Simulation Based on Michel Fodje's epr-simple simulation translated from Python to Mathematica by John Reed 13 Nov 2013
Modified by Fred Diether for Completely Local-Realistic July 2021
Some parts by Bill Nelson. Using Joy's 3-sphere GA model.

Set Run Time Parameters, Initialize Arrays and Tables

```
<< "clifford.m"
m = 200 000;
trialDeg = 720;
Ls1 = ConstantArray[0, m];
Ls2 = ConstantArray[0, m];
λ = ConstantArray[0, m];
λ2 = ConstantArray[0, m];
outA = Table[{0, 0, 0, 0, 0}, m];
outB = Table[{0, 0, 0, 0, 0}, m];
outA1 = Table[{0, 0, 0, 0, 0}, m];
outB1 = Table[{0, 0, 0, 0, 0}, m];
outA2 = Table[{0, 0, 0, 0, 0}, m];
outB2 = Table[{0, 0, 0, 0, 0}, m];
listAa1 = Table[{0, 0, 0, 0, 0}, m];
listBb1 = Table[{0, 0, 0, 0, 0}, m];
listAa2 = Table[{0, 0, 0, 0, 0}, m];
listBb2 = Table[{0, 0, 0, 0, 0}, m];
listAa6 = Table[{0, 0, 0, 0, 0}, m];
listBb6 = Table[{0, 0, 0, 0, 0}, m];
listAa7 = Table[{0, 0, 0, 0, 0}, m];
listBb7 = Table[{0, 0, 0, 0, 0}, m];
a1 = ConstantArray[0, m];
b1 = ConstantArray[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];
I1 = Pseudoscalar[3];
```

Generate Particle Data with 3 Do Loops

```
Do[s = RandomReal[{-179, 180}];  (*Singlet vector angle*)
 s1 = N[FromPolarCoordinates[{1, s * π / 180}]];
 s2 = ({e[1], e[2]}).s1;  (*Convert to GA*)
 λ2[[i]] = Sign[s];
 Ls1[[i]] = λ2[[i]] * InnerProduct[I1, s2];
 Ls2[[i]] = -Ls1[[i]];
 λ[[i]] = 0.25 (Cos[s/2]^2),  (*Hidden Variables*)
 {i, m}]
```

In[1513]:=
```
Do[a = RandomInteger[{-179, 180}]; (*Detector vector angle 1 degree increments*)
 a1 = N[FromPolarCoordinates[{1, a * π / 180}]];
 a2 = ({e[1], e[2]}).a1; (*Convert to GA*)
 Da = InnerProduct[I1, a2];
 IPa = InnerProduct[Da, Ls1[[i]]];
 GPa = GeometricProduct[Da, Ls1[[i]]];
 If[Abs[IPa] < λ[[i]], C1 = f1, C1 = g1];
 If[Abs[IPa] > λ[[i]], A = Sign[IPa],
  A = -λ2[[i]] * Sign[InnerProduct[Grade[GPa, 2], e[2], e[1]]]];
 AA = -λ2[[i]] * Sign[InnerProduct[Grade[GPa, 2], e[2], e[1]]];
 outA[[i]] = {a, A, i, C1, AA}, {i, m}]
```

In[1514]:=
```
Do[b = RandomInteger[{-179, 180}]; (*Detector vector angle 1 degree increments*)
 b1 = N[FromPolarCoordinates[{1, b * π / 180}]];
 b2 = ({e[1], e[2]}).b1; (*Convert to GA*)
 Db = InnerProduct[I1, b2];
 IPb = InnerProduct[Ls2[[i]], Db];
 GPb = GeometricProduct[Ls2[[i]], Db];
 If[Abs[ IPb] < λ[[i]], C2 = f2, C2 = g2];
 If[Abs[ IPb] > λ[[i]], B = Sign[ IPb],
  B = λ2[[i]] * Sign[InnerProduct[Grade[GPb, 2], e[2], e[1]]]];
 BB = λ2[[i]] * Sign[InnerProduct[Grade[GPb, 2], e[2], e[1]]];
 outB[[i]] = {b, B, i, C2, BB}, {i, m}]
```

