**Simulation Based on Michel Fodje's epr-simple and**
**Joy Christian's Updated 3-Sphere Model.**
**Parts of Quaternion and Matching Code by John Reed.**
**Created by Fred Diether, Jan. 2022.**

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

```
In[847]:= << 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 = 50000;  (*Number of trials to perform.*)
trialDeg = 721;
ss = ConstantArray[0, m];
ss1 = ConstantArray[0, m];
λ = ConstantArray[0, m];
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
qa1 = ConstantArray[0, m];
qb1 = ConstantArray[0, m];
h1 = ConstantArray[0, m];
h2 = ConstantArray[0, m];
Da1 = ConstantArray[0, m];
outqA = Table[{0, 0, 0, 0}, m];
outqB = Table[{0, 0, 0, 0}, m];
outA12 = Table[{0, 0, 0, 0}, m];
outB12 = Table[{0, 0, 0, 0}, m];
outA22 = Table[{0, 0, 0, 0}, m];
outB22 = Table[{0, 0, 0, 0}, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
aa1 = ConstantArray[0, m];
bb1 = 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*)
```

## Generating Particle Data with Three Independent Do-Loops

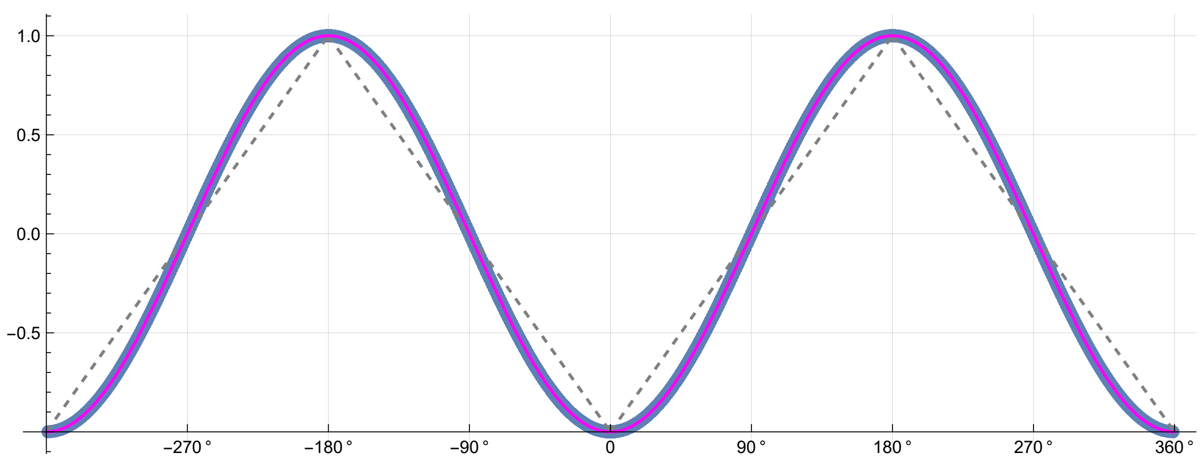
```
In[884]:= Do[s = RandomPoint[Sphere[]];  (*Singlet 3D vector*)  (*Hidden Variable*)
  ss⟦i⟧ = s;
  θ1 = ToSphericalCoordinates[s]⟦3⟧ * 180 / π;
  ss1⟦i⟧ = θ1;
  λ⟦i⟧ = β (Cos[θ1/ϕ]^2);  (*Hidden variable mechanism*)
  Ls1⟦i⟧ = s.Qcoordinates;  (*Convert to quaternion coordinates*)(*A particle spin*)
  Ls2⟦i⟧ = -s.Qcoordinates, {i, m}]
 (*B particle spin plus conservation of angular momentum*)

In[885]:= Do[a = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
  aa = N[Flatten[{FromPolarCoordinates[{1, a * π / 180}], 0.000001}]];
  aa1⟦i⟧ = aa;
  Da = aa.Qcoordinates;  (*Convert to quaternion coordinates*)
  Da1⟦i⟧ = N[Flatten[{FromPolarCoordinates[{1, a * π / 180}], 0}]].Qcoordinates;
  a1⟦i⟧ = a;
  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⟧, outA12⟦i⟧ = outqA⟦i⟧, outA22⟦i⟧ = outqA⟦i⟧], {i, m}]
 outqA2 = DeleteCases[outA22, {0, 0, 0, 0}];
 outqA3 = outqA;
 Do[qA2tn = outqA2⟦i⟧⟦3⟧;  (*Trial numbers from outA2*)
  h1⟦qA2tn⟧ = 1, {i, Length[outqA2]}]

In[889]:= Do[b = RandomInteger[{-179, 180}];  (*Detector 2D vector angle 1 degree increments*)
  bb = N[Flatten[{FromPolarCoordinates[{1, b * π / 180}], 0.000001}]];
  bb1⟦i⟧ = bb;
  Db = bb.Qcoordinates;  (*Convert to quaternion coordinates*)
  b1⟦i⟧ = b;
  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⟧, outB12⟦i⟧ = outqB⟦i⟧, outB22⟦i⟧ = outqB⟦i⟧], {i, m}]
 outqB2 = DeleteCases[outB22, {0, 0, 0, 0}];
 outqB3 = outqB;
 Do[qB2tn = outqB2⟦i⟧⟦3⟧;  (*Trial numbers from outB2*)
  h2⟦qB2tn⟧ = 1, {i, Length[outqB2]}]
```

