Simulation Based on Joy Christian's updated 3-Sphere model and Michel Fodje's epr-simple simulation translated from Python to Mathematica by John Reed 13 Nov 2013 plus some Quaternion Parts and using John Reed's trial number matching code. Modified, Created by Fred Diether for Completely Local-Realistic Feb. 2022

Load Quaternion Package, Set Run Time Parameters, Initialize Arrays and Tables

```
In[1]:= << 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 = 2000000; (*Number of events to perform.*)
trialDeg = 720;
ss = ConstantArray[0, m];
λ = ConstantArray[0, m];
λ1 = ConstantArray[0, m];
λ2 = ConstantArray[0, m];
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
outqA = Table[{0, 0, 0}, m];
outqB = Table[{0, 0, 0}, m];
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];
ϕ = 3; β = 0.23; ξ = -15;(*Adjustable parameters*)
```

Generating Particle Data with Three Independent Do-Loops

```
In[28]:= Do[s = RandomPoint[Sphere[]];  (*Singlet 3D vector; Hidden Variable*)
 θ1 = ToSphericalCoordinates[s][[3]] * 180 / π;
 ss[[k]] = θ1;
 λ[[k]] = β (Cos[θ1/ϕ]^2);  (*Hidden variable mechanism*)
 Ls1[[k]] = s.Qcoordinates;  (*Convert to quaternion coordinates; A particle spin*)
 Ls2[[k]] = -s.Qcoordinates, {k, m}] (*B particle spin plus conservation of angular momentum*)

In[29]:= Do[a = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
 a1[[k]] = a;
 aa = N[Flatten[{FromPolarCoordinates[{1, a * π / 180}], 0}]];
 Da = aa.Qcoordinates;  (*Convert to quaternion coordinates; A detector*)
 qa = Da ** Ls1[[k]];
 qA = Sign[Re[qa] + (Sin[(a - ss[[k]] + ξ) Degree]) / 4];
 A0 = Sign[Sin[(a - ss[[k]] + ξ) Degree]];
 outqA[[k]] = {a, qA, A0};
 If[Abs[Re[qa]] > λ[[k]], λ1[[k]] = 0, λ1[[k]] = k], {k, m}]
outqA3 = outqA;

In[31]:= Do[b = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
 b1[[k]] = b;
 bb = N[Flatten[{FromPolarCoordinates[{1, b * π / 180}], 0}]];
 Db = bb.Qcoordinates;  (*Convert to quaternion coordinates; B detector*)
 qb = Ls2[[k]] ** Db;
 qB = Sign[Re[qb] - (Sin[(b - ss[[k]] + ξ) Degree]) / 4];
 B0 = -Sign[Sin[(b - ss[[k]] + ξ) Degree]];
 outqB[[k]] = {b, qB, B0};
 If[Abs[Re[qb]] > λ[[k]], λ2[[k]] = 0, λ2[[k]] = k], {k, m}]
outqB3 = outqB;
```

Spinorial Sign Changes in A and B

For the spinorial sign changes we will need,

$$\mathbf{q}(\eta_{\text{sn}} + \delta\,\pi,\, \mathbf{r}) = (-1)^{\delta}\, \mathbf{q}(\eta_{\text{sn}},\, \mathbf{r}) \quad \text{for } \delta = 0, 1, 2, 3, \ldots$$

```
In[33]:= ssca = ConstantArray[0, m];
sscb = ConstantArray[0, m];
Do[If[λ2[[k]] == k && outqA[[k]][[2]] ≠ outqA[[k]][[3]], outqA3[[k]][[2]] = outqA[[k]][[2]] * -1];
 (*Spinorial sign change*)
 If[λ2[[k]] == k && outqA[[k]][[2]] ≠ outqA[[k]][[3]], ssca[[k]] = 1, ssca[[k]] = 0], {k, m}]
A = outqA3[[All, 2]];

In[37]:= Do[If[λ1[[k]] == k && outqB[[k]][[2]] ≠ outqB[[k]][[3]], outqB3[[k]][[2]] = outqB[[k]][[2]] * -1];
 (*Spinorial sign change*)
 If[λ1[[k]] == k && outqB[[k]][[2]] ≠ outqB[[k]][[3]], sscb[[k]] = 1, sscb[[k]] = 0], {k, m}]
B = outqB3[[All, 2]];
N[Total[ssca] / m] * 100 (*Percentage of spinorial sign changes.*)
N[Total[sscb] / m] * 100

Out[39]= 3.3382

Out[40]= 3.3387
```

