Simulation Based on Michel Fodje's epr-simple simulation translated from Python to Mathematica by John Reed 13 Nov 2013 Plus Quaternions Modified by Fred Diether for Completely Local-Realistic Oct. 2021 Includes Joy's S^3 Quaternion Model. With 3D Vectors!

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 = 6 000 000;
trialDeg = 361;
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
λ1 = ConstantArray[0, m];
λ2 = ConstantArray[0, m];
λ3 = ConstantArray[0, m];
Daa = ConstantArray[0, m];
Dbb = ConstantArray[0, m];
qA = ConstantArray[0, m];
qB = ConstantArray[0, m];
aa1 = ConstantArray[0, m];
bb1 = ConstantArray[0, m];
outA1 = Table[{0, 0}, m];
outA2 = Table[{0, 0}, m];
outB1 = Table[{0, 0}, m];
outB2 = Table[{0, 0}, m];
a1 = ConstantArray[0, m];
b1 = 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 = 0.284; ξ = 0.892; (*Adustable parameters for fine tuning*)
```

Generating Particle Data with Three Independent Do-Loops

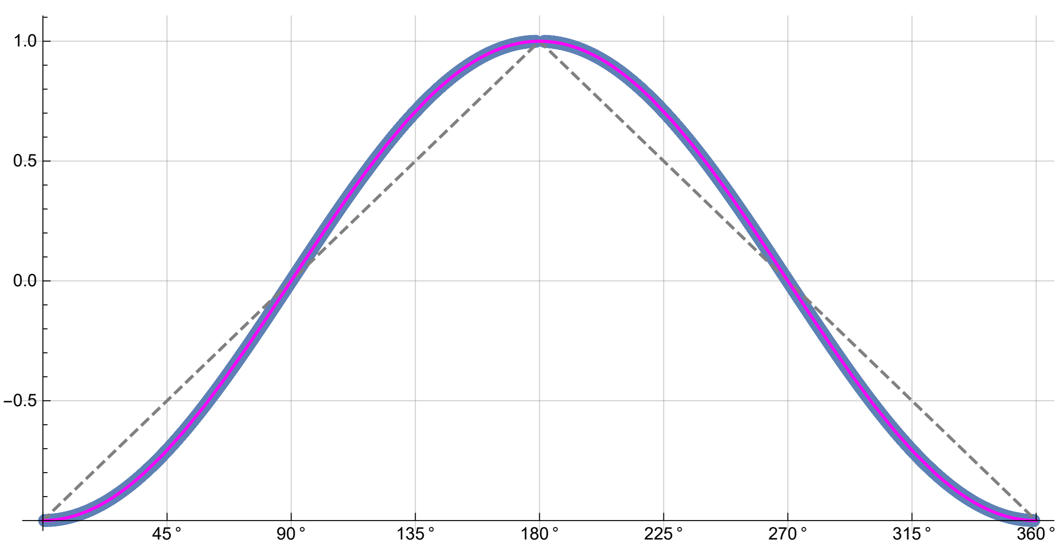
```
In[35]:= Do[θ = RandomPoint[Sphere[]]; (*Singlet 3D vector*) (*Hidden Variable*)
  θ1 = ToSphericalCoordinates[θ][[3]] * 180 / π;
  θ2 = ToSphericalCoordinates[θ][[2]];
  λ1[[i]] = β (Cos[θ1/ϕ]^2);
  λ2[[i]] = (Cos[(θ2 * ξ)/2]^2);
  λ3[[i]] = Sign[θ1];
  Ls1[[i]] = λ3[[i]] * θ.Qcoordinates; (*Convert to quaternion coordinates*)
  Ls2[[i]] = -λ3[[i]] * θ.Qcoordinates, {i, m}]

In[36]:= Do[a = RandomPoint[Sphere[]]; (*Detector 3D vector angle*)
  aa1[[i]] = a;
  Da = a.Qcoordinates; (*Convert to quaternion coordinates*)
  Daa[[i]] = Da;
  qa = Da ** Ls1[[i]];
  aq = -Da ** Ls1[[i]];
  If[Abs[Re[qa]] > λ1[[i]],
   qA1 = Re[Da ** Limit[Ls1[[i]], Ls1[[i]] → Sign[Re[Da ** Ls1[[i]]]] Da]],
   qA1 = Sign[aq[[4]]] + 0.001];
  outA1[[i]] = {a, qA1};
  If[Abs[Re[qa]] > λ2[[i]],
   qA2 = Re[Da ** Limit[Ls1[[i]], Ls1[[i]] → Sign[Re[Da ** Ls1[[i]]]] Da]],
   qA2 = Sign[aq[[4]]] + 0.001];
  outA2[[i]] = {a, qA2}, {i, m}]
 outA = Catenate[{outA1, outA2}];

