Simulation Based on Joy Christian's new 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 Jan. 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};
Qcoordinates2 = {β0, β1, β2, β3};
m = 6000000; (*Number of events to perform.*)
trialDeg = 720;
ss1 = ConstantArray[0, m];
λ = ConstantArray[0, m];
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
aa1 = ConstantArray[0, m];
bb1 = ConstantArray[0, m];
qa1 = ConstantArray[0, m];
qb1 = ConstantArray[0, m];
outqA = Table[{0, 0, 0, 0}, m];
outqB = Table[{0, 0, 0, 0}, m];
h1 = ConstantArray[0, m];
h2 = ConstantArray[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*)
```

We note here that the Limit[x, x → Sign[x]] = Sign[x] so that we just use the sign function in the Do-loops instead of the limits.

Generating Particle Data with Three Independent Do-Loops

In[33]:=
```
Do[s = RandomPoint[Sphere[]];  (*Singlet 3D vector; Hidden Variable*)
 θ1 = ToSphericalCoordinates[s][[3]] * 180 / π;
 ss1[[i]] = θ1;
 λ[[i]] = β (Cos[θ1/ϕ]^2);  (*Hidden variable mechanism*)
 Ls1[[i]] = s.Qcoordinates;  (*Convert to quaternion coordinates; A particle spin*)
 Ls2[[i]] = -s.Qcoordinates, {i, m}] (*B particle spin plus conservation of angular momentum*)
```

In[34]:=
```
Do[a = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
 a1[[i]] = a;
 aa = N[Flatten[{FromPolarCoordinates[{1, a * π / 180}], 0.000001}]];
 aa1[[i]] = aa;
 Da = aa.Qcoordinates;  (*Convert to quaternion coordinates; A detector*)
 qa = Da ** Ls1[[i]];
 qa1[[i]] = qa;
 If[Abs[Re[qa]] > λ[[i]], qA = Sign[Re[qa]], qA = Sign[Sin[(a - ss1[[i]] + ξ) Degree]]];
 A0 = Sign[Sin[(a - ss1[[i]] + ξ) Degree]];
 outqA[[i]] = {a, qA, i, A0};
 If[Abs[Re[qa]] > λ[[i]], h1[[i]] = 0, h1[[i]] = outqA[[i]][[3]]], {i, m}]
outqA3 = outqA;
```

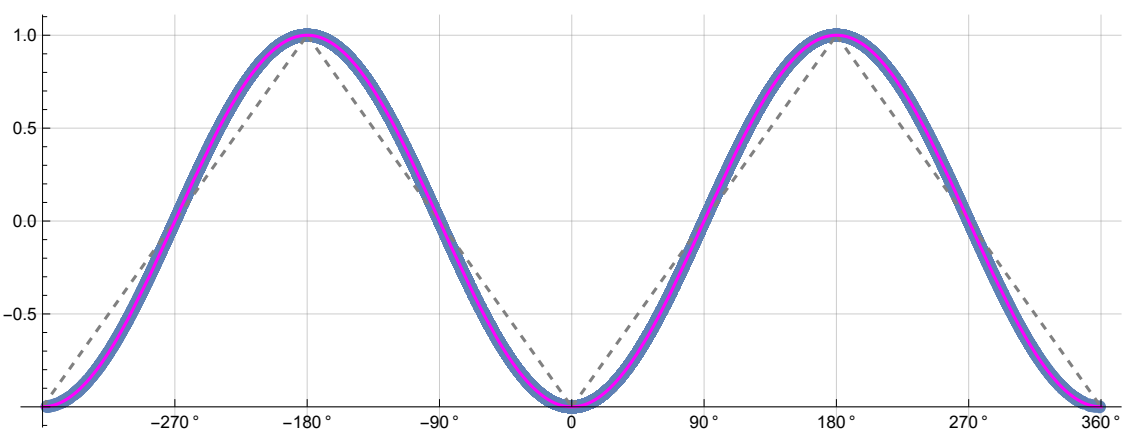
In[36]:=
```
Do[b = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
 b1[[i]] = b;
 bb = N[Flatten[{FromPolarCoordinates[{1, b * π / 180}], 0.000001}]];
 bb1[[i]] = bb;
 Db = bb.Qcoordinates;  (*Convert to quaternion coordinates; B detector*)
 qb = Ls2[[i]] ** Db;