Statistical Analysis of the Particle Data Received from Alice and Bob

```
In[41]:= Do[θ2 = a1[[k]] - b1[[k]] + 360;  (*Angles shifted by 360 since θ2 is an index*)
 aliceD = A[[k]]; bobD = B[[k]];
 If[aliceD == 1, nAP[[θ2]] ++];
 If[bobD == 1, nBP[[θ2]] ++];
 If[aliceD == -1, nAN[[θ2]] ++];
 If[bobD == -1, nBN[[θ2]] ++];
 If[aliceD == 1 && bobD == 1, nPP[[θ2]] ++];
 If[aliceD == 1 && bobD == -1, nPN[[θ2]] ++];
 If[aliceD == -1 && bobD == 1, nNP[[θ2]] ++];
 If[aliceD == -1 && bobD == -1, nNN[[θ2]] ++], {k, m}]
```

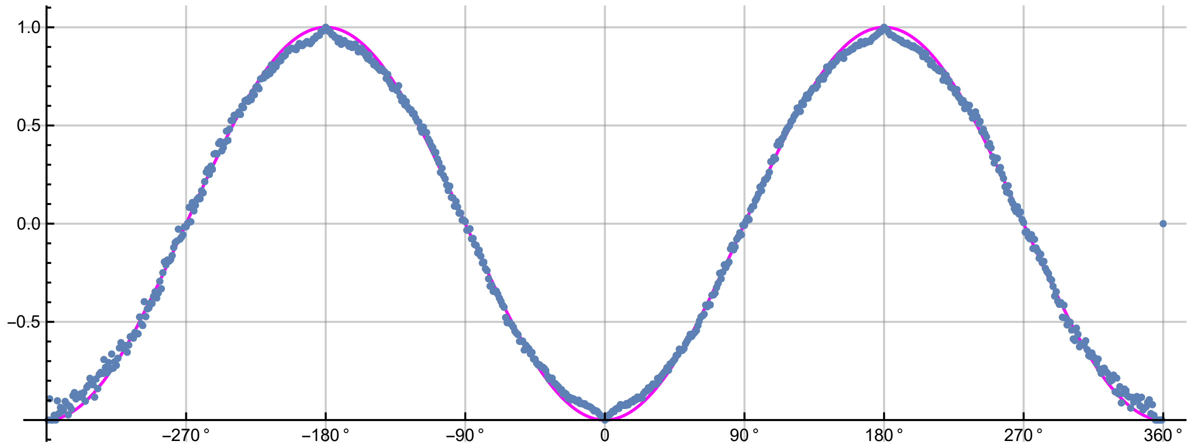
Calculating Mean Values of AB

```
In[42]:= mean = ConstantArray[0, trialDeg];
sum1 = ConstantArray[0, trialDeg];
sum2 = ConstantArray[0, trialDeg];
Do[sum1[[i]] = (nPP[[i]] + nNN[[i]] - nPN[[i]] - nNP[[i]]);
 sum2[[i]] = nPP[[i]] + nPN[[i]] + nNP[[i]] + nNN[[i]] + 0.0000001;
mean[[i]] = sum1[[i]] / sum2[[i]], {i, trialDeg}]
```

Plotting the Results Comparing Mean Values with -Cosine Curve

```
In[46]:= sim2 = ListPlot[mean, PlotMarkers → {Automatic, Tiny}];
negcos = Plot[-Cos[x7 Degree], {x7, 0, 720}, PlotStyle → {Magenta}, AspectRatio → 3 / 8,
   Ticks → {{{0, -360 °}, {90, -270 °}, {180, -180 °}, {270, -90 °}, {360, 0 °}, {450, 90 °},
      {540, 180 °}, {630, 270 °}, {720, 360 °}}, Automatic}, GridLines → Automatic,
   AxesOrigin → {0, -1.0}];
p5 = Plot[-5 + 2 x5 Degree / π, {x5, 360, 540}, PlotStyle → {Gray, Dashed}];
p6 = Plot[7 - 2 x6 Degree / π, {x6, 540, 720}, PlotStyle → {Gray, Dashed}];
Show[negcos, sim2]
```

