Simulation for "Local Quantum Mechanical Prediction of the Singlet State"
http://dx.doi.org/10.13140/RG.2.2.22142.25927
Validation of the Local QM Product Calculation Prediction Using Pauli Matrices and Quaternions with 3D Vectors. Based on Joy Christian 's 3-Sphere Model.
Created by Fred Diether July, 2022

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

In[185]:=
```
<< Quaternions`;
m = 50000;
s1 = ConstantArray[0, m];
s2 = ConstantArray[0, m];
σs1 = ConstantArray[0, m];
σs2 = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
ra1 = ConstantArray[0, m];
rb1 = ConstantArray[0, m];
qA = ConstantArray[0, m];
qB = ConstantArray[0, m];
A = ConstantArray[0, m];
B = ConstantArray[0, m];
pc = ConstantArray[0, m];
plotpc = Table[{0, 0}, m];
```

Generating Particle Data with Three Independent Do-Loops

In[201]:=
```
Do[s = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors*)
 s1[[k]] = s;  (*Spin vector to A*)
 s2[[k]] = -s;  (*Spin vector to B*)
 σs1[[k]] = PauliMatrix[1] * s[[1]] + PauliMatrix[2] * s[[2]] + PauliMatrix[3] * s[[3]];
 (*Particle spin to A*)
 σs2[[k]] = -(PauliMatrix[1] * s[[1]] + PauliMatrix[2] * s[[2]] + PauliMatrix[3] * s[[3]]), {k, m}]
(*Particle spin to B*)
```

In[202]:=
```
Do[a = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors*)
 a1[[k]] = a;
 σa = PauliMatrix[1] * a[[1]] + PauliMatrix[2] * a[[2]] + PauliMatrix[3] * a[[3]];
 cosas1 = Re[Extract[Flatten[1/2 ((1 0).σa.σs1[[k]].(1
                                                     0) + (0 1).σa.σs1[[k]].(0
                                                                             1))], 1]];
 (*Particle - Detector interaction*)
 ra = Cross[a, s1[[k]]];  (*Vector cross products*)
 ra1[[k]] = ra;
 qA[[k]] = ToQuaternion[{cosas1, ra[[1]], ra[[2]], ra[[3]]}.{1, i, J, K}];
 (*Convert to quaternion*)
 A[[k]] = Sign[a.s1[[k]]], {k, m}]
```

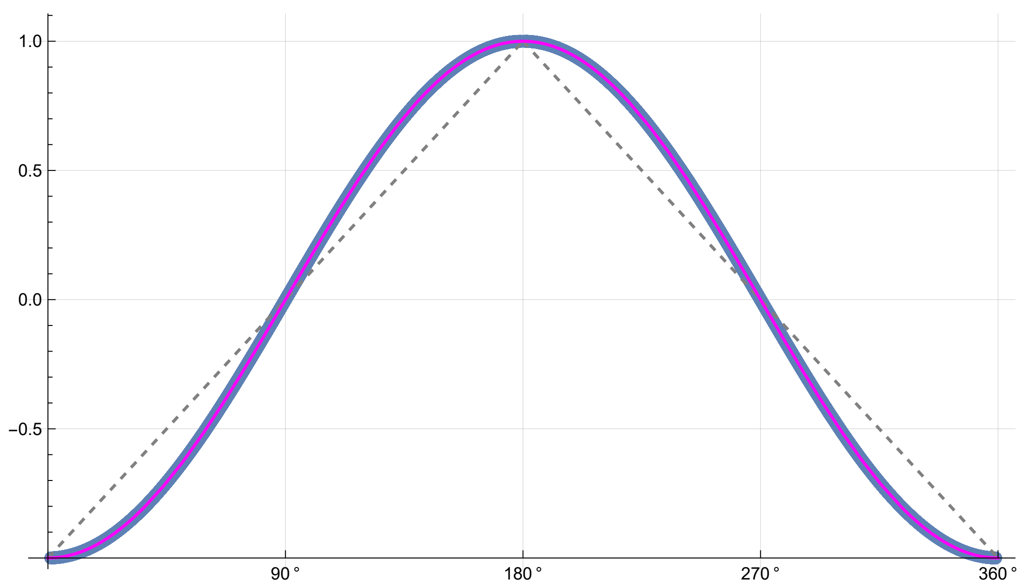
In[203]:=
```
Do[b = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors*)
 b1⟦k⟧ = b;
 σb = PauliMatrix[1] * b⟦1⟧ + PauliMatrix[2] * b⟦2⟧ + PauliMatrix[3] * b⟦3⟧;
 cosbs2 = Re[Extract[Flatten[1/2 ((1 0).σs2⟦k⟧.σb.(1
                                                  0) + (0 1).σs2⟦k⟧.σb.(0
                                                                         1))], 1]];
 (*Particle - Detector interaction*)
 rb = Cross[s2⟦k⟧, b];  (*Vector cross products*)
 rb1⟦k⟧ = rb;
 qB⟦k⟧ = ToQuaternion[{cosbs2, rb⟦1⟧, rb⟦2⟧, rb⟦3⟧}.{1, i, J, K}];
 (*Convert to quaternion*)
 B⟦k⟧ = Sign[b.s2⟦k⟧], {k, m}]
```

