Bell type test with a double slit setup

The setup

- Consider the double slit setup as shown [TODO: add an image]
- In a typical Bell test, we have Alice and Bob who measure along [x,x'] or [z,z']. However, this can be equivalently stated as: Alice and Bob apply a unitary, given as $U(\theta) = e^{i\sigma_y \theta}$ with setting $[\theta_1, \theta_2]$ and $[\phi_1, \phi_2]$ respectively, while both measure along x.
  NB: A measurement along x for instance, means σ_x is measured.
- In this setup, Alice and Bob, both measure the modular momentum. They can as before, use the different settings. These apply a relative phase.
- The target is to see if in this setup, we can violate the corresponding inequality:
  $E(x[z - z'] + x'[z + z']) \leq 2$, given that $-1 \leq z, z', x, x' \leq 1$
  - So we start with inputting the setting, one from Alice, one from Bob. Call it γ_A, γ_B. Given these, I find the resultant state, which is essentially just $(q_1 p_1 + \mathrm{Exp}[\phi]\, q_2 p_2)/\sqrt{2}$ with $\phi = \gamma_A + \gamma_B$.
  - To measure x_{mod} I simply measure $e^{i\,\hat{x}\,2\pi/L}$, whose operation is simply to shift the lumps.

The Code

Approach one (only phases, doesn't violate | as intuitively expected)

This has been matched with analytic results for m=2,3 and the rest seem to be correct also.

```
In[104]:= m  = 50;
ex[α_, β_] := (
    (*Print["Initial state:"];*)
    (*stateEntangled=(q1 p1+ Exp[ⅈ (α+β)] q2 p2)/√2;*)
    stateEntangled = (Exp[ⅈ (α+β)] q1 p1 + Sum [qn pn, {n, 2, m }])/√m;
    (*state=(p1+ Exp[ϕ] p2)/√2;*)
    state = stateEntangled;
    (*Print[state];*)
    (* dispRuleP={pn_->pn+1};*)
```

```
    dispRule= {p_n_ -> p_n+1, q_n_ → q_n+1};
    eispRule= {p_n_ -> p_n+1, q_n_ → q_n-1};
    conjDispRule= {p_n_ -> p_n-1, q_n_ → q_n-1};
    conjEispRule= {p_n_ -> p_n-1, q_n_ → q_n+1};
    orthRule = {p_m_  p_n_ → 0, q_m_  q_n_ → 0};
    normRule  = {p_n_ p_n_ → 1, q_n_ q_n_ → 1};
    conjRule = {Exp[ϕ_] → Exp[-ϕ]};

    (*Print["So the expectation for identity is"]*);
    (*newState =state ; (*/. dispRuleP*);*)
    (*ans=Expand[ (state /. conjRule) newState] /. normRule  /. orthRule;*)
    (*Print[ans, " so no surprises there!"]*);

    (*Print["And the expectation for modular  x is "]*);
    newState1 = state /. dispRule;
    newState2 = state /. eispRule;
    (*Print[newState1];*)
    newState3 = state /. conjEispRule; (*Analytically, these are zero*)
    newState4 = state /. conjDispRule;
    (*newState2 and 3 are zero | Look at your notes for the calculations*)
    newState = (newState1 + newState2 + newState3 + newState4) / 4;
    ans = Expand[ (state /. conjRule) newState] /. normRule  /. orthRule;

    Return[ans])

chsh[x_, y_, u_, v_] := ex[x, u] - ex[x, v] + ex[y, u] + ex[y, v]


Print["Expectation for arbitrary ϕ and δ, for n=", m , ":"]
ExpToTrig[ex[ϕ, δ]]
Print["CHSH/Bell:"]
ExpToTrig[chsh[x, y, u, v]] (*/. {u+x → A, v+x→ B}*)

(*ExpToTrig[chsh[0,0,0,0]] *)
(*Plot3D[1/2 Cos[1+x]-1/2 Cos[x]+1/2 Cos[1+y]+1/2 Cos[y],{x,-π,π},{y,-π,π}]*)
(*Maximize [{1/2 Cos[u+x]-1/2 Cos[v+x]+1/2 Cos[u+y]+1/2 Cos[v+y],
    -π≤ u && u≤ π && -π≤ v && v≤ π && -π≤ x&& x≤ π && -π≤ y&&y≤ π},{u,v,x,y}]*)
(*Maximize [{1/2 Cos[(u+v)],u<π && -π<u && v<π && -π<v},{u,v}]*)
(*Maximize [{ 1/(u+1),u>0},{u}]*)
(*Maximize [{1/(x+1),x>  0},{x}]*)

(*
Just testing if the substitution I'm
  doing is indeed linearly independent, its not..;
a={{1,0,1,0},{1,0,0,1},{0,1,1,0},{0,1,0,1}};
a={{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}};
Det[a]
*)
```

