UCSB, Physics 129AL, Computational Physics: Section Worksheet, Week 4

Zihang Wang (UCSB), zihangwang@ucsb.edu

January 31, 2025

Section Participation and Submission Guidelines

Section attendance is required, but you do not need to complete all the work during the section. At each section, the TA will answer any questions that you might have, and you are encouraged to work with others and look for online resources during the section and outside of sections. Unless otherwise stated, the work will be due one week from the time of assignment. The TA will give you 1 point for each task completed. You can see your grades on Canvas.

We will use GitHub for section worksheet submissions. By the due date, you should have a single public repository on GitHub containing all the work you have done for the section work. Finally, upload a screenshot or a .txt file to Canvas with your GitHub username and repository name so the TA knows who you are and which repository you are using for the section.

Remember: talk to your fellow students, work together, and use GPTs. You will find it much easier than working alone. Good luck! All work should be done in the Docker container, and don't forget to commit it to Git!

Task 1: Master Theorem, Time Complexity, and Strassen's Algorithm

As we discussed in the lecture, **master theorem** is useful in estimating the asymptotic time complexity of analysis of **divide-and-conquer** algorithms that has the following recursive form,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n). \tag{1}$$

You are ask to design the Strassen's Algorithm that efficiently compute matrix multiplications, with the following information.

Question 1: Naive Divide and Conquer Approach

The naive divide-and-conquer algorithm divides two $n \times n$ matrices A and B into four smaller submatrices of size $n/2 \times n/2$:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

The product matrix $C = AB$ is similarly divided into four submatrices:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

At each step, the naive matrix multiplication requires 8 multiplications of submatrices. The recursion formula is given by the following expression,

$$T(n) = 8T(n/2) + \mathcal{O}(n^2).$$

- Use Python to construct the above native divide-and-conquer algorithm. You can use either recursive (top-down) or iterative (bottom-up) method in dynamic programming.
- Use the master theorem to determine the time complexity, and find the **critical exponent**.
- Compare and visualize the time complexity obtained from your Python program and the time complexity calculated from the master theorem. Do the asymptotic behaviors agree? Hint: you should estimate the critical exponent for your program.

Question 2: Strassen's algorithm

Strassen's algorithm reduces the number of matrices to 7 by introducing intermediate matrices:

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}), \quad M_2 = (A_{21} + A_{22})B_{11},$$

$$M_3 = A_{11}(B_{12} - B_{22}), \quad M_4 = A_{22}(B_{21} - B_{11}),$$

$$M_5 = (A_{11} + A_{12})B_{22}, \quad M_6 = (A_{21} - A_{11})(B_{11} + B_{12}),$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22}).$$

Using these, the submatrices of C are computed as:

$$C_{11} = M_1 + M_4 - M_5 + M_7, \quad C_{12} = M_3 + M_5,$$

$$C_{21} = M_2 + M_4, \quad C_{22} = M_1 - M_2 + M_3 + M_6.$$

- Use Python to construct the above Strassen's algorithm. You can use either recursive (top-down) or iterative (bottom-up) method in dynamic programming.

- Find a, b, f of the recursive expression below,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n). \tag{2}$$

- With the above recursion expression, use the master theorem to determine the time complexity, and find the **critical exponent**.

- Compare and visualize the time complexity obtained from your Python program and the time complexity calculated from the master theorem. Do the asymptotic behaviors agree? Hint: you should estimate the critical exponent for your program.

Task 2: Heisenberg XXX Hamiltonian on a Ring

The Heisenberg XXX Hamiltonian is very important of quantum spin systems. It is used to describe interacting spins arranged in a circular or periodic arrangement, where the interactions between neighboring spins are isotropic, meaning they are independent of direction. In the context of a ring, the system consists of a series of spins arranged in a closed loop, such that the first spin interacts with the last one, completing the periodic boundary conditions.

Let us introduce the raising and lowering operators $S_\pm = S_x \pm iS_y$, such that

$$S_+|\uparrow\rangle = 0, \quad S_-|\uparrow\rangle = |\downarrow\rangle, \quad S_z|\uparrow\rangle = \frac{1}{2}|\uparrow\rangle$$

$$S_+|\downarrow\rangle = |\uparrow\rangle, \quad S_-|\downarrow\rangle = 0, \quad S_z|\downarrow\rangle = -\frac{1}{2}|\downarrow\rangle. \tag{4}$$

We can write the XXX Hamiltonian as the following,

$$H = \frac{JN}{4} - J\sum_{i=1}^{N} \frac{1}{2}\left(S_{+i}S_{-i+1} + S_{-i}S_{+i+1}\right) + S_{zi}S_{zi+1}, \tag{5}$$

with periodic boundary condition, such that $N + 1 = 1$.

Let us look at the different terms. The terms involving $S_\pm$ are called hopping terms since they move a spin up or spin down to a neighboring site. The constant term proportional to N is added for convenience. It is simply an overall shift of the energy levels. Depending on the sign, you want to align or anti-align the spins, i.e. (anti-)ferromagnetism.