In[38]:= Do[b = RandomPoint[Sphere[]]; (*Detector 3D vector angle*)
  bb1[[i]] = b;
  Db = b.Qcoordinates; (*Convert to quaternion coordinates*)
  Dbb[[i]] = Db;
  qb = Ls2[[i]] ** Db;
  bq = -Ls2[[i]] ** Db;
  If[Abs[Re[qb]] > λ1[[i]],
   qB1 = Re[Db ** Limit[Ls2[[i]], Ls2[[i]] → Sign[Re[Db ** Ls2[[i]]]] Db]],
   qB1 = Sign[bq[[4]]] + 0.001];
  outB1[[i]] = {b, qB1};
  If[Abs[Re[qb]] > λ2[[i]],
   qB2 = Re[Db ** Limit[Ls2[[i]], Ls2[[i]] → Sign[Re[Db ** Ls2[[i]]]] Db]],
   qB2 = Sign[bq[[4]]] + 0.001];
  outB2[[i]] = {b, qB2}, {i, m}]
 outB = Catenate[{outB1, outB2}];
```

VERIFICATION OF THE ANALYTICAL 3-SPHERE MODEL BASED ON GEOMETRIC ALGEBRA USING QUATERNIONS

In[40]:=
```
q = 0; s = 0;
m3 = 20000;
plotq = Table[{0, 0}, m3];
angle = ConstantArray[0, m3];
DA = Take[Daa, m3];
DB = Take[Dbb, m3];
Ls11 = Take[Ls1, m3];
Ls22 = Take[Ls2, m3];
(*qA=Re[DA**Limit[Ls11[[i]],Ls11[[i]]→Sign[Re[DA**Ls11[[i]]]]DA]];
qB=Re[DB**Limit[Ls22[[i]],Ls22[[i]]→Sign[Re[DB[[i]]**Ls22[[i]]]]DB]];*)
(*These two lines moved to the A and B Do-
 loops for further proper local processing*)
Do[If[λ3[[i]] == 1, q = Limit[DA[[i]] ** Ls11[[i]] ** Ls22[[i]] ** DB[[i]],
     {Ls11[[i]] → Sign[Re[DA[[i]] ** Ls11[[i]]]] DA[[i]],
      Ls22[[i]] -> Sign[Re[Ls22[[i]] ** DB[[i]]]] DB[[i]]}],
   q = Limit[DB[[i]] ** Ls22[[i]] ** Ls11[[i]] ** DA[[i]],
     {Ls22[[i]] -> Sign[Re[Ls22[[i]] ** DB[[i]]]] DB[[i]],
      Ls11[[i]] → Sign[Re[DA[[i]] ** Ls11[[i]]]] DA[[i]]}]];
  ϕA = ArcTan[aa1[[i]][[1]], aa1[[i]][[2]]] / 50;
  ϕB = ArcTan[bb1[[i]][[2]], bb1[[i]][[1]]] / 50;
  If[ϕA * ϕB > 0, angle = ArcCos[aa1[[i]].bb1[[i]]] * 180 / π,
   angle = (2 π - ArcCos[aa1[[i]].bb1[[i]]]) * 180 / π];
  s = s + q;
