Ultimate local-realistic simulation of two level entangled state event by event.
By Fred Diether with 2D vectors, also with Joy's hidden variable and modified epr-simple HV by Michel Fodje. Some parts by John Reed and Bill Nelson. 6-14-2021

In[138]:=
```
<< Quaternions`;
β0 = Quaternion[1, 0, 0, 0];
β1 = Quaternion[0, 1, 0, 0];
β2 = Quaternion[0, 0, 1, 0];
β3 = Quaternion[0, 0, 0, 1];
Qcoordinates = {β1, β2, β3};
m = 200000;
λ2 = ConstantArray[0, m];
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
lista = Table[{0, 0, 0}, m];
listb = Table[{0, 0, 0}, m];
lista1 = Table[{0, 0, 0}, m];
listb1 = Table[{0, 0, 0}, m];
lista2 = Table[{0, 0, 0}, m];
listb2 = Table[{0, 0, 0}, m];
lista3 = Table[{0, 0, 0}, m];
listb3 = Table[{0, 0, 0}, m];
lista4 = Table[{0, 0, 0}, m];
listb4 = Table[{0, 0, 0}, m];
lista5 = Table[{0, 0, 0}, m];
listb5 = Table[{0, 0, 0}, m];
lista6 = Table[{0, 0, 0}, m];
listb6 = Table[{0, 0, 0}, m];
lista7 = Table[{0, 0, 0}, m];
listb7 = Table[{0, 0, 0}, m];
lista8 = Table[{0, 0, 0}, m];
listb8 = Table[{0, 0, 0}, m];
```

In[166]:=
```
Do[vectorS = Flatten[{RandomPoint[Circle[]], 0}]; (*Singlet spin vector*)
 s1 = ToSphericalCoordinates[vectorS][[3]];
 λ2[[i]] = 1/4 Cos[(s1 * 90)/2]^2; (*hidden variable*)
 λ = Sign[Cos[s1]]; (*hidden variable*)
 Ls1a = λ * vectorS.Qcoordinates; (*singlet spin quaternion*)
 Ls2b = -Ls1a;
 Ls1[[i]] = Ls1a;
 Ls2[[i]] = Ls2b, {i, m}]
```

In[167]:=
```
Do[vectorA = Flatten[{RandomPoint[Circle[]], 0}];
 Da = vectorA.Qcoordinates;  (*Convert to quaternion coordinates*)
 AT = ArcTan[vectorA[[1]], vectorA[[2]]];
 AS = Abs[Re[Da ** Ls1[[i]]]] ≤ λ2[[i]];
 If[1 π/8 ≤ AT + π ≤ π/2, Aa1 = -1, Aa1 = nothing];
 If[29 π/16 ≤ AT + π ≤ 2 π/1, Aa2 = 1, Aa2 = nothing];
 If[3 π/4 ≤ AT + π ≤ 3 π/2, Aa3 = 1, Aa3 = nothing];
 If[0 π/4 ≤ AT + π ≤ 4 π/16, Aa4 = 1, Aa4 = nothing];
 If[0 π/4 ≤ AT + π ≤ 6 π/16, Aa5 = -1, Aa5 = nothing];
 If[1 π/16 ≤ AT + π ≤ 3 π/16, Aa6 = 1, Aa6 = nothing];
 lista1[[i]] = {AT, Aa1, m + i};
 lista2[[i]] = {AT, Aa2, 2 m + i};
 lista3[[i]] = {AT, Aa3, 3 m + i};
 lista4[[i]] = {AT, Aa4, 4 m + i};
 lista5[[i]] = {AT, Aa5, 5 m + i};
 lista6[[i]] = {AT, Aa6, 6 m + i};
 If[Abs[Re[Da ** Ls1[[i]]]] > λ2[[i]], A = Sign[Re[Da ** Ls1[[i]]]], A = nothing];
 lista[[i]] = {AT, A, i}, {i, m}]
lista7 = Catenate[{lista, lista1, lista2, lista3, lista4, lista5, lista6}];
lista8 = DeleteCases[lista7, {_, nothing, _}];
outA = lista8;
```

