector 1 mode and then detector 2, since the detector state is already ordered, when we split job for each process according to detector 1 state’s index, the full system state generated have definite rank relationship between different process.

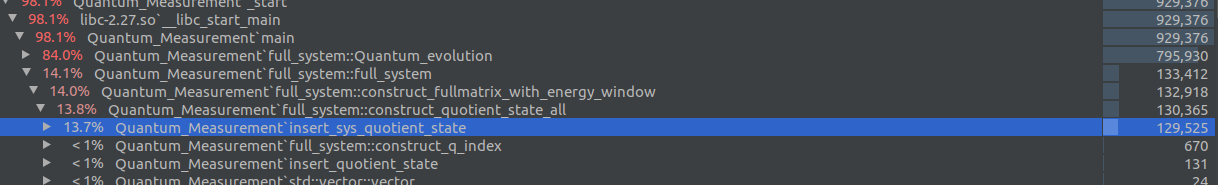
2 For detector quotient state:

for detector 2 quotient state, same applies, since when we order the list according to detector 1 vmode and system quotient state, it is ordered between list.

However, for detector 1 quotient state, we have to order it according to detector 2 vmode. (Worst scenario: trasmit information to detector 0 and let detector 0 do the job.)

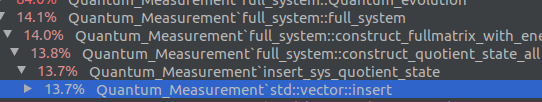
Solution is use Merge sort algorithm. each process sort result in their own process and then merge result in master process.

Master process then scatter result to all slave process.



Code need improvement.

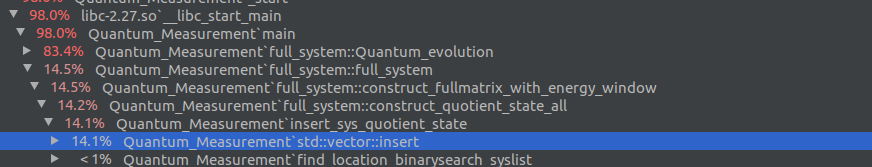
insert\_sys\_quotient\_state is too slow.



Most time spend at vector::insert()

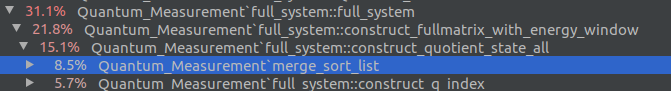
Maybe use binary search tree to search and insert at same time.

resize vector in advance doesn’t help much.



use merge sort algorithm improve speed :

Merge sort:



MPI\_Code Analysis:

We finish MPI version of code in 2020.07.22. Now we begin to debug this code.

Below I will recrod point worth mentioning when I am debugging code and make future reader easier to understand this code.

error message:

-1 : log file fails to open

-2 : input file fails to open

-3: output file fails to open

-4: delt or other time variable is wrong

-5: tlnum is not right.

-6: fail to load detector wave frequency.

-7: Can not find initial state or bright state in vmode.

-8: Wrong number of process as input.

-9: can not open file for Full system’s Hamiltonian

-10: can not open the file for Full system’s state.

-11 : can not open the file for detector Hamiltonian

-12: can not open the file for detector wave function

We are now at line 128 at full system: We finish shift energy of full system and construct initial state and pre\_coupling evolution of detector. We also check the total energy computed is right.

Caution:

The detector off diagonal element have double number than serial code version, because we have to make matrix row-based form to enable SUR algorithm for parallel version.

This may impact the way we construct full-matrix (double counting)

(For example when computing q\_index\_list, if we do not add

if(i>j) continue,

they will count all off diagonal element, so number of element is different from original result. if we add

if(i>j) continue,

result is the same.

So now we are sure there is no bug in construct\_q\_index\_all().

When compute corresponding off-diagonal term in full matrix, we have to record both a\_{ij} and a\_{ji}.

One interesting worth recording how we able to count transpose off-diagonal term in our code:

For sys\_detector coupling and corresponding detector off-diagonal term in full matrix, we just manually double count it.

For detector\_detector\_coupling ,in original code, we have a constraint:

n > i.

Which is index in slist (n) larger than another index in slist (i) , this will avoid double counting .

In MPI code we simply delete this constraint to count transpose off-diagonal term.

In parallel code, result may be slightly different from serial code due to round off error.

Some additional feature for parallel code worthnoting;

1 When we construct diagonal matrix for full system, we partition the part for first detector between different process.

Thus we have to re-order d1list created here.

2 When we compute off-diagonal part of full matrtix, we have to broadcast possible combination of vmode to all process and check individual sys\_quotient\_state or quotient\_state there.