Expectation for arbitrary ϕ and δ, for n=50:

Out[108]= $\frac{12}{25}+\frac{1}{100}\text{Cos}[\delta+\phi]$

CHSH/Bell:

Out[110]= $$\frac{24}{25}+\frac{1}{100}\text{Cos}[u+x]-\frac{1}{100}\text{Cos}[v+x]+\frac{1}{100}\text{Cos}[u+y]+\frac{1}{100}\text{Cos}[v+y]$$

Approach two [use a different but closer to the bell situation state]

So far, this is miserable. The expectation of a Hermitian operator is turning up to be complex. So again, I'll have to calculate the whole thing analytically first and then check. Worst part, is that it is not even obvious yet how this should be scaled.

```
In[138]:= m  = 2; (*This is the number  of lumps *)
ex[α_, β_] := (
    (*Print["Initial state:"];*)
    (*stateEntangled=(q1 p1+ Exp[i (α+β)] q2 p2)/Sqrt[2];*)
    stateEntangled = 1/Sqrt[m] (Exp[i (α+β)] q1 p1 + Sum [qn pn, {n, 2, m }]);
    ψ = Simplify [
        1/Sqrt[2] (((Exp[iα] q1 + q2)/Sqrt[2]) ((Exp[iβ] p1 - p2)/Sqrt[2]) - ((Exp[iα] q1 - q2)/Sqrt[2]) ((Exp[iβ] p1 + p2)/Sqrt[2]))];
    Print[ψ];
    (*state=(p1+ Exp[ϕ] p2)/Sqrt[2];*)
    state = ψ;
    (*state=stateEntangled;*)
    (*Print[state];*)
    (* dispRuleP={pn_->pn+1};*)
    dispRule= {pn_ -> pn+1, qn_ → qn+1};
    eispRule= {pn_ -> pn+1, qn_ → qn-1};
    conjDispRule= {pn_ -> pn-1, qn_ → qn-1};
    conjEispRule= {pn_ -> pn-1, qn_ → qn+1};
    orthRule = {pm_  pn_ → 0, qm_  qn_ → 0};
    normRule  = {pn_ pn_ → 1, qn_ qn_ → 1};
    conjRule = {Exp[ ϕ_] → Exp[-ϕ]};
    Print[state /. conjRule];
    (*conjRule = {α_→ -α, β_→ -β};*)

    (*Print["So the expectation for identity is"];
    newState =state ; (*/. dispRuleP*);
    ans=Expand[ (state /. conjRule) newState] /. normRule  /. orthRule;
    Print[ans, " so no surprises there!"];
    *)
    (*Print["And the expectation for modular  x is "]*);
    newState1 = state /. dispRule;
    newState2 = state /. eispRule;
    (*Print[newState1];*)
    newState3 = state /. conjEispRule; (*Analytically, these are zero*)
    newState4 = state /. conjDispRule;
    (*newState2 and 3 are zero | Look at your notes for the calculations*)
    newState = (newState1 + newState2 + newState3 + newState4) / 4;
    ans = Expand[ (state /. conjRule) newState] /. normRule  /. orthRule;
    Return[ans])
chsh[x_, y_, u_, v_] := ex[x, u] - ex[x, v] + ex[y, u] + ex[y, v]


(*Print["Expectation for arbitrary ϕ and δ, for n=",m ,":"]*)
ex[ϕ, δ]
(*Print["CHSH/Bell:"]
  ExpToTrig[chsh[x,y,u,v]] (*/. {u+x → A, v+x→ B}*)
*)
```

$$\frac{-e^{i\alpha}p_2 q_1 + e^{i\beta}p_1 q_2}{\sqrt{2}}$$

$$\frac{-e^{-i\alpha}p_2 q_1 + e^{-i\beta}p_1 q_2}{\sqrt{2}}$$

Out[140]= $-\frac{e^{i\alpha - i\beta}}{8} - \frac{e^{-i\alpha + i\beta}}{8}$