The above Hamiltonian can be written in a matrix form with a given Hilbert space. let's consider a simple 2-spin chain: The basis states are, $|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\downarrow\downarrow\rangle$, where the index location represents the sites, i.e. $1, 2$. The Hamiltonian for the Heisenberg XXX model with $N = 2$ spins is given by,

$$H = \frac{3J}{4} - J\left[\frac{1}{2}\left(S_{+1}S_{-2} + S_{-1}S_{+2}\right) + S_{z1}S_{z2} + \frac{1}{2}\left(S_{+2}S_{-3} + S_{-2}S_{+3}\right) + S_{z2}S_{z3} + \frac{1}{2}\left(S_{+3}S_{-1} + S_{-3}S_{+1}\right) + S_{z3}S_{z1}\right]$$

with states,

$$|\uparrow\uparrow\uparrow\rangle, |\uparrow\uparrow\downarrow\rangle, |\uparrow\downarrow\uparrow\rangle, |\uparrow\downarrow\downarrow\rangle, |\downarrow\uparrow\uparrow\rangle, |\downarrow\uparrow\downarrow\rangle, |\downarrow\downarrow\uparrow\rangle, |\downarrow\downarrow\downarrow\rangle.$$

For example, matrix element can be calculate via the following expression,

$$\langle\downarrow\uparrow\downarrow| \, S_{+2}S_{-3}|\downarrow\downarrow\uparrow\rangle = 1. \tag{3}$$

Question 1:

Using a similar idea provided above, write a Python program that constructs the Hamiltonian matrix with arbitrary number of chain elements. **Make sure to include the periodic boundary condition!** Explain the time complexity of your program, and compare with the numerical simulation for various N.

Here are things that you should keep in mind: 1) the Hilbert space grows as 2^N. 2) There are symmetries associated with the Hamiltonian. 3) Sparsity.

Question 2:

Recall the **QR algorithm** we discussed in the lecture. It is an iterative method for diagonalizing a matrix, which works by repeatedly performing QR decompositions:

1. Start with the matrix $H_0 = H$.
2. Perform a QR decomposition of H_0, i.e., write

   $$H_0 = Q_0 R_0,$$

   where Q_0 is orthogonal and R_0 is upper triangular.
3. Construct the new matrix

   $$H_1 = R_0 Q_0.$$

4. Repeat the process until converging to a diagonal matrix,

   $$H_{n+1} = R_n Q_n.$$

Write a Python program that produces the diagonal matrix for a given input Hamiltonian. You can use either recursive (top-down) or iterative (bottom-up) method in dynamic programming.

Question 3: Green's function

Write two python programs that preform **LU decomposition and Cholesky decomposition** of the following expression,

$$(\omega I - H)G = I, \tag{4}$$

where I is the identity operator, the input will be the frequency ω and H, and output the G. G is called the **Green's function**, and remain essential in many-body perturbation theory. Since in this case, the Hamiltonian is Hermitian, the greens function is real-valued. For $N = 30$, plot the Green's function with respect to different frequency, and explain its behavior. Hint: singularities are called resonance.

Question 4: Magnon (Spin wave)

Ground state. Since the total spin is conserved, the state in which all spins are aligned in the same direction must be an eigenstate of the Hamiltonian. This corresponds to the ferromagnetic vacuum. Therefore, we define the vacuum state as

$$|0\rangle = |\uparrow\uparrow \dots \uparrow\rangle.$$

It is straightforward to calculate the energy of this state:

$$H|0\rangle = 0.$$

The **magnon (spin wave) state** is defined as:

$$|p\rangle = \sum_n e^{ipn} S_{-n}|0\rangle$$

where i is the imaginary unit, S_{-n} is the spin lowering operator at site n, and $|0\rangle$ is the ground state. The periodic boundary condition ensures that, $e^{ipN} = 1$. This ensures that the momentum are quantized,

$$e^{ipn} = e^{ip(n+N)}$$

which implies that the momentum p is quantized in units of $\frac{2\pi}{N}$. Thus, the allowed values for p are:

$$p = \frac{2\pi k}{N}, \quad k = 0, 1, 2, \dots, N-1$$

The energy corresponding to each allowed value of p is given by the dispersion relation:

$$E(p) = 2J \sin^2\left(\frac{p}{2}\right)$$

Substituting the quantized values of p, we get the energy levels for the magnon as:

$$E_k = 2J \sin^2\left(\frac{\pi k}{N}\right), \quad k = 0, 1, 2, \ldots, N-1$$

Write a python program (any method you like) to calculate the energy directly by solving the following linear equation,

$$H\,|p\rangle = E\,|p\rangle\,,$$

The input will be p and H, and the output will be the energy. Compare this energy with the above expression. For the above spin chain with $N = 30$, plot the energies associated with different p, and compare it with the above energy expression. If you have problems dealing with complex numbers, you can make a tuple that offsets the imaginary part to a new vector space, i.e. $x+iy \to (x, y)$.