 qb1[[i]] = qb;
 If[Abs[Re[qb]] > λ[[i]], qB = Sign[Re[qb]], qB = Sign[Sin[-(b - ss1[[i]] + ξ) Degree]]];
 B0 = Sign[Sin[-(b - ss1[[i]] + ξ) Degree]];
 outqB[[i]] = {b, qB, i, B0};
 If[Abs[Re[qb]] > λ[[i]], h2[[i]] = 0, h2[[i]] = outqB[[i]][[3]]], {i, m}]
outqB3 = outqB;
```

Verification of the Analytical 3-Sphere Model Product Calculation

In[38]:=
```
m2 = 20000;
r0 = ConstantArray[0, m2];
r1 = ConstantArray[0, m2];
r2 = ConstantArray[0, m2];
QAB = ConstantArray[0, m2];
plotq = Table[{0, 0}, m2];
(*qA=Re[Da**Limit[Ls1⟦i⟧,Ls1⟦i⟧→Sign[Re[Da**Ls1⟦i⟧]]Da]];
qB=Re[Db**Limit[Ls2⟦i⟧,Ls2⟦i⟧→Sign[Re[Db**Ls2⟦i⟧]]Db]];*)
(*The above two lines are moved to the independent A and B Do-loops
  and modified for proper calculation of A and B outcomes.*)
Do[r1⟦i⟧ = {qa1⟦i⟧⟦2⟧, qa1⟦i⟧⟦3⟧, qa1⟦i⟧⟦4⟧};
 r2⟦i⟧ = {qb1⟦i⟧⟦2⟧, qb1⟦i⟧⟦3⟧, qb1⟦i⟧⟦4⟧};
 QAB⟦i⟧ = Re[qa1⟦i⟧] * Re[qb1⟦i⟧] - r1⟦i⟧.r2⟦i⟧;
r0⟦i⟧ = (Re[qa1⟦i⟧] Limit[Cross[s2, bb1⟦i⟧], s2 → Sign[Re[qb1⟦i⟧]] bb1⟦i⟧]
      + Re[qb1⟦i⟧] Limit[Cross[aa1⟦i⟧, s1], s1 → Sign[Re[qa1⟦i⟧]] aa1⟦i⟧]
      - Cross[Limit[Cross[aa1⟦i⟧, s1], s1 → Sign[Re[qa1⟦i⟧]] aa1⟦i⟧],
       Limit[Cross[s2, bb1⟦i⟧], s2 → Sign[Re[qb1⟦i⟧]] bb1⟦i⟧]]) /
    (Sin[ArcCos[a1⟦i⟧.b1⟦i⟧]]);
 q = {Re[QAB⟦i⟧], r0⟦i⟧⟦1⟧, r0⟦i⟧⟦2⟧, r0⟦i⟧⟦3⟧}.Qcoordinates2;
 θ = a1⟦i⟧ - b1⟦i⟧ + 360;
 plotq⟦i⟧ = {θ, Re[q]}, {i, m2}]
meanq = Mean[plotq⟦All, 2⟧];
Print["Imaginary part vanishes. meanq = ", meanq]
sim = ListPlot[plotq, PlotMarkers → {Automatic, Small}, AspectRatio → 3 / 8,
   Ticks → {{{450, 90 °}, {90, -270 °}, {540, 180 °}, {180, -180 °}, {630, 270 °}, {270, -90 °},
      {720, 360 °}, {360, 0 °}}, Automatic}, GridLines → Automatic, AxesOrigin → {0, -1.0}];
p1 = Plot[-1 + 2 x Degree / π, {x, 0, 180}, PlotStyle → {Gray, Dashed}];
p2 = Plot[3 - 2 x Degree / π, {x, 180, 360}, PlotStyle → {Gray, Dashed}];
p3 = Plot[-5 + 2 x Degree / π, {x, 360, 540}, PlotStyle → {Gray, Dashed}];
p4 = Plot[7 - 2 x Degree / π, {x, 540, 720}, PlotStyle → {Gray, Dashed}];
negcos = Plot[-Cos[x Degree], {x, 0, 720}, PlotStyle → {Magenta}];
Show[sim, p1, p2, p3, p4, negcos]
```

