PHYS 357 Pset 4. Due 11:59 PM Thursday Oct. 17

In some of these problems, we'll see that interesting behavior can emerge when we work with states with larger angular momentum. It's a pain to do this by hand, so feel free to carry out calculations on a computer. If you do this, do please explain how your codes work so the TAs can understand what you did when it comes time to mark.

1. Townsend 3.24. Please do this one by hand - Townsend helpfully supplies all the relevant matrices for you.

2. Consider a ping pong ball (mass 2.7 grams and radius 20 mm) rotating at 10 radians per second (in real play, the spins can be 100 times higher). What is the approximate angular momentum (order of mangitude is fine, so you can use MR^2 for the moment of inertia)? If we say the ping pong ball is in a state $|j, j\rangle$ rotating about the $z-$axis, what is the approximate value of j? Given that value of j, what is the total angular momentum in the $x-y$ plane, $\sqrt{J_x^2 + J_y^2}$? We can express this as an uncertainty in the rotation axis of the ping pong ball, with $\sigma(\tan(\theta)) \sim \frac{\sqrt{J_x^2+J_y^2}}{J_z}$. What is your approximate value for $\sigma(\tan(\theta))$? For a macroscopic object, talking about "the" axis of rotation is a very good approximation!

3. **Part A)** Consider a spin-1 Stern-Gerlach experiment where you send a beam of particles from the oven down a modified SGz machine, where you block the $J_z = 0$ beam before recombining. You might think that would leave your particles in the state $J_z = (1, 0, 1)/\sqrt{2}$. If that were the case, what would you see (in terms of what fraction of particles go up, go down, or aren't deflected at all) if you then send the beam through an SGx machine? Through an SGy machine?

   Part B) Explain why these results can't be correct (hint - what happens if I take my SGz into SGx apparatus, and rotate the whole setup by 90 degrees about the $z-$axis).

   Part C) If you were to actually set this experiment up and run it, what you would see is that 50% of the particles show up with $J_{x,y} = 0$, 25% show up with $J_{x,y} = +\hbar$, and 25% show up with $J_{x,y} = -\hbar$ (where $J_{x,y}$ means I measure *either* J_x or J_y), so indeed the

apparatus behaves the same if I rotate my coordinate system by 90 degrees, as it must. Can you quantitatively explain the actual observed results? You may use a computer if you like.

4. **Part A)** We saw that J_z commutes with $J_x^2 + J_y^2$. Does J_x commute with $J_x^2 + J_y^2$? If not, what is the commutator $[J_x, J_x^2 + J_y^2]$? Feel free to express as an anti-commutator where $\{\mathrm{A}, \mathrm{B}\} = \mathrm{AB} + \mathrm{BA}$.

   Part B) Let's say I take a set of spin-2 particles, all of which have $J_z = 2\hbar$, which we write as $|2, 2\rangle_z$ (note the z subscript, which says this state is an eigenstate of J_z). If I were to measure $J_x^2 + J_y^2$, what would I observe, and what would the uncertainty in that measurement be?

   Part C) I now send my beam of $|2, 2\rangle_z$ particles down an SGx machine. Show that the expectation of J_x^2 that you see is one half of $< J_x^2 + J_y^2 >$ you calculated in part B).

   Part D) What is the probability that you measure $J_x = +2\hbar$? If this is non-zero, can you explain the apparent contradiction between finding a particle witih $J_x = 2\hbar$ (and hence $J_x^2 = 4\hbar^2$) and the maximum value you found for $J_x^2 + J_y^2$ from part B?

5. **Part A)** Sticking with spin-2 particles, express the pure state $|2, 2\rangle_y$ in both the z-basis and the y-basis. In the $z-$basis, please pick the phase that makes the amplitude of $J_z = +2\hbar$ purely real and positive.

   Part B) For this same $|2, 2\rangle_y$ pure state, what are the uncertainty in J_x and J_z? Show that the uncertainty relation is satisfied.

6. **Part A)** Calculate the matrix that rotates a state about the $y-$axis by $+90$ degrees for a spin-2 particle. Print the absolute value of the matrix.

   Part B) Show that the above matrix rotates the state $|2, 2\rangle_z$ to the pure state $|2, 2\rangle_x$, and $|2, 2\rangle_x$ to $|2, -2\rangle_z$.

