Validation of the Local Product Calculation Prediction Using Quaternions with 3D Vectors, Based on Joy Christian ‘s 3-Sphere Model.
Created by Fred Diether Oct. 2023

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

In[61]:=
```
<< "clifford.m"
Qcoordinates = {i, j, k};
m = 30 000;
s1 = ConstantArray[0, m];
s2 = ConstantArray[0, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[0, m];
qA = ConstantArray[0, m];
qB = ConstantArray[0, m];
Aq = ConstantArray[0, m];
Bq = ConstantArray[0, m];
A = ConstantArray[0, m];
B = ConstantArray[0, m];
pc = ConstantArray[0, m];
plotpc = Table[{0, 0}, m];
I3 = Pseudoscalar[3];
```

Generating Particle Data with Three Independent Do-Loops

In[77]:=
```
Do[s = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors, Singlet Spin Vector*)
 s1[[h]] = s.Qcoordinates;  (*Spin quaternion to A*)
 s2[[h]] = -s.Qcoordinates,  (*Spin quaternion to B*)
 {h, m}]
```

In[78]:=
```
Do[a = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors*)
 aa = a.Qcoordinates;  (*Convert to quaternion*)
 a1[[h]] = a;
 qA[[h]] = QuaternionProduct[aa, s1[[h]]];
 Aq[[h]] = QuaternionProduct[s1[[h]], aa];
 A[[h]] = Sign[Re[qA[[h]]]], {h, m}]
```

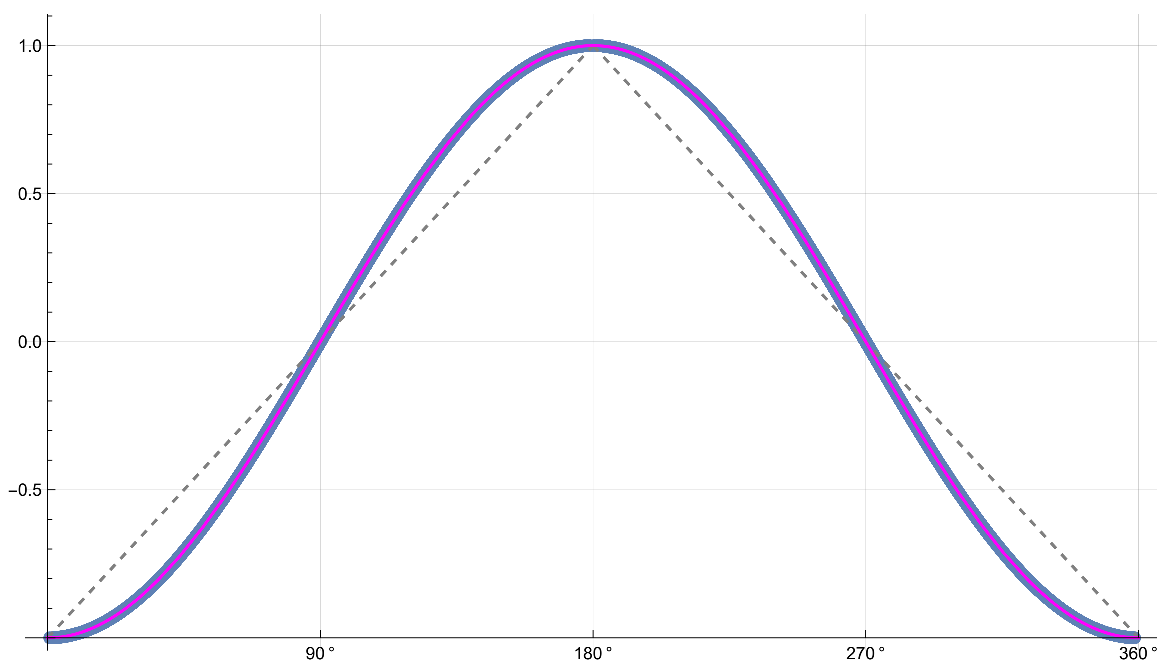
In[79]:=
```
Do[b = RandomPoint[Sphere[]];  (*Uniform Unit 3D Vectors*)
 bb = b.Qcoordinates;  (*Convert to quaternion*)
 b1[[h]] = b;
 qB[[h]] = QuaternionProduct[s2[[h]], bb];
 Bq[[h]] = QuaternionProduct[bb, s2[[h]]];
 B[[h]] = Sign[Re[qB[[h]]]], {h, m}]
```

Verification of the Local Product Calculation Prediction

In[80]:=
```
Do[rAB0 = Expand[
    ({e[1], e[2], e[3]}).(Re[qA[[h]]] * Limit[Cross[s4, b1[[h]]], s4 → Sign[Re[qB[[h]]]] b1[[h]]] +
        Re[qB[[h]]] * Limit[Cross[a1[[h]], s3], s3 → Sign[Re[qA[[h]]]] a1[[h]]] -
        Cross[Limit[Cross[a1[[h]], s3], s3 → Sign[Re[qA[[h]]]] a1[[h]]],
         Limit[Cross[s4, b1[[h]]], s4 → Sign[Re[qB[[h]]]] b1[[h]]]]) /
     (Sin[ArcCos[a1[[h]].b1[[h]]]])];
  LrAB0 = InnerProduct[I3, rAB0];
  rBA0 = Expand[
    ({e[1], e[2], e[3]}).(Re[Aq[[h]]] * Limit[Cross[b1[[h]], s4], s4 → Sign[Re[Bq[[h]]]] b1[[h]]] +
        Re[Bq[[h]]] * Limit[Cross[s3, a1[[h]]], s3 → Sign[Re[Aq[[h]]]] a1[[h]]] -
        Cross[Limit[Cross[b1[[h]], s4], s4 → Sign[Re[Bq[[h]]]] b1[[h]]],
         Limit[Cross[s3, a1[[h]]], s3 → Sign[Re[Aq[[h]]]] a1[[h]]]]) /
     (Sin[ArcCos[a1[[h]].b1[[h]]]])];
  LrBA0 = InnerProduct[I3, rBA0];
  qpc = 1/2 (Re[qA[[h]]] * Re[qB[[h]]] - Im[qA[[h]]].Im[qB[[h]]] + Re[Aq[[h]]] * Re[Bq[[h]]] -
      Im[Aq[[h]]].Im[Bq[[h]]] + LrAB0 + LrBA0);  (*Product Calculation*)
  pc[[h]] = qpc;
  ϕa = ArcTan[a1[[h]][[1]], a1[[h]][[2]]];
  ϕb = ArcTan[b1[[h]][[2]], b1[[h]][[1]]];
  If[ϕa * ϕb > 0, angle = ArcCos[a1[[h]].b1[[h]]] / Degree,
   angle = (2 π - ArcCos[a1[[h]].b1[[h]]]) / Degree];
  plotpc[[h]] = {angle, Re[qpc]}, {h, m}]
```

In[81]:=
```
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[85]=



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

Computing Averages

In[86]:=
```
AveA = N[Total[A] / m];
AveB = N[Total[B] / m];
Print[" <A> = ", AveA, "  <B> = ", AveB];
meanpc = Expand[Mean[pc]];
Print["Cross products vanish, meanpc = ", meanpc];
```

```
 <A> = -0.005  <B> = -0.00906667
```

```
Cross products vanish, meanpc = 0.00135924
```