

1 Task-1:

Chosen code: Density Matrix Evolution of a 12-qubit system (noiseless)

1.1 :

System description:

I have modeled an n-qubit system, where we take $n = 12$, initialized at thermal equilibrium state represented by a density matrix. The system's time dynamics are generated by a TFIM (Transverse Field Ising Model) model Hamiltonian, and for solving the ODE I used RK_4 method.

So, first we are calculating the Hamiltonian for a set of parameters ($J = 1.0, h = 0.5$). Then we are obtaining eigenvalues of Hamiltonian to obtain energy, from this we get probability density. Thus, we calculate the density matrix for n-qubit system. Finally, we use the Liouville-von Neumann Equation ODE to solve for a period of time. Finally, from here we can obtain various quantities for analysis like purity, magnetization dynamics and so on.

Computation/Mathematical fomulation:

The basic mathematical equations used are as follows.

$$H = -J \sum_{i=1}^n Z_i Z_{i+1} - h \sum_{i=1}^n X_i$$

$$\rho = \frac{e^{-\frac{E}{T}}}{Z}$$

$$\frac{\partial \rho}{\partial t} = [H, \rho]$$

Pseudocode:

```
#calculationg of Hamiltonian
for i in 1 to n:
    H=summation of zi*z(i+1), xi for each i, with respective coefficients
E=eigval(H)
#density matrix
rho=e^(-E/T)/sum(rho)
#time evolution
function: rk4 solver
function time evolution: store values of ode
```

Sizes:

Size of each tensor (approx.):

$$\rho = (2^n, 2^n)$$

$$H = (2^n, 2^n)$$

RK_4 tensors (temporary): $\mathbb{C}^{2^n \times 2^n}$

Loop bounds:

Hamiltonian construction loops: $i \rightarrow 1, n$ (very small number)

ODE solver loop: a loop for number of steps (usually between 100 to 1000)

Opportunity for parallelism: Matrix multiplications of exponential sizes $H\rho, \rho H, \rho \times \rho$

1.2 :

Memory bound or compute bound? Even though matrix multiplication is highly compute-bound, the storage of these matrices are also highly memory bound. Not sure of any **possible reuse**. **Sequential steps:**

In ODE, k_1, k_2, k_3, k_4 calculation, since each step depends on previous one

1.3 :

How would CUDA threads map into the iteration process:

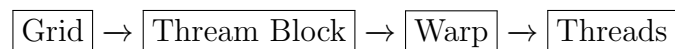
Each thread computes one output element of the matrix multiplication

Will it scale well to thousands of threads?

Yes, since the sizes of matrix go up exponentially with n , there are millions of required computations

Challenges: for further increased n , there is a memory saturation

2 Task-2:



- Thread \rightarrow computes each multiplication to give output in matrix
- Warp (32 threads) \rightarrow contiguous elements in a row/column tile
- Thread block \rightarrow computes a square tile
- Grid \rightarrow covers the full $2^n \times 2^n$ matrix

Synchronization might be required within a block after loading tiles into shared memory.

Memory bottlenecks:

There will be bottlenecks due to repeated full matrix reads/writes since, because matrices are too large to fit in cache, there will be frequent global memory accesses.

I don't think there is any possibility of shared memory usage (but I'm not sure).