7. Bonus: Our end goal is to derive the angular momentum commutation relation $[J_a, J_b] = ih\epsilon_{abc}J_c$ but to do that we'll first need the canonical commutation relation $[x, p] = i\hbar$. In the first bonus we'll work that out, in the second bonus, we'll use it to derive the angular

momentum commutation relations. You are more than welcome to do the second bonus only, and just use the canonical commutation relation as supplied.

In 1924, Louis De Broglie hypothesized (in his PhD thesis!) that all matter behaved like waves, with wavelength set by their momentum. This was the key insight that opened the door to modern quantum mechanics, and within a couple of years both wave and matrix mechanics were basically fleshed out in their modern forms. The De Broglie relation is

$$p = \hbar k \tag{1}$$

where $k = 2\pi/\lambda$ and p is the momentum. If I have a wave function of a particle with wave vector k, I can write that down as a function of x as follows: $\Psi = \exp(ikx)$.

Part A:Show that the operator $-i\hbar\frac{\partial}{\partial x}$ operating on Ψ returns $p\Psi$. In other words, the plane wave is an eigenstate of momentum, and the operator $\hat{p} = -i\hbar\frac{\partial}{\partial x}$ returns the wave function times the momentum.

Part B: Now the position operator $\hat{x}$ in position space (where $\Psi = \Psi(x)$) is, not surprisingly, just x. You can now derive the canonical commutation relation $[x, p]$ by applying the operators to a wave function $(\hat{x}\hat{p} - \hat{p}\hat{x})\Psi$. Show that when you do this, you get $i\hbar\Psi$. That is, when you apply $[x, p]$ to a wave function, you get $i\hbar$ times the wave function, so $[x, p] = i\hbar$.

8. Bonus 2. Now we'll work out the angular momentum commutators, using the fact that $J = r \times p$. If you expand that out, you can see that $J_x = r_y p_z - r_z p_y$, which will all get applied to a wave function.

   Position and momentum on the same axis do not commute, but they do commute if you measure position along one axis, and momentum along another, perpendicular axis. More formally, $[x_i, p_j] = i\hbar\delta_{ij}$. Use this commutation relation and the expressions for Jx, Jy, Jz you get from $J = r \times p$ to show that $[Jx, Jy] = i\hbar Jz$. The commutators for other pairs of axes follow from cyclic permutations of the cross product, so you only need to do it once.

   Note - technically we've only shown this for orbital angular momentum and not spin. We did show that spin-1/2 particles obey the same commutation relation, so I hope it's not a surprise that the relation holds for higher spins as well.

Pset 5

Q1

3.24. A spin-$\frac{3}{2}$ particle is in the state

$$|\psi\rangle \xrightarrow[S_z \text{ basis}]{} N \begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}$$

(a) Determine a value for N so that $|\psi\rangle$ is appropriately normalized.

(b) What is $\langle S_x \rangle$ for this state? *Suggestion:* The matrix representation of $\hat{S}_x$ is given in Example 3.4.

(c) What is the probability that a measurement of S_x will yield the value $\hbar/2$ for this state? *Suggestion:* See Problem 3.23.

a) $|\psi\rangle = N\begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}$ and $\langle\psi| = N^*(-i\ 2\ 3\ -4i)$

we require $\langle\psi|\psi\rangle = 1 = N^*N(-i\ 2\ 3\ -4i)\begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}$

$$1 = N^*N(-i^2 + 4 + 9 - 16i^2)$$
$$1 = N^*N(30)$$
$$\rightarrow |N|^2 = \frac{1}{30}$$
$$|N| = \frac{1}{\sqrt{30}}$$

choose $N = \frac{1}{\sqrt{30}}$

b) from Ex 3.4, $\hat{S}_x = \frac{\hbar}{2}\begin{pmatrix} 0 & \sqrt{3} & 0 & 0 \\ \sqrt{3} & 0 & 2 & 0 \\ 0 & 2 & 0 & \sqrt{3} \\ 0 & 0 & \sqrt{3} & 0 \end{pmatrix}$ and we have $|\psi\rangle = \frac{1}{\sqrt{30}}\begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}$