Out[50]=



Computing Averages

```
In[51]:= AveA = N[Sum[A⟦k⟧, {k, m}] / m];
AveB = N[Sum[B⟦k⟧, {k, m}] / m];
Print["AveA  = ", AveA]
Print["AveB  = ", AveB]
PAP = N[Sum[nAP⟦i⟧, {i, trialDeg}]];
PBP = N[Sum[nBP⟦i⟧, {i, trialDeg}]];
PAN = N[Sum[nAN⟦i⟧, {i, trialDeg}]];
PBN = N[Sum[nBN⟦i⟧, {i, trialDeg}]];
PA1 = PAP / (PAP + PAN);
PB1 = PBP / (PBP + PBN);
Print["P(A+) = ", PA1]
Print["P(B+) = ", PB1]
totAB = Total[nPP + nNN + nPN + nNP];
Print["Total Events Detected = ", totAB]
PP = N[Sum[nPP⟦i⟧, {i, trialDeg}] / totAB];
NN = N[Sum[nNN⟦i⟧, {i, trialDeg}] / totAB];
PN = N[Sum[nPN⟦i⟧, {i, trialDeg}] / totAB];
NP = N[Sum[nNP⟦i⟧, {i, trialDeg}] / totAB];
Print["P(++)  = ", PP]
Print["P(--)  = ", NN]
Print["P(+-)  = ", PN]
Print["P(-+)  = ", NP]
CHSH = Abs[N[mean⟦315⟧] - N[mean⟦225⟧] + N[mean⟦405⟧] + N[mean⟦45⟧]];
Print["Approx. CHSH = ", CHSH]
```

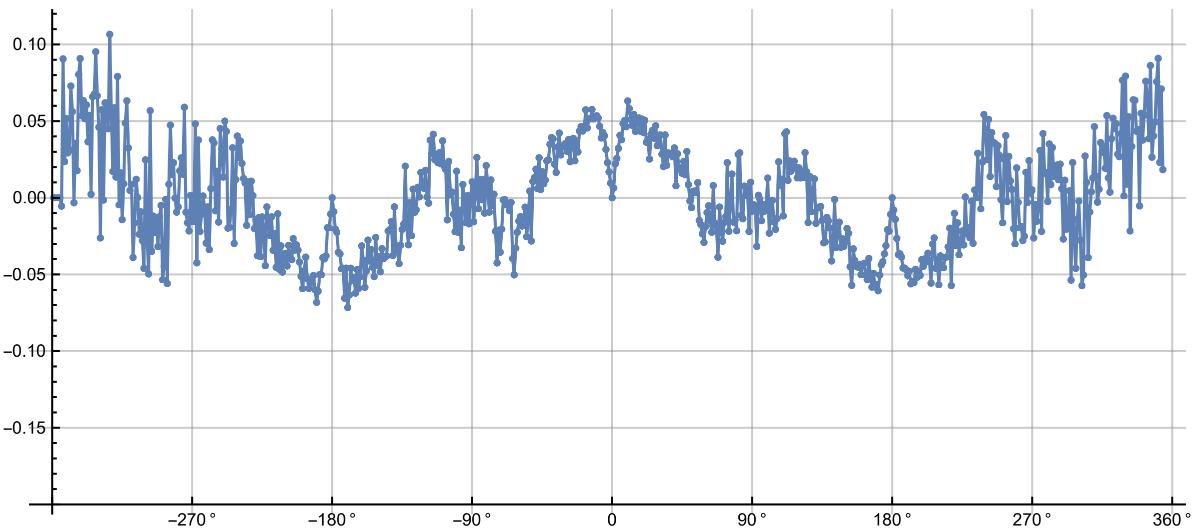
```
AveA = -0.000124

AveB = 0.002534

P(A+) = 0.499938

P(B+) = 0.501267

Total Events Detected = 2000000

P(++) = 0.250544

P(--) = 0.249339

P(+-) = 0.249395

P(-+) = 0.250724

Approx. CHSH = 2.78905
```

Deviation from Negative Cosine Curve

In[75]:=

```
dev = ConstantArray[0, 714];
devang = ConstantArray[0, 714];
Do[devang[[i]] = i;
dev[[i]] = mean[[i]] + Cos[devang[[i]] Degree], {i, 6, 714}]
ListPlot[N[dev], PlotMarkers → {Automatic, Tiny}, Joined → True,
 AspectRatio → 7/16, Ticks → {{{0, -360 °}, {90, -270 °}, {180, -180 °}, {270, -90 °},
    {360, 0 °}, {450, 90 °}, {540, 180 °}, {630, 270 °}, {720, 360 °}}, Automatic},
 GridLines → Automatic, AxesOrigin → {0, -0.2}]
```

Out[78]=



In[79]:=

```
Mean[N[dev]]
Mean[N[Abs[dev]]]
```

Out[79]= `0.0000534223`

Out[80]= `0.0285102`

In[81]:=

```
0.000210912; 0.0224585; (*deviations, 1 million events*)
```