```
Imaginary part vanishes. meanq = 0.00392211
```

Out[53]=



Blue is the correlation data, magenta is the negative cosine curve for an exact match.

Spinorial Sign Changes in A and B

For the spinorial sign changes we will need,

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

In[54]:=
```
ssca = ConstantArray[0, m];
sscb = ConstantArray[0, m];
Do[If[h2⟦i⟧ ≠ 0 && outqA⟦i⟧⟦2⟧ ≠ outqA⟦i⟧⟦4⟧, outqA3⟦i⟧⟦2⟧ = outqA⟦i⟧⟦2⟧ * -1], {i, m}]
(*Spinorial sign change*)
Do[If[h2⟦i⟧ ≠ 0 && outqA⟦i⟧⟦2⟧ ≠ outqA⟦i⟧⟦4⟧, ssca⟦i⟧ = 1, ssca⟦i⟧ = 0], {i, m}]
A = outqA3⟦All, 2⟧;
```

In[59]:=
```
Do[If[h1⟦i⟧ ≠ 0 && outqB⟦i⟧⟦2⟧ ≠ outqB⟦i⟧⟦4⟧, outqB3⟦i⟧⟦2⟧ = outqB⟦i⟧⟦2⟧ * -1], {i, m}]
(*Spinorial sign change*)
Do[If[h1⟦i⟧ ≠ 0 && outqB⟦i⟧⟦2⟧ ≠ outqB⟦i⟧⟦4⟧, sscb⟦i⟧ = 1, sscb⟦i⟧ = 0], {i, m}]
B = outqB3⟦All, 2⟧;
N[Total[ssca] / m] * 100 (*Percentage of spinorial sign changes.*)
N[Total[sscb] / m] * 100
```

Out[62]= 3.32185

Out[63]= 3.32443

Statistical Analysis of the Particle Data Received from Alice and Bob

In[64]:=
```
Do[θ2 = a1⟦i⟧ - b1⟦i⟧ + 360;  (*Angles shifted by 360 since θ2 is an index*)
 aliceD = A⟦i⟧; bobD = B⟦i⟧;
 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⟧ ++], {i, m}]
```

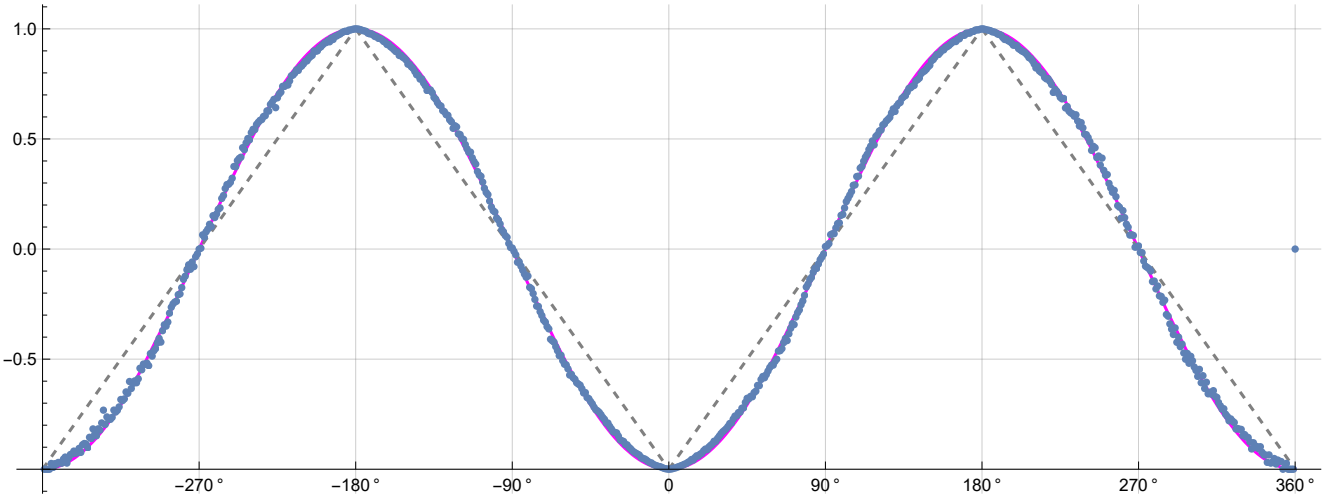
Calculating Mean Values of AB

In[65]:=
```
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[69]:=
```
sim2 = ListPlot[mean, PlotMarkers → {Automatic, Tiny}];
negcos = Plot[-Cos[x Degree], {x, 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}];
Show[negcos, p1, p2, p3, p4, sim2]
```