$$\langle S_x \rangle = \langle\psi|\hat{S}_x|\psi\rangle = \frac{1}{30}(-i\ 2\ 3\ -4i)\frac{\hbar}{2}\begin{pmatrix} 0 & \sqrt{3} & 0 & 0 \\ \sqrt{3} & 0 & 2 & 0 \\ 0 & 2 & 0 & \sqrt{3} \\ 0 & 0 & \sqrt{3} & 0 \end{pmatrix}\begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}$$
$$= \frac{\hbar}{60}(-i\ 2\ 3\ -4i)\begin{pmatrix} 2\sqrt{3} \\ i\sqrt{3} + 6 \\ 4 + 4i\sqrt{3} \\ 3\sqrt{3} \end{pmatrix}$$
$$= \frac{\hbar}{60}(-2\sqrt{3}i + 2i\sqrt{3} + 12 + 12 + 12i\sqrt{3} - 12i\sqrt{3})$$
$$= \frac{\hbar}{60} \cdot 24$$

$\langle S_x \rangle = \frac{2\hbar}{5}$

c) $P(\hbar/2) = |\langle\psi|\Psi\rangle|^2$

↳ state associated with $S_x = \hbar/2 \rightarrow j = \frac{1}{2}$

using the eigenstates from problem 3.23, $|\Psi\rangle = |\frac{3}{2}, \frac{1}{2}\rangle_x = \frac{1}{2\sqrt{2}}\begin{pmatrix} \sqrt{3} \\ 1 \\ -1 \\ -\sqrt{3} \end{pmatrix}$

$$= \left|\frac{1}{2\sqrt{2}}(\sqrt{3}\ 1\ -1\ -\sqrt{3})\frac{1}{\sqrt{30}}\begin{pmatrix} i \\ 2 \\ 3 \\ 4i \end{pmatrix}\right|^2$$
$$= \frac{1}{4 \cdot 2} \cdot \frac{1}{30}|i\sqrt{3} + 2 - 3 - 4i\sqrt{3}|^2$$
$$= \frac{1}{240} \cdot |-1 - i3\sqrt{3}|^2$$
$$= \frac{1}{240}(1 + 9 \cdot 3)$$

$P(\hbar/2) = \frac{7}{60}$

2. Consider a ping pong ball (mass 2.7 grams and radius 20 mm) rotating at 10 radians per second (in real play, the spins can be 100 times higher). What is the approximate angular momentum (order of mangitude is fine, so you can use MR^2 for the moment of inertia)? If we say the ping pong ball is in a state $|j, j\rangle$ rotating about the $z-$axis, what is the approximate value of j? Given that value of j, what is the total angular momentum in the $x-y$ plane, $\sqrt{J_x^2+J_y^2}$? We can express this as an uncertainty in the rotation axis of the ping pong ball, with $\sigma(\tan(\theta)) \sim \frac{\sqrt{J_x^2+J_y^2}}{J_z}$. What is your approximate value for $\sigma(\tan(\theta))$? For a macroscopic object, talking about "the" axis of rotation is a very good approximation!

① $L \sim 10^4 \ mm^2 g \frac{rad}{s}$

```python
import numpy as np
import astropy.units as u
import scipy.constants as const

hbar = const.hbar

# Ping pong ball info
m = 2.7 * u.g
r = 20 * u.mm
w = 10 *u.rad/u.s

# Angular momentum
I = m*r**2
L = I*w
print(f"L = {L}")
L = L.value
```

```
L = 10800.0 mm2 g rad / s
```

② $|j, j\rangle$, $j = \max m$ value

we know $L^2 = \hbar^2 l = \hbar^2 j(j+1)$

$0 = \hbar^2 j^2 + \hbar^2 j - L^2$ (j large so ignore $\hbar^2 j$ term)

$\rightarrow j \approx \frac{L}{\hbar} = 1.024 \times 10^{38}$

```python
# Solve for j
j = np.roots([hbar**2, hbar**2, -L**2])
j = j[j>0][0]
print(f"j = {j}")
```

```
j = 1.0241123287464671e+38
```