  plotq[[i]] = {angle, Re[q]}, {i, m3}]
Meanq = FromQuaternion[s / m3]; (*shows vanishing of the non-real part iJK*)
Print["Meanq = ", Meanq]
sim = ListPlot[plotq, PlotMarkers → {Automatic, Small},
   AspectRatio → 8 / 16, Ticks → {{{0, 0 °}, {45, 45 °}, {90, 90 °}, {135, 135 °},
      {180, 180 °}, {225, 225 °}, {270, 270 °}, {315, 315 °}, {360, 360 °}}, 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}];
negcos1 = Plot[-Cos[x Degree], {x, 0, 360}, PlotStyle → {Magenta}];
Show[sim, p1, p2, negcos1]
```

Meanq = (0.00146492 − 0.00418864 i) − 0.00487291 J + 0.00432036 K

Out[55]=



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

Statistical Analysis of the Particle Data Received from Alice and Bob

In[56]:=
```
m2 = 2 m;
theta = ConstantArray[0, m2];
th1 = ConstantArray[0, m2];
a1 = outA[[All, 1]];
qA = outA[[All, 2]];
b1 = outB[[All, 1]];
qB = outB[[All, 2]];
Do[ϕA1 = ArcTan[a1[[i]][[1]], a1[[i]][[2]]] / 50;
 ϕB1 = ArcTan[b1[[i]][[2]], b1[[i]][[1]]] / 50;
 If[ϕA1 * ϕB1 > 0, th1[[i]] = ArcCos[a1[[i]].b1[[i]]],
  th1[[i]] = 2 π - ArcCos[a1[[i]].b1[[i]]]];
 theta[[i]] = Round[th1[[i]] * 180 / π] + 1;
 th = theta[[i]];
 aliceD = qA[[i]]; bobD = qB[[i]];
 If[aliceD == 1, nAP[[th]] ++];
 If[bobD == 1, nBP[[th]] ++];
 If[aliceD == -1, nAN[[th]] ++];
 If[bobD == -1, nBN[[th]] ++];
 If[aliceD == 1 && bobD == 1, nPP[[th]] ++];
 If[aliceD == 1 && bobD == -1, nPN[[th]] ++];
 If[aliceD == -1 && bobD == 1, nNP[[th]] ++];
 If[aliceD == -1 && bobD == -1, nNN[[th]] ++], {i, m2}]
```

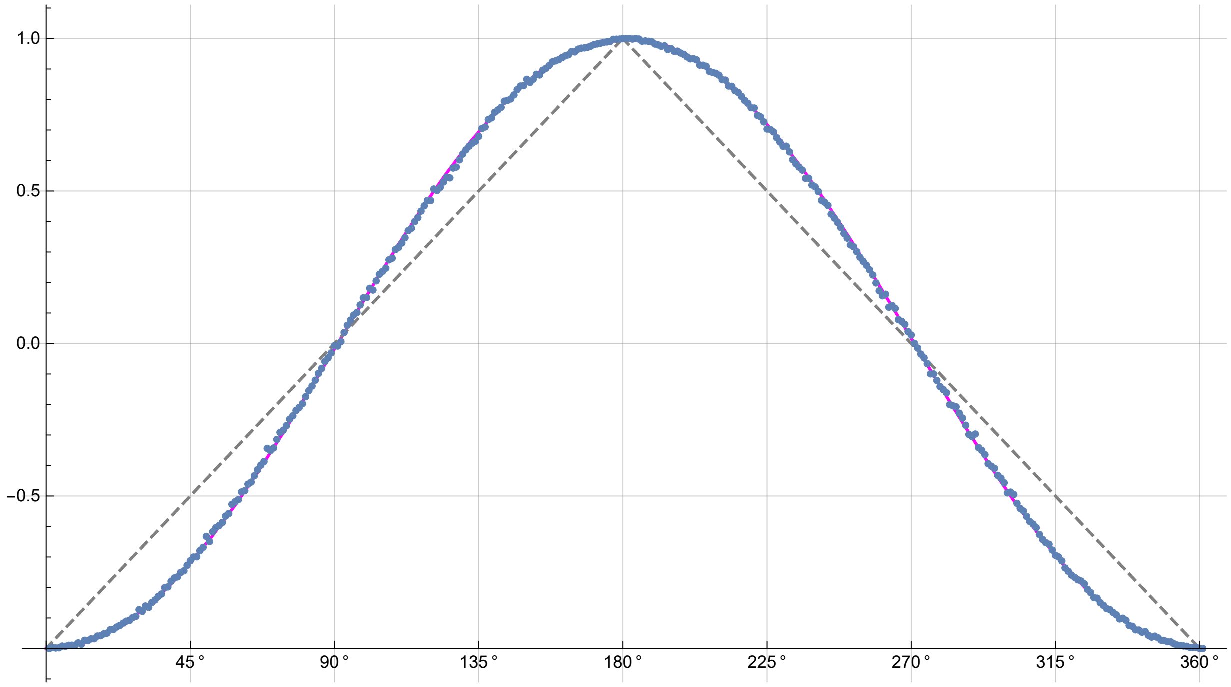
Calculating Mean Values of AB

In[61]:=
```
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 and Comparing Mean Values with -Cosine Function