Verification of the Local QM Product Calculation Prediction

In[204]:=
```
Do[qpc = (Re[qA⟦k⟧] * Re[qB⟦k⟧] - ra1⟦k⟧.rb1⟦k⟧) +
    (Re[qA⟦k⟧] * Limit[Cross[s4, b1⟦k⟧], s4 → Sign[Re[qB⟦k⟧]] b1⟦k⟧] +
       Re[qB⟦k⟧] * Limit[Cross[a1⟦k⟧, s3], s3 → Sign[Re[qA⟦k⟧]] a1⟦k⟧] -
       Cross[Limit[Cross[a1⟦k⟧, s3], s3 → Sign[Re[qA⟦k⟧]] a1⟦k⟧],
        Limit[Cross[s4, b1⟦k⟧], s4 → Sign[Re[qB⟦k⟧]] b1⟦k⟧]]) /
     (Sin[ArcCos[a1⟦k⟧.b1⟦k⟧]]);  (*Product Calculation*)
 pc⟦k⟧ = qpc⟦1⟧;
 ϕa = ArcTan[a1⟦k⟧⟦1⟧, a1⟦k⟧⟦2⟧];
 ϕb = ArcTan[b1⟦k⟧⟦2⟧, b1⟦k⟧⟦1⟧];
 If[ϕa * ϕb > 0, angle = ArcCos[a1⟦k⟧.b1⟦k⟧] / Degree,
  angle = (2 π - ArcCos[a1⟦k⟧.b1⟦k⟧]) / Degree];
 plotpc⟦k⟧ = {angle, qpc⟦1⟧}, {k, m}]
```

In[205]:=
```
simulation = ListPlot[plotpc, PlotMarkers → {Automatic, Small}, AspectRatio → 9 / 16,
   Ticks → {{{90, 90 °}, {180, 180 °}, {0, 0 °}, {270, 270 °}, {360, 360 °}}, Automatic},
   GridLines → Automatic, AxesOrigin → {0, -1.0}];
negcos = Plot[-Cos[x Degree], {x, 0, 360}, PlotStyle → {Magenta}];
p1 = Plot[-1 + 2 x1 Degree / π, {x1, 0, 180}, PlotStyle → {Gray, Dashed}];
p2 = Plot[3 - 2 x2 Degree / π, {x2, 180, 360}, PlotStyle → {Gray, Dashed}];
Show[ simulation, p1, p2,  negcos]
```

Out[209]=



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

Computing Averages

In[210]:=
```
AveA = N[Total[A] / m];
AveB = N[Total[B] / m];
Print[" <A> = ", AveA, "  <B> = ", AveB];
meanpc = Mean[pc];
Print["Imaginary components vanish, meanpc = ", meanpc];
```

```
 <A> = 0.00596  <B> = -0.01224
Imaginary components vanish, meanpc = -0.00240793
```