③ $J_z^2 = m^2\hbar^2 = j^2\hbar^2$ and $J^2 = \hbar^2 j(j+1)$

$$\begin{aligned} \text{So } \sqrt{J_x^2 + J_y^2} &= (J^2 - J_z^2)^{1/2} \\ &= (\hbar^2 j(j+1) - \hbar^2 j^2)^{1/2} \\ &= (\hbar^2 j^2 + \hbar^2 j - \hbar^2 j^2)^{1/2} \\ &= \hbar\sqrt{j} \\ &= \hbar\sqrt{\frac{L}{\hbar}} \end{aligned}$$

④ $\sigma(\tan\theta) \approx \frac{\sqrt{J_x^2+J_y^2}}{J_z} = \frac{\hbar\cdot\sqrt{L/\hbar}}{j\hbar}$

$= \sqrt{\frac{L}{\hbar}} \cdot \frac{\hbar}{L}$

$\sigma(\tan\theta) = \left(\frac{\hbar}{L}\right)^{1/2}$

```python
# Momentum
Jz = hbar * j
J_xy_plane = hbar* np.sqrt(L/hbar)
print(f"J_xy_plane = {J_xy_plane}")

# Uncertainty
sigma = J_xy_plane/Jz
print(f"sigma = {sigma}")
```

```
J_xy_plane = 1.0672101775460393e-15
sigma = 9.881575718018882e-20
```

Q3 General code (functions)

```python
import numpy as np
import scipy.constants as const
hbar = const.hbar

np.set_printoptions(precision=4) # settings to make arrays easier to read
np.set_printoptions(suppress=True)

# Functions
def compute_prob(SGn, state, nbra, en):
    """
    Get the probability of each eigenstate of Jn for a given state in the Jz basis
    nbra and state expressed in the Jz basis
    """
    # Change basis of state to Jn (Jx or Jy)
    # equiv.: how much of each eigenstate of Jn is in the initial state?
    state_in_n = nbra@state
    print(f"\nState in J{SGn[-1]} basis: {np.round(state_in_n,2)}")
    # Calculate probabilities for each basis state of Jn
    prob_n = np.abs(state_in_n)**2
    print(f"Probabilities in J{SGn[-1]} basis:")
    for i,e in enumerate(en):
        print(f"P({e:.0f}) = {prob_n[i].real:.2f}")

def R_matrix_z(theta):
    """
    Rotation matrix around z-axis for an angle theta
    """
    r1 = [np.exp(1j*theta), 0, 0]
    r2 = [0, 1, 0]
    r3 = [0,0,np.exp(-1j*theta)]
    return np.array([r1,r2,r3])
```

```python
def R_from_J(J,th):
    """
    rotation about an axis is exp(-i th J/hbar)
    """
    e,v=np.linalg.eigh(J)
    e_new=-1j*th*e
    return v@np.diag(np.exp(e_new))@v.conj().T


# Operators for spin-1
Jz = np.diag([1,0,-1])
Jx = np.array([[0,1,0],[1,0,1],[0,1,0]])/np.sqrt(2)
Jy = np.array([[0,-1j,0],[1j,0,-1j],[0,1j,0]])/np.sqrt(2)

# Jx eigenstates
ex,vx = np.linalg.eigh(Jx)
ex,vx = np.flipud(ex), np.fliplr(vx)
xbra=np.conj(vx).T  # z to x

# Jy eigenstates
ey,vy = np.linalg.eigh(Jy)
ey,vy = np.flipud(ey), np.fliplr(vy)
ybra=np.conj(vy).T  # z to y
```

Part A) Consider a spin-1 Stern-Gerlach experiment where you send a beam of particles from the oven down a modified SGz machine, where you block the $J_z = 0$ beam before recombining. You might think that would leave your particles in the state $J_z = (1, 0, 1)/\sqrt{2}$. If that were the case, what would you see (in terms of what fraction of particles go up, go down, or aren't deflected at all) if you then send the beam through an SGx machine? Through an SGy machine?

