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

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

In[429]:=
```
<< 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 = 20000;
trialDeg = 721;
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
λ1 = ConstantArray[0, m];
λ2 = ConstantArray[0, m];
outA = Table[{0, 0}, m];
outB = Table[{0, 0}, m];
plotq = Table[{0, 0}, m];
Da1 = ConstantArray[0, m];
Db1 = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
A = ConstantArray[0, m];
B = ConstantArray[0, m];
ϕ = 3; β = 0.3; ξ = -π/12;        (*Adustable parameters for fine tuning*)
```

Generating Particle Data with Three Independent Do-Loops

In[451]:=
```
Do[e = RandomReal[{-179, 180}];  (*Singlet vector angle*)  (*Hidden Variable*)
 λ1[[i]] = β (Cos[e/ϕ]^2);
 λ2[[i]] = Sign[e];
 ee = N[Flatten[{FromPolarCoordinates[{1, e * π/180}], 0}]];
 Ls1[[i]] = λ2[[i]] * ee.Qcoordinates;
 Ls2[[i]] = -λ2[[i]] * ee.Qcoordinates, {i, m}]
```

In[452]:=
```
Do[a = RandomInteger[{-179, 180}];  (*Detector vector angle 1 degree increments*)
 aa = N[Flatten[{FromPolarCoordinates[{1, a * π/180}], 0}]];
 Da = aa.Qcoordinates;  (*Convert to quaternion coordinates*)
 Da1[[i]] = Da;
 qa = Da ** Ls1[[i]];
 If[Abs[Re[qa]] > λ1[[i]],
  A = Re[Da ** Limit[ Ls1[[i]], Ls1[[i]] -> Sign[Re[Da ** Ls1[[i]]]] Da]],
  A = λ2[[i]] * Sign[qa[[4]] + ξ]];
 outA[[i]] = {a, A}, {i, m}]
```

In[453]:=
```
Do[b = RandomInteger[{-179, 180}];  (*Detector vector angle 1 degree increments*)
 bb = N[Flatten[{FromPolarCoordinates[{1, b * π / 180}], 0}]];
 Db = bb.Qcoordinates;  (*Convert to quaternion coordinates*)
 Db1[[i]] = Db;
 qb = Ls2[[i]] ** Db;
 If[Abs[Re[qb]] > λ1[[i]],
  B = Re[Db ** Limit[ Ls2[[i]], Ls2[[i]] -> Sign[Re[Db ** Ls2[[i]]]] Db]],
  B = -λ2[[i]] * Sign[qb[[4]] + ξ]];
 outB[[i]] = {b, B}, {i, m}]
```

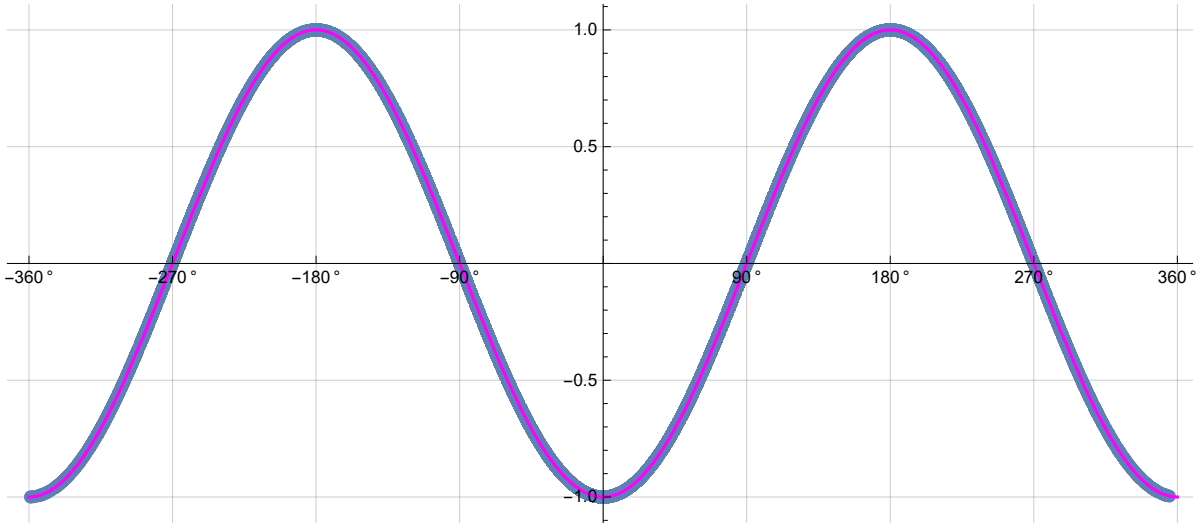
Product Calculation

In[454]:=
```
a1 = outA[[All, 1]]; b1 = outB[[All, 1]]; A = outA[[All, 2]]; B = outB[[All, 2]];
q = 0; s = 0; t = 0; u = 0;
Do[If[λ2[[i]] == 1,
  q = Limit[(Da1[[i]] ** Ls1[[i]] ** Ls2[[i]] ** Db1[[i]]), {Ls1[[i]], Ls2[[i]]} →
     {Sign[Re[Da1[[i]] ** Ls1[[i]]]] Da1[[i]], Sign[Re[Db1[[i]] ** Ls2[[i]]]] Db1[[i]]}],
  q = Limit[(Db1[[i]] ** Ls2[[i]] ** Ls1[[i]] ** Da1[[i]]), {Ls1[[i]], Ls2[[i]]} →
     {Sign[Re[Da1[[i]] ** Ls1[[i]]]] Da1[[i]], Sign[Re[Db1[[i]] ** Ls2[[i]]]] Db1[[i]]}]];
 angle = a1[[i]] - b1[[i]];
 s = s + q; t = t + A[[i]]; u = u + B[[i]];
 plotq[[i]] = {angle, Re[q]}, {i, m}]
Meanq = FromQuaternion[s / m]; (*shows vanishing of the non-real part iK*)
Print["Meanq = ", Meanq]
aveA = t / m;
aveB = u / m;
Print[" <A> = ", aveA, "  <B> = ", aveB]
simulation1 = ListPlot[plotq, PlotMarkers → {Automatic, Small}, AspectRatio → 7 / 16,
   Ticks → {{{-360, -360 °}, {-270, -270 °}, {-180, -180 °}, {-90, -90 °}, {0, 0 °}, {90, 90 °},
      {180, 180 °}, {270, 270 °}, {360, 360 °}}, Automatic}, GridLines → Automatic];
negcos1 = Plot[-Cos[x Degree], {x, -360, 360}, PlotStyle → {Magenta}];
Show[simulation1, negcos1]
```

```
Meanq = (-0.00164109 + 0. i) + 0.00168218 K
```

```
<A> = -0.0036  <B> = -0.0021
```

Out[463]=



Blue is the correlation data and magenta is the -cosine curve for an exact match.