Out[71]=



Computing Averages

In[72]:=
```
AveA = N[Sum[A[[i]], {i, m}] / m];
AveB = N[Sum[B[[i]], {i, 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]
```

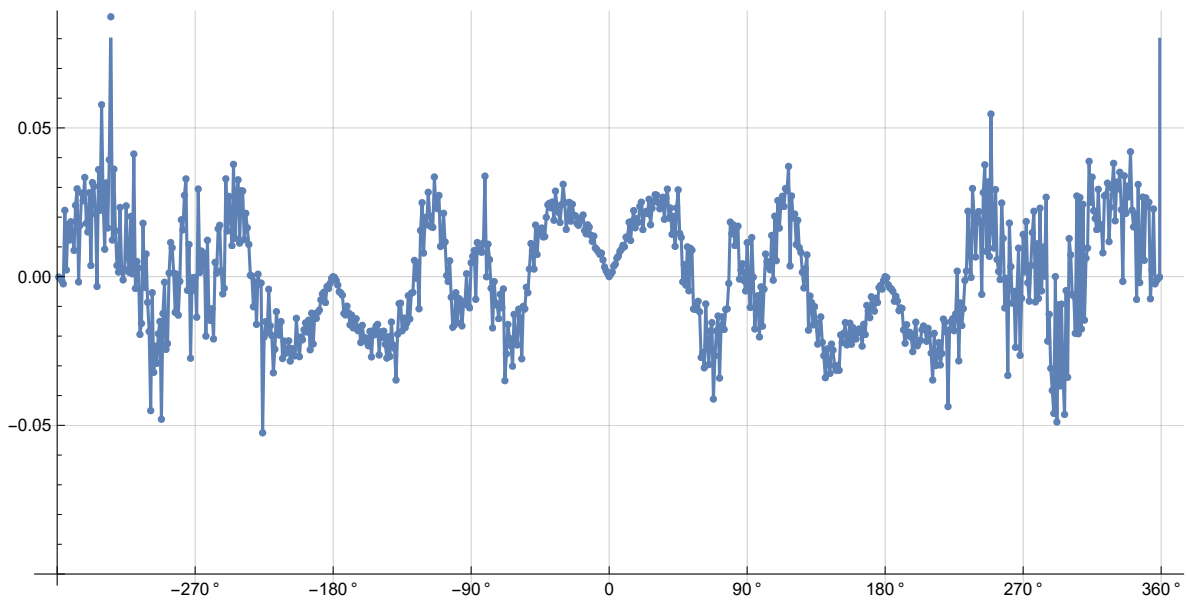
```
AveA = 0.000445

AveB = 0.000727667

P(A+) = 0.500223

P(B+) = 0.500364

Total Events Detected = 6000000

P(++) = 0.250314

P(--) = 0.249728

P(+-) = 0.249908

P(-+) = 0.25005

Approx. CHSH = 2.74253
```

Deviation from Negative Cosine Curve

In[96]:=
```
dev1 = ConstantArray[2, 720];
dev2 = ConstantArray[2, 720];
dev3 = ConstantArray[2, 720];
Do[dev1 = mean⟦i⟧; dev2⟦i⟧ = {dev1, i}, {i, 720}]
devang = dev2⟦All, 2⟧;
```

In[101]:=
```
Do[dev3⟦i⟧ = mean⟦i⟧ + Cos[devang⟦i⟧ Degree], {i, 720}]
ListPlot[N[dev3], PlotMarkers → {Automatic, Tiny}, Joined → True,
 AspectRatio → 8 / 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, -0.1}]
```

Out[102]=



In[103]:=
```
Mean[N[dev3]]
Mean[N[Abs[dev3]]]
```

Out[103]= 0.00164826

Out[104]= 0.01741

In[105]:=
```
-0.00521477;  0.0227552; (*deviations, 1 million events*)
```