If after SG_z we have $|\psi\rangle_z = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\end{pmatrix}$ (in J_z basis), we can get each probability for the J_x components ($\hbar, 0, -\hbar$ along J_x) by doing $|{}_x\langle 1,1|\psi\rangle|^2$, $|{}_x\langle 1,0|\psi\rangle|^2$, $|{}_x\langle 1,-1|\psi\rangle|^2$

$\rightarrow |\psi\rangle_z = \frac{1}{\sqrt{2}}|1,1\rangle_z + \frac{1}{\sqrt{2}}|1,-1\rangle_z$ and $|1,m\rangle_x$ = eigenstates of J_x (expressed in J_z basis)

$\Rightarrow$ prob$(m\hbar) = |{}_x\langle 1,m|\psi\rangle|^2 \rightarrow$ $|x\text{bra@state}|^2$

(both express in J_z basis)

(can also see as changing state's basis to J_x & then just getting the coeffs of each J_x basis state)

```
58  # Part A ------------------------------------
59  print("Part A")
60  # State (kets) in Jz basis
61  state_in_z = np.array([1,0,1])/np.sqrt(2)
62  print(f'State after SGz: {np.round(state_in_z,2)}')
63
64  # Go through SGx
65  compute_prob('SGx', state_in_z, xbra, ex)
66  # Go through SGy
67  compute_prob('SGy', state_in_z, ybra, ey)
68
```

```
Part A
State after SGz: [0.71 0.   0.71]

State in Jx basis: [0.71 0.   0.71]
Probabilities in Jx basis:
P(1) = 0.50
P(-0) = 0.00
P(-1) = 0.50

State in Jy basis: [-0.+0.j  1.+0.j  0.+0.j]
Probabilities in Jy basis:
P(1) = 0.00
P(0) = 1.00
P(-1) = 0.00
```

(after going through SG_x)

$\hbar$

after going through SG_y

Part B) Explain why these results can't be correct (hint - what happens if I take my SGz into SGx apparatus, and rotate the whole setup by 90 degrees about the $z-$axis).

The choice of what 'orientation' is x or y is arbitrary so we should be getting the same probabilities for SG_x & SG_y, and unchanged probabilities when we rotate the setup by 90°.
Since we don't get the same probabilities if $|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\end{pmatrix}$ after SG_z, then $|\psi\rangle$ must be wrong

Part C) If you were to actually set this experiment up and run it, what you would see is that 50% of the particles show up with $J_{x,y} = 0$, 25% show up with $J_{x,y} = +\hbar$, and 25% show up with $J_{x,y} = -\hbar$ (where $J_{x,y}$ means I measure *either* J_x or J_y), so indeed the apparatus behaves the same if I rotate my coordinate system by 90 degrees, as it must. Can you quantitatively explain the actual observed results? You may use a computer if you like.

```
88  # Part C ------------------------------------
89  print("\nPart C")
90  # Correct state
91  correct_state_in_z = np.array([1,0,1j])/np.sqrt(2)
92  print(f'Correct state after SGz: {np.round(correct_state_in_z,2)}')
93  # Compute the probabilities
94  compute_prob('SGx', correct_state_in_z, xbra, ex)
95  compute_prob('SGy', correct_state_in_z, ybra, ey)
96
```

```
Part C
Correct state after SGz: [0.71+0.j   0.  +0.j   0.  +0.71j]

State in Jx basis: [ 0.35+0.35j -0.5 +0.5j   0.35+0.35j]
Probabilities in Jx basis:
P(1) = 0.25
P(-0) = 0.50
P(-1) = 0.25

State in Jy basis: [0.35-0.35j 0.5 +0.5j  0.35-0.35j]
Probabilities in Jy basis:
P(1) = 0.25
P(0) = 0.50
P(-1) = 0.25
```

we get these probabilities due to how the state $|\psi\rangle_z$ gets projected on each $J_{x,y}$ eigenstate

this computes prob = $|\langle \text{bra} | \text{state} \rangle|^2$ for every bra ie every eigenstate / eigenvalue

4. **Part A)** We saw that J_z commutes with $J_x^2 + J_y^2$. Does J_x commute with $J_x^2 + J_y^2$? If not, what is the commutator $[J_x, J_x^2 + J_y^2]$? Feel free to express as an anti-commutator where $\{A, B\} = AB + BA$.

$$[J_x, J_x^2 + J_y^2] = \underbrace{[J_x, J_x^2]}_{0} + [J_x, J_y^2]$$

$$= J_y[J_x, J_y] + [J_x, J_y]J_y$$
$$= J_y \cdot i\hbar J_z + (i\hbar J_z)J_y$$
$$= i\hbar(J_y J_z + J_z J_y)$$
$$= i\hbar\{J_y, J_z\}$$

Part B) Let's say I take a set of spin-2 particles, all of which have $J_z = 2\hbar$, which we write as $|2, 2\rangle_z$ (note the z subscript, which says this state is an eigenstate of J_z). If I were to measure $J_x^2 + J_y^2$, what would I observe, and what would the uncertainty in that measurement be?

