Simulation Based on Joy Christian's original 3-Sphere Model.
Created by Fred Diether Feb. 2022. With 3D Vectors!

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

In[420]:=
```
<< 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 = 20 000;
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
qa1 = ConstantArray[0, m];
qb1 = ConstantArray[0, m];
A = ConstantArray[0, m];
B = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
pc = ConstantArray[0, m];
plotpc = Table[{0, 0}, m];
```

Generating Particle Data with Three Independent Do-Loops

In[438]:=
```
Do[s = RandomPoint[Sphere[]]; (*Singlet 3D vector*) (*Hidden Variable*)
 Ls1[[i]] = s.Qcoordinates; (*Convert to quaternion coordinates*)
 Ls2[[i]] = -s.Qcoordinates, {i, m}]

Do[a = RandomPoint[Sphere[]]; (*Detector 3D vector angle*)
 a1[[i]] = a;
 Da = a.Qcoordinates; (*Convert to quaternion coordinates*)
 qa = Da ** Ls1[[i]]; (*Particle spin - detector interaction*)
 qa1[[i]] = qa;
 A[[i]] = Re[Da ** Limit[Ls1[[i]], Ls1[[i]] → Sign[Re[Da ** Ls1[[i]]]] Da]], {i, m}]

Do[b = RandomPoint[Sphere[]]; (*Detector 3D vector angle*)
 b1[[i]] = b;
 Db = b.Qcoordinates; (*Convert to quaternion coordinates*)
 qb = Ls2[[i]] ** Db; (*Particle spin - detector interaction*)
 qb1[[i]] = qb;
 B[[i]] = Re[Db ** Limit[Ls2[[i]], Ls2[[i]] → Sign[Re[Db ** Ls2[[i]]]] Db]], {i, m}]
```

Verification of the Analytical 3-Sphere Model Prediction Based on Geometric Algebra using Quaternions

In[441]:=
```
Do[r1 = {qa1[[i]][[2]], qa1[[i]][[3]], qa1[[i]][[4]]};
 r2 = {qb1[[i]][[2]], qb1[[i]][[3]], qb1[[i]][[4]]};
 cosab = Re[qa1[[i]]] * Re[qb1[[i]]] - r1.r2;
r0 = (Re[qa1[[i]]] Limit[Cross[s2, b1[[i]]], s2 → Sign[Re[qb1[[i]]]] b1[[i]]] +
      Re[qb1[[i]]] Limit[Cross[a1[[i]], s1], s1 → Sign[Re[qa1[[i]]]] a1[[i]]] -
      Cross[Limit[Cross[a1[[i]], s1], s1 → Sign[Re[qa1[[i]]]] a1[[i]]],
       Limit[Cross[s2, b1[[i]]], s2 → Sign[Re[qb1[[i]]]] b1[[i]]]]) /
   (Sin[ArcCos[a1[[i]].b1[[i]]]]);
 qpc = {cosab, r0[[1]], r0[[2]], r0[[3]]}.Qcoordinates2;
 pc[[i]] = FromQuaternion[qpc];
 ϕA = ArcTan[a1[[i]][[1]], a1[[i]][[2]]] / 50; ϕB = ArcTan[b1[[i]][[2]], b1[[i]][[1]]] / 50;
 If[ϕA * ϕB > 0, Θ = ArcCos[a1[[i]].b1[[i]]] * 180 / π, Θ = (2 π - ArcCos[a1[[i]].b1[[i]]]) * 180 / π];
 plotpc[[i]] = {Θ, Re[qpc]}, {i, m}]
```

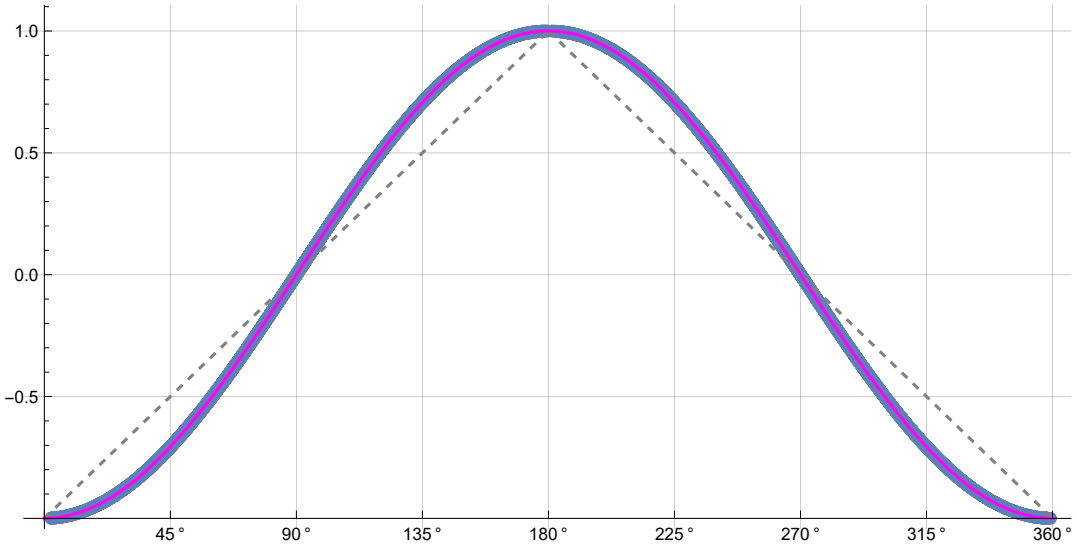
In[442]:=
```
AveA = N[Total[A] / m]; AveB = N[Total[B] / m];
Print["<A> = ", AveA]; Print["<B> = ", AveB];
meanpc = Mean[pc];
Print["Imaginary part vanishes, meanpc = ", meanpc]
sim = ListPlot[plotpc, PlotMarkers → {Automatic, Small},
   AspectRatio → 8 / 16, Ticks → {{{0, 0 °}, {45, 45 °}, {90, 90 °}, {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}];
negcos = Plot[-Cos[x Degree], {x, 0, 360}, PlotStyle → {Magenta}];
Show[sim, p1, p2, negcos]
```

```
<A> = 0.0104
```

```
<B> = 0.0042
```

```
Imaginary part vanishes, meanpc = -0.00105811 + 0. i
```

Out[450]=



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