## Verification of the Updated Analytical 3-Sphere Model Product Calculation

In[893]:=
```
m2 = m;
r0 = ConstantArray[0, m2];
r1 = ConstantArray[0, m2];
r2 = ConstantArray[0, m2];
QAB = ConstantArray[0, m2];
plotq = Table[{0, 0}, m2];
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⟧];  (*Shows the complete vanishing of imaginary parts*)
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 = 0.00306566
```

Out[907]=




**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,

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

In[908]:=
```
ssca = ConstantArray[0, m];
sscb = ConstantArray[0, m];
Do[If[h2⟦i⟧ == 1 && outqA⟦i⟧⟦2⟧ ≠ outqA⟦i⟧⟦4⟧, outqA3⟦i⟧⟦2⟧ = outqA⟦i⟧⟦2⟧ * -1], {i, m}]
(*Spinorial sign change*)
Do[If[h2⟦i⟧ == 1 && outqA⟦i⟧⟦2⟧ ≠ outqA⟦i⟧⟦4⟧, ssca⟦i⟧ = 1, ssca⟦i⟧ = 0], {i, m}]
A = outqA3⟦All, 2⟧;
a1 = outqA3⟦All, 1⟧;
```

In[914]:=
```
Do[If[h1⟦i⟧ == 1 && outqB⟦i⟧⟦2⟧ ≠ outqB⟦i⟧⟦4⟧, outqB3⟦i⟧⟦2⟧ = outqB⟦i⟧⟦2⟧ * -1], {i, m}]
(*Spinorial sign change*)
Do[If[h1⟦i⟧ == 1 && outqB⟦i⟧⟦2⟧ ≠ outqB⟦i⟧⟦4⟧, sscb⟦i⟧ = 1, sscb⟦i⟧ = 0], {i, m}]
B = outqB3⟦All, 2⟧;
b1 = outqB3⟦All, 1⟧;
N[Total[ssca] / m] * 100 (*Percentage of spin sign changes*)
N[Total[sscb] / m] * 100
```

Out[918]= 3.368

Out[919]= 3.388

## Statistical Analysis of the Particle Data Received from Alice and Bob

In[920]:=
```
theta = ConstantArray[0, m];
Do[θ2 = a1⟦i⟧ - b1⟦i⟧ + 360;
 theta⟦i⟧ = θ2;
 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}]
```

## Computing Averages

In[922]:=
```
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]
```

```
AveA = 0.0034

AveB = -0.00656

P(A+)= 0.5017

P(B+)= 0.49672

Total Events Detected = 50000

P(++) = 0.2508

P(--) = 0.25238

P(+-) = 0.2509

P(-+) = 0.24592
```