If $|\psi\rangle = |2,2\rangle_z$, we would measure $\langle\psi|J_x^2 + J_y^2|\psi\rangle = \langle\psi|J^2 - J_z^2|\psi\rangle$

$J^2 \to \hbar^2 j(j+1) = \hbar^2 \cdot 2 \cdot 3 = 6\hbar^2$

$$= 6\hbar^2 - (2\hbar)^2$$
$$= 2\hbar^2$$

Since J_z commutes with $J_x^2 + J_y^2$, they share eigenstates & eigenvalues
→ we get to ask about J_z and $J_x^2 + J_y^2$ at the same time so $\sigma(J_x^2 + J_y^2) = 0$ ($\sigma(J_z) = 0$ since all in $|2,2\rangle_z$)

```python
# Setting up the problem ------------------------------------
j = 2
m = np.linspace(j,-j,int(2*j)+1)
Jz = np.diag(m)

# State |2,2>z
state_ket = np.array([1,0,0,0,0])
state_bra = np.conj(state_ket.T)

# Building the raising & lowering operators
# (for Jz, in the Jz basis)
# Use these to get Jx and Jy for spin-2
n = len(m)
Jp_z = np.zeros([n,n])
for i in range(1,n):
    val = np.sqrt(j*(j+1)-m[i]*(m[i]+1))
    Jp_z[i-1,i]=val  # we want the i'th element to end up in the (i-1)'th spot
Jm_z = Jp_z.conj().T

# Momentum operators (in the Jz basis)
Jx = (Jp_z + Jm_z)/2
Jy = (Jp_z - Jm_z)/(2j)
```

```python
def uncertainty(bra, ket, J):
    """
    Calculate the uncertainty of a measurement
    sigma(A) = sqrt(<A^2> - <A>^2)
    * all matrices need to be expressed in the same basis
    """
    term1 = bra@J@J@ket  # <J^2>
    term2 = (bra@J@ket)**2  # <J>^2
    return np.sqrt(term1 - term2)
```

```python
# Part B
print("\nPart B")
# Measurement of Jx^2 + Jy^2
# <state|Jx^2+Jy^2|state>
measure_J_sum = state_bra@J_sum@state_ket
print(f"Measurement for Jx^2 + Jy^2: {measure_J_sum.real} hbar^2")

# Uncertainty
# sigma(A) = sqrt(<A^2> - <A>^2)
uncert = uncertainty(state_bra, state_ket, J_sum)
print(f"Uncertainty: {uncert.real}")
```

```
Part B
Measurement for Jx^2 + Jy^2: 2.0 hbar^2
Uncertainty: 0.0
```

Part C) I now send my beam of $|2,2\rangle_z$ particles down an SGx machine. Show that the expectation of J_x^2 that you see is one half of $< J_x^2 + J_y^2 >$ you calculated in part B).

$\langle J_x^2 \rangle = \langle \psi | J_x^2 | \psi \rangle = \langle 2,2 | J_x^2 | 2,2 \rangle = \hbar = \frac{1}{2} \langle J_x^2 + J_y^2 \rangle$

```
90    # Part C
91    print("\nPart C")
92    # <state|Jx^2|state>
93    measure_Jx2 = state_bra@Jx@Jx@state_ket
94    print(f"Measurement for Jx^2: {measure_Jx2.real} hbar^2")
```

```
Part C
Measurement for Jx^2: 1.0 hbar^2
```

Part D) What is the probability that you measure $J_x = +2\hbar$? If this is non-zero, can you explain the apparent contradiction between finding a particle witih $J_x = 2\hbar$ (and hence $J_x^2 = 4\hbar^2$) and the maximum value you found for $J_x^2 + J_y^2$ from part B?