In[65]:=
```
simulation = ListPlot[mean, PlotMarkers → {Automatic, Tiny}];
negcos =
  Plot[-Cos[x Degree - 1 Degree], {x, 0, 361}, PlotStyle → {Magenta}, AspectRatio → 9 / 16, Ticks →
    {{{0, 0 °}, {45, 45 °}, {90, 90 °}, {135, 135 °}, {180, 180 °}, {225, 225 °}, {270, 270 °},
      {315, 315 °}, {360, 360 °}}, 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}];
Show[negcos, p1, p2, simulation]
```

Out[69]=



Computing Averages

In[70]:=
```
A1 = ConstantArray[0, m2];
B1 = ConstantArray[0, m2];
Do[If[qA[[i]] == 1 || qA[[i]] == -1, A1[[i]] = qA[[i]]];
 If[qB[[i]] == 1 || qB[[i]] == -1, B1[[i]] = qB[[i]]], {i, m2}]
AveA = N[Sum[A1[[i]], {i, m2}] / m2];
AveB = N[Sum[B1[[i]], {i, m2}] / m2];
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 = Sum[nPP[[i]] + nNN[[i]] + nPN[[i]] + nNP[[i]], {i, trialDeg}];
Print["Total Events = ", 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];
totP = PP + NN + PN + NP;
Print["Ave ++ = ", PP]
Print["Ave -- = ", NN]
Print["Ave +- = ", PN]
Print["Ave -+ = ", NP]
CHSH = Abs[N[mean[[23]]] - N[mean[[135]]] + N[mean[[68]]] + N[mean[[45]]]];
Print["Approx. CHSH = ", CHSH]
```

```
AveA = 0.00068225

AveB = 0.000214417

P(A+)= 0.50053

P(B+)= 0.500166

Total Events = 6000834

Ave ++ = 0.250214

Ave -- = 0.249492

Ave +- = 0.25024

Ave -+ = 0.250054

Approx. CHSH = 2.70335
```

In[98]:=

$$E_{ab} = \text{TrigReduce}\left[\frac{\text{Sin}[(\eta_{ab})/2]^2}{2} + \frac{\text{Sin}[(\eta_{ab})/2]^2}{2} - \frac{\text{Cos}[(\eta_{ab})/2]^2}{2} - \frac{\text{Cos}[(\eta_{ab})/2]^2}{2}\right] \Big/ \text{totP};$$

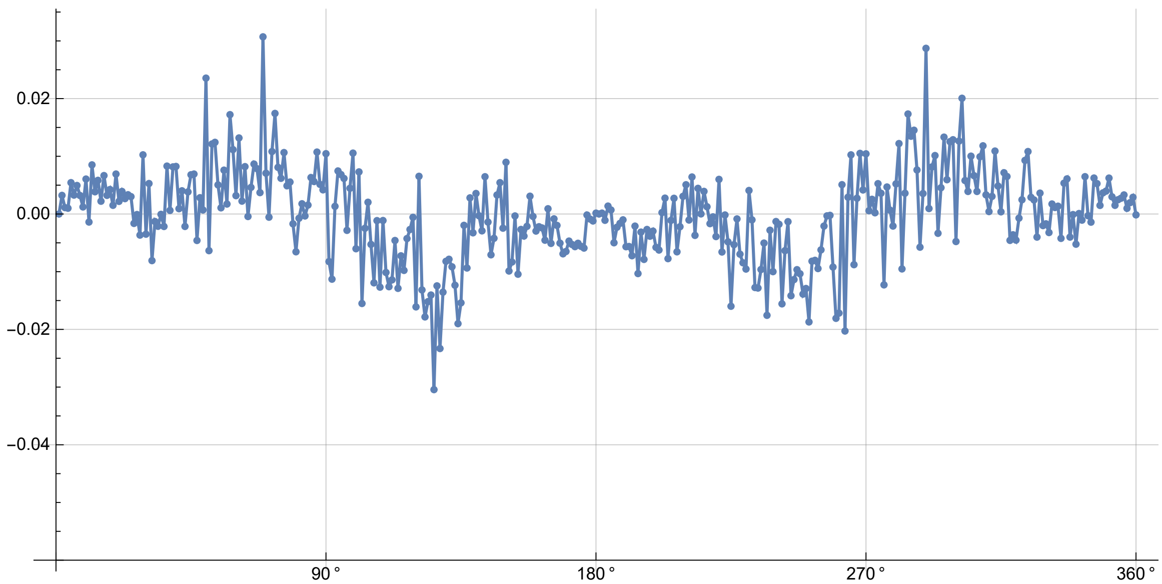
Print["E(a, b) = ", E_{ab}]

E(a, b) = -1. Cos[η_{ab}]

Calculating Deviation from -Cosine Curve

In[100]:=
```
dev1 = ConstantArray[2, 360];
dev2 = ConstantArray[2, 360];
dev3 = ConstantArray[2, 360];
Do[dev1 = mean[[i]];
 dev2[[i]] = {dev1, i}, {i, 360}]
devang = dev2[[All, 2]];
Do[dev3[[i]] = mean[[i]] + Cos[devang[[i]] Degree - 1 Degree], {i, 360}]
ListPlot[N[dev3], PlotMarkers → {Automatic, Tiny}, Joined → True, AspectRatio → 1/2,
 Ticks → {{{0, 0 °}, {90, 90 °}, {180, 180 °}, {270, 270 °}, {360, 360 °}}, Automatic},
 GridLines → Automatic, AxesOrigin → {0, -0.06}]
```

Out[106]=



In[107]:=
```
N[Mean[Abs[dev3]]]
N[Mean[dev3]]
```

Out[107]= 0.00595887

Out[108]= -0.000216198