3 We record all off-diagonal part not only upper-triangle one to enable SUR algorithm run in parallel.

4 We have to prepare the ghost zone, information about matrix x,y in other process before we begin SUR algorithm.

5 When compute output, for sys\_energy, we do not have to communicate with other process, because slist is created locally.

When compute detector\_energy, we need to communicate with other process and we create separate data structure to do so.

6 We can use diagonal Hamiltonian itself to construct quotient\_state\_list and sys\_quotient\_state\_list. No need to save and load quotient state in our program.

7 dv\_all is shared for all process.

Worth noting point in MPI Programming:

1 When you use MPI\_Scatter, for count of sending buffer, that should equal to number of element send to each process , not total length of this buffer.

Same apply MPI\_Allgather and MPI\_gather

Attention to MPI\_Alltoall:

the number of element to send is the one each process send to each process.

So pay attention to num\_proc and 1 etc.

2 When you use MPI\_scatterv , even though you don’t use value in displs and vector count in slave process, you have allocate space for them otherwise you will get Segmentation fault.

3 You can not directly pass variable into vector .data(). This will cause vector fail to trace its element number.

You have to use push\_back() to make container know the size of element list.

4 We find the matrix element in full matrix is not distributed evenly between two detector. This problem need to be solved in the future. (or may not)

5 vector . resize() will change the size of vector. Do not confuse it with reserve().

6 don’t pass unintialized vector pointer into function, vector constructed in function will be delete after you leave function.

Also do not pass uninitialized pointer into process, when you allocate space for pointer in function ,it will just change location and pointer just access this memory after you leave that function (because it returns to original one in main function).

Way to solve it: pass pointer of pointer.

Command use for debug MPI code:

1 You have to create separate folder as Debug and Release for different building.

Also, when you have Cmake\_Cache.txt in your folder, you will not be able to build in subfolder.

2 Mpich and openmpi will contradict each other if you install both of them in your machine.

This will render each process fails to know existence of each other , i.e.MPI\_Comm\_World only include themself at run time.

3 To use Cmake:

Cmake file:

set(CMAKE\_C\_FLAGS "${CMAKE\_C\_FLAGS} -O0")  
set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -O0")  
SET(CMAKE\_CXX\_COMPILER mpicxx)  
SET(CMAKE\_C\_COMPILER mpicc)  
target\_link\_libraries(Quantum\_Measurement stdc++fs mpi) # if you use filesystem, you will need to link executable with stdc++fs

4 Command for Cmake for debug:

cmake -DCMAKE\_BUILD\_TYPE=Debug (location of CMakelist.txt)

(https://stackoverflow.com/questions/7724569/debug-vs-release-in-cmake)

then :

mpirun -np 2 xterm -e gdb (my\_mpi\_application)

Now we combine MPI with VanVleck transformation.

In contrast with serialization code, in parallel code, we should pass dv\_all instead of dv to our process.

Also in loop of construct\_state\_coupling\_subroutine, we should do for i = start\_index to end\_index for outer layer. (each process only construct part of detector matrix).

We also have to pass dmat0 / dmat1 as state\_energy because there we have state energy information across all process.

We have to pass dmat[0] / dmat[1] as state\_energy\_local, because when we want to push back off-diagonal part into matrix in local process, we need them.

We treat the anharmonic(diagonal part) and inter-state coupling differently.

When we have anharmonic correction, we make change both to dmat0/dmat1 (state\_energy) and dmat[0]/dmat[1] (state\_energy\_local). (dmat0 and dmat1 will still be used when we want to construct diagonal part of full\_matrix).

When we have inter-state-coupling ,we only push it into dmat[0]/dmat[1] (state\_energy\_local) because that’s where we construct our detector matrix.

Note the function of dmat0 / dmat1 and total\_dmat[0] / total\_dmat[1] is different.

In total\_dmat[0] we have the rank of matrix element in order of process id, so we can not distinguish diagonal part and andiagonal part in total\_dmat. It’s only used when we want to store and load detector Hamiltonian.

The problem is at SUR algorithm. We should update x,y after we finish update x, otherwise we are not using SUR algorithm and there is blow up problem.

Find extra bug when we construct receive buffer for detector.

Be very careful about last process.

col\_index\_copy[i] / vsize

may be larger than last process’s index and give wrong answer.

something wrong with p which is index for dr.

Still have bug if we run too many process.

Problem is at compute important state index():

for(index =0 ; index < 1; index ++)

We set index <1 because now bright state is not included when we only simulate initial state.

if(my\_id == bright\_state\_pc\_id[m] and bright\_state\_index[m] < dmatsize[m])

Also this is to fit the condition we do not set bright state index (default is 0), that case we have to avoid Seggev problem.

bug in Compute\_sys\_reduced\_density\_matrix.cpp

if sys\_quotient\_state size is 0, we may get error. Now fixed.

The program now pass the final test and is able to use. Bug free version. Cheers.