$P(2\hbar) = |{}_x\langle 2,2 | 2,2 \rangle_z|^2 = 0.0625 \neq 0$

↳ prob that the state $|2,2\rangle_z$ is in $|2,2\rangle_x$

```
96    # Part D
97    print("\nPart D")
98    # Jx eigenstates
99    ex,vx = np.linalg.eigh(Jx)
100   ex,vx = np.flipud(ex), np.fliplr(vx)
101   xket = vx
102   xbra = np.conj(vx).T  # expressed in Jz basis, use to rotate to Jx basis
103
104   # Compute probability for Jx eigenstates
105   # (everything is expressed in the Jz basis)
106   prob = np.abs(xbra@state_ket)**2
107   print(f"P(Jx={ex[0]:.0f}hbar) = {prob[0]:.4f}")
```

```
Part D
P(Jx=2hbar) = 0.0625
```

$J_x = 2\hbar$ and we had $J_x^2 + J_y^2 = J^2 - J_z^2 = 2\hbar^2$ but this assumed we had $J_z^2 = 4\hbar^2$

This would be $4\hbar^2$ which doesn't seem possible

If we measure $J_x = 2\hbar$, then we get uncertainty in J_y & J_z so we actually have

$J_x^2 + J_y^2 + J_z^2 = 6\hbar^2$

J_x^2: $4\hbar^2$; J_z^2: NOT $4\hbar^2$

5. **Part A)** Sticking with spin-2 particles, express the pure state $|2,2\rangle_y$ in both the z-basis and the y-basis. In the $z-$basis, please pick the phase that makes the amplitude of $J_z = +2\hbar$ purely real and positive.

```python
# Part A
print("Part A")
# Jy eigenstates
ey,vy = np.linalg.eigh(Jy)
ey,vy = np.flipud(ey), np.fliplr(vy)
yket = vy
ybra = np.conj(yket).T  # expressed in Jz basis, use to rotate to Jy basis

# |2,2>y in the Jz basis
state_y_bra = de_angle(yket)[:,0]
print(f'|2,2>y in Jz basis: {np.round(state_y_bra,2)}')
# other method: transforming from y to z basis
# Jz isnt normalized so need /2 to have the state (1,0,0,0,0) with eigenvalue 2hbar
# instead of (2,0,0,0,0) with eigenvalue hbar
rotated_y_to_z = yket@(Jz[:,0]/2)
phase = np.exp(1j*np.pi)  # phase to make amplitude of Jz=+2hbar real and positive.
print(f'                    {np.round(rotated_y_to_z*phase,2)}')

# |2,2>y in the Jy basis
# this is the same as the |2,2>z in the Jz basis
# dividing by 2 to get the correct normalization
print(f'|2,2>y in Jy basis: {Jz[:,0]/2}')
```

```python
def de_angle(mat):
    mm = mat.copy()
    # this will loop over every column
    for i in range(mat.shape[1]):
        th = np.angle(mat[0,i]) # return theta from a=c exp(i theta)
        mm[:,i] = mm[:,i]/np.exp(1J*th)
    return mm
```

```
Part A
|2,2>y in Jz basis: [ 0.25+0.j  -0.  +0.5j -0.61-0.j   0.  -0.5j  0.25+0.j ]
                    [ 0.25-0.j   0.  +0.5j -0.61+0.j  -0.  -0.5j  0.25-0.j ]
|2,2>y in Jy basis: [1. 0. 0. 0. 0.]
```

Part B) For this same $|2,2\rangle_y$ pure state, what are the uncertainty in J_x and J_z? Show that the uncertainty relation is satisfied.

```python
# Part B
print("\nPart B")
state_y_ket = state_y_bra.conj().T
# Note: uncertainty values are real, using .real to remove the
# imaginary part (which is 0) when printing

# Calculate uncertainties for Jx and Jz
uncert_Jx = uncertainty(state_y_ket, state_y_bra, Jx)
print(f"Uncertainty for Jx: {uncert_Jx.real:.2f} hbar")
uncert_Jz = uncertainty(state_y_ket, state_y_bra, Jz)
print(f"Uncertainty for Jz: {uncert_Jz.real:.2f} hbar")

# Checkig uncertainty relation: σ(Jx)σ(Jz) ≥ 1/2|<Jy>|
print(f"σ(Jx)σ(Jz) = {uncert_Jx.real*uncert_Jz.real:.2f} hbar^2")
print(f"1/2|<Jy>| = {np.abs(state_y_ket@Jy@state_y_bra).real/2:.2f} hbar^2")
```