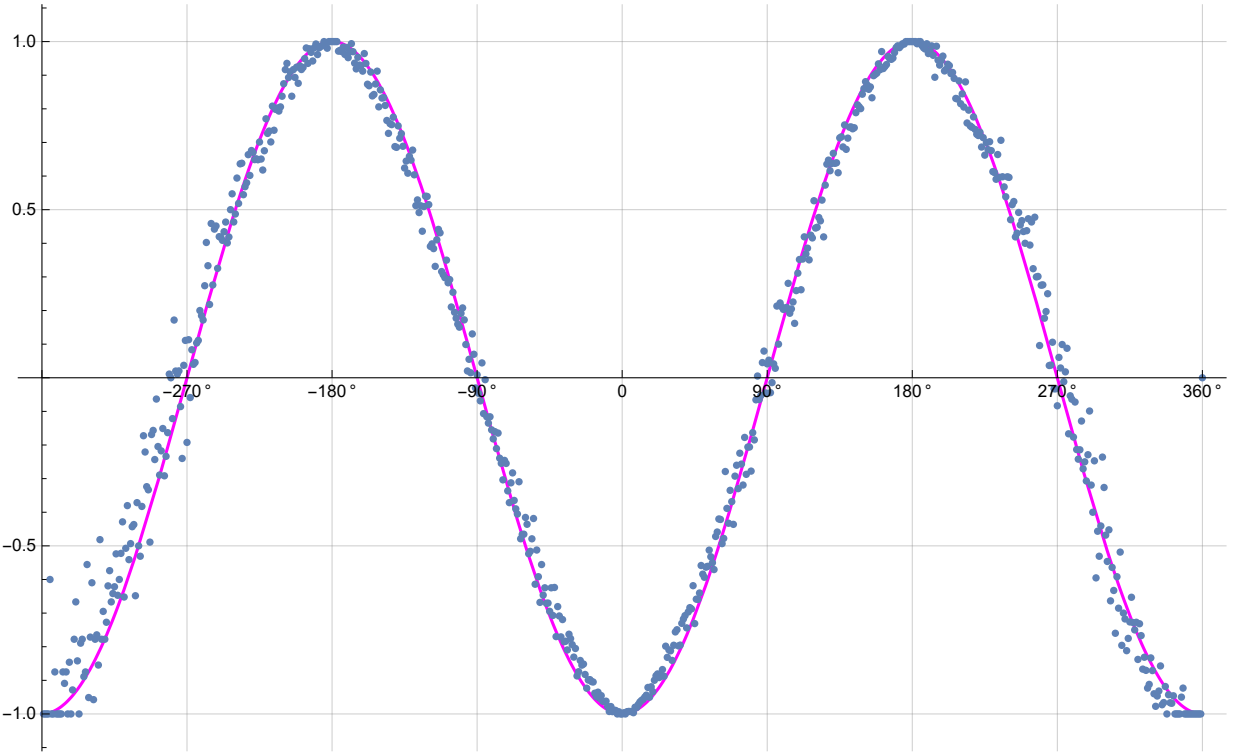
In[171]:=
```
Do[vectorB = Flatten[{RandomPoint[Circle[]], 0}];
 Db = vectorB.Qcoordinates;
 BT = ArcTan[vectorB[[1]], vectorB[[2]]];
 BS = Abs[Re[Ls2[[i]] ** Db]] ≤ λ2[[i]];
 If[0 π/4 ≤ BT + π ≤ π/4, Bb1 = 1, Bb1 = nothing];
 If[0 π/4 ≤ BT + π ≤ 3 π/16, Bb2 = -1, Bb2 = nothing];
 If[0 π/4 ≤ BT + π ≤ π/4, Bb3 = 1, Bb3 = nothing];
 If[3 π/4 ≤ BT + π ≤ 3 π/2, Bb4 = 1, Bb4 = nothing];
 If[1 π/4 ≤ BT + π ≤ π/2, Bb5 = 1, Bb5 = nothing];
 If[30 π/16 ≤ BT + π ≤ 2 π/1, Bb6 = -1, Bb6 = nothing];
 listb1[[i]] = {BT, Bb1, m + i};
 listb2[[i]] = {BT, Bb2, 2 m + i};
 listb3[[i]] = {BT, Bb3, 3 m + i};
 listb4[[i]] = {BT, Bb4, 4 m + i};
 listb5[[i]] = {BT, Bb5, 5 m + i};
 listb6[[i]] = {BT, Bb6, 6 m + i};
 If[Abs[Re[Ls2[[i]] ** Db]] > λ2[[i]], B = Sign[Re[Ls2[[i]] ** Db]], B = nothing];
 listb[[i]] = {BT, B, i}, {i, m}]
listb7 = Catenate[{listb, listb1, listb2, listb3, listb4, listb5, listb6}];
listb8 = DeleteCases[listb7, {_, nothing, _}];
outB = listb8;
```