*see Q4 for def of uncertainty() function in code

```
Part B
Uncertainty for Jx: 1.00 hbar
Uncertainty for Jz: 1.00 hbar
σ(Jx)σ(Jz) = 1.00 hbar^2
1/2|<Jy>| = 1.00 hbar^2
```

Uncertainty relation: $\Delta J_x \Delta J_z \geq \frac{\hbar}{2}|\langle J_y\rangle|$ ($\hbar = 1$ in code)

$\Delta J_x = \hbar$, $\Delta J_z = \hbar$, $|\langle J_y\rangle| = 2\hbar$

$\hbar^2 \geq \hbar^2$

6. **Part A)** Calculate the matrix that rotates a state about the $y-$axis by $+90$ degrees for a spin-2 particle. Print the absolute value of the matrix.

```
157   # Part A
158   print("Part A")
159   # Rotation matrix (90 deg around y-axis)
160   theta = np.pi/2
161   Ry = R_from_J(Jy,theta)
162   print(f'Rotation matrix:\n{np.round(np.abs(Ry),2)}')
163   print(f'det = {np.linalg.det(Ry)}')
164   # Other method
165   R_in_y = R_matrix(theta)  # rotate around y-axis, in Jy basis
166   R = yket@R_in_y@ybra # rotate the state expressed in Jz basis
167   # print(np.round(np.abs(R),2))  # gives same thing as above
168
```

```
Part A
Rotation matrix:
[[0.25 0.5  0.61 0.5  0.25]
 [0.5  0.5  0.   0.5  0.5 ]
 [0.61 0.   0.5  0.   0.61]
 [0.5  0.5  0.   0.5  0.5 ]
 [0.25 0.5  0.61 0.5  0.25]]
det = (0.9999999999999971+4.202006606452647e-16j)
```

```
25  def R_from_J(J,th):
26      """
27      rotation about an axis is exp(-i th J/hbar)
28      """
29      e,v=np.linalg.eigh(J)
30      e_new=-1j*th*e
31      return v@np.diag(np.exp(e_new))@v.conj().T
32
33  def R_matrix(theta):
34      """
35      Rotation matrix for an angle theta
36      rotate a state around n in the n basis
37      """
38      r0 = [np.exp(1j*2*theta),0,0,0,0]
39      r1 = [0,np.exp(1j*theta),0,0,0]
40      r2 = [0,0,1,0,0]
41      r3 = [0,0,0,np.exp(-1j*theta),0]
42      r4 = [0,0,0,0,np.exp(-1j*2*theta)]
43      return np.array([r0,r1,r2,r3,r4])
```

Part B) Show that the above matrix rotates the state $|2,2\rangle_z$ to the pure state $|2,2\rangle_x$, and $|2,2\rangle_x$ to $|2,-2\rangle_z$.

```
169   # Part B
170   print("\nPart B")
171   # Initial state, |2,2>z
172   print(f'|2,2>z: {np.round(state_ket,2)}')
173   # Rotate |2,2>z --> |2,2>x
174   state_rotated = Ry@state_ket
175   print(f"Rotated |2,2>z: {np.round(state_rotated.real,2)}")  # values are real, using .real just for printing clarity
176   state_x_in_z = vx[:,0]
177   print(f"Expected |2,2>x: {np.round(state_x_in_z,2)}")
178
179   # Rotate |2,2>x --> |2,-2>z
180   state_rotated2 = Ry@state_rotated
181   print(f'Rotated |2,2>x: {np.round(state_rotated2,2).real}')  # values are real, using .real just for printing clarity
182   # print(np.round(R@state_x_in_z,2).real)  # gives the same thing! (just another method)
183   state_z2 = Jz[:,4]/(-2)  # dividing by -2 to get the correct normalization for the eigenvalue -2hbar
184   print(f'Expected |2,-2>z: {np.round(state_z2,2)}')
185
```

```
Part B
|2,2>z: [1 0 0 0 0]
Rotated |2,2>z: [0.25 0.5  0.61 0.5  0.25]
Expected |2,2>x: [0.25 0.5  0.61 0.5  0.25]
Rotated |2,2>x: [ 0. -0.  0.  0.  1.]
Expected |2,-2>z: [-0. -0. -0. -0.  1.]
```