Match trial numbers and do statistical analysis

```
listad = outA[[All, 3]];  (*Match Trial Numbers*)
listbd = outB[[All, 3]];
listAa = Take[Select[outA, Intersection[{#[[3]]}, listbd] == {#[[3]]} &], m];
listBb = Take[Select[outB, Intersection[{#[[3]]}, listad] == {#[[3]]} &], m];
trials = Length[listAa];
trialDeg = 720;
a1 = ConstantArray[0, trials];
b1 = ConstantArray[0, trials];
A1 = ConstantArray[0, trials];
B1 = ConstantArray[0, trials];
nPP = ConstantArray[0, trialDeg];
nNN = ConstantArray[0, trialDeg];
nPN = ConstantArray[0, trialDeg];
nNP = ConstantArray[0, trialDeg];
nAP = ConstantArray[0, trialDeg];
nBP = ConstantArray[0, trialDeg];
nAN = ConstantArray[0, trialDeg];
nBN = ConstantArray[0, trialDeg];
a1 = listAa[[All, 1]];
b1 = listBb[[All, 1]];
A1 = listAa[[All, 2]];
B1 = listBb[[All, 2]];
```

In[194]:=
```
Do[a2 = a1[[j]];
 b2 = b1[[j]];
 t1 = Round[(a2 - b2) / Degree + 360];
 aliceD = A1[[j]]; bobD = B1[[j]];
 If[aliceD == 1, nAP[[t1]] ++];
 If[bobD == 1, nBP[[t1]] ++];
 If[aliceD == -1, nAN[[t1]] ++];
 If[bobD == -1, nBN[[t1]] ++];
 If[aliceD == 1 && bobD == 1, nPP[[t1]] ++];
 If[aliceD == 1 && bobD == -1, nPN[[t1]] ++];
 If[aliceD == -1 && bobD == 1, nNP[[t1]] ++];
 If[aliceD == -1 && bobD == -1, nNN[[t1]] ++], {j, trials}]
pPP = 0; pPN = 0; pNP = 0; pNN = 0;
mean = ConstantArray[0, trialDeg];
Do[sum = nPP[[i]] + nPN[[i]] + nNP[[i]] + nNN[[i]];
If[sum == 0, Goto[jump],
  {pPP = nPP[[i]] / sum;
   pNP = nNP[[i]] / sum;
   pPN = nPN[[i]] / sum;
   pNN = nNN[[i]] / sum;
mean[[i]] = (pPP + pNN - pPN - pNP)}];
 Label[jump], {i, trialDeg}]
sim = ListPlot[(mean), PlotMarkers → {Automatic, Tiny}];
negcos = Plot[-Cos[x Degree], {x, 0, 720}, PlotStyle → {Magenta},
   Ticks → {{{0, -360 °}, {90, -270 °}, {180, -180 °}, {270, -90 °}, {360, 0 °}, {450, 90 °},
      {540, 180 °}, {630, 270 °}, {720, 360 °}}, Automatic}, GridLines → Automatic];
Show[
 negcos,
 sim]
```

Out[200]=



In[201]:=
```
AveA = N[Sum[A1[[i]], {i, trials}] / trials];
AveB = N[Sum[B1[[i]], {i, trials}] / trials];
Print["AveA = ", AveA]
Print["AveB = ", AveB]
totAB = Sum[nPP[[i]] + nNN[[i]] + nPN[[i]] + nNP[[i]], {i, trialDeg}]
CHSH = Abs[N[mean[[45]]] + N[mean[[45]]] - N[mean[[135]]] + N[mean[[45]]]];
Print["CHSH = ", CHSH]
```

```
AveA = 0.05283

AveB = 0.13051
```

Out[205]= 200 000

```
CHSH = 2.56611
```

In[208]:=
```
trials
```

Out[208]= 200 000