Match Trial Numbers and do Statistical Analysis of Particle Data

In[1515]:=
```
outA1 = Select[outA, MemberQ[#, g1] &];
outA2 = Select[outA, MemberQ[#, f1] &];
outB1 = Select[outB, MemberQ[#, g2] &];
outB2 = Select[outB, MemberQ[#, f2] &];
listad = outA1[[All, 3]]; (*Match Trial Numbers*)
listbd = outB1[[All, 3]];
listAa1 = Select[outA1, Intersection[{#[[3]]}, listbd] == {#[[3]]} &];
listBb1 = Select[outB1, Intersection[{#[[3]]}, listad] == {#[[3]]} &];
listad2 = outA1[[All, 3]];
listad3 = listAa1[[All, 3]];
listAa3 = Select[outA1, Intersection[{#[[3]]}, listad3] != {#[[3]]} &];
listAa4 = Select[listAa1, Intersection[{#[[3]]}, listad2] != {#[[3]]} &];
listbd2 = outB1[[All, 3]];
listbd3 = listBb1[[All, 3]];
listBb3 = Select[outB1, Intersection[{#[[3]]}, listbd3] != {#[[3]]} &];
listBb4 = Select[listBb1, Intersection[{#[[3]]}, listbd2] != {#[[3]]} &];
M = Length[listAa3];
listAa7 = Table[{0, 0, 0, 0, 0}, M];
a2 = ConstantArray[0, M];
A2 = ConstantArray[0, M];
ind2 = ConstantArray[0, M];
A3 = ConstantArray[0, M];
A5 = ConstantArray[0, M];
A4 = ConstantArray[0, M];
A6 = ConstantArray[0, M];
a2 = listAa3[[All, 1]];
A2 = listAa3[[All, 2]];
ind2 = listAa3[[All, 3]];
```

```
A5 = listAa3[[All, 5]];
Do[A4 = A2[[i]]; A6 = A5[[i]];
 If[A4 == A6, A2 = A2, A2 = A5];
 listAa7[[i]] = {a2[[i]], A2[[i]], ind2[[i]], f1, A5[[i]]}, {i, M}]
M2 = Length[listBb3];
listBb7 = Table[{0, 0, 0, 0, 0}, M2];
b2 = ConstantArray[0, M2];
B2 = ConstantArray[0, M2];
ind3 = ConstantArray[0, M2];
B3 = ConstantArray[0, M2];
B5 = ConstantArray[0, M2];
B4 = ConstantArray[0, M2];
B6 = ConstantArray[0, M2];
b2 = listBb3[[All, 1]];
B2 = listBb3[[All, 2]];
ind3 = listBb3[[All, 3]];
B5 = listBb3[[All, 5]];
Do[B4 = B2[[i]]; B6 = B5[[i]];
 If[B4 == B6, B2 = B2, B2 = B5];
 listBb7[[i]] = {b2[[i]], B2[[i]], ind3[[i]], f1, B5[[i]]}, {i, M2}]
outA4 = Sort[Catenate[{outA2, listAa7}], #1[[3]] < #2[[3]] &];
outB4 = Sort[Catenate[{outB2, listBb7}], #1[[3]] < #2[[3]] &];
outA5 = Sort[Catenate[{listAa1, outA4}], #1[[3]] < #2[[3]] &];
outB5 = Sort[Catenate[{listBb1, outB4}], #1[[3]] < #2[[3]] &];
trials2 = Length[outA5];
theta = ConstantArray[0, m];
a1 = outA5[[All, 1]];
b1 = outB5[[All, 1]];
A1 = outA5[[All, 2]];
B1 = outB5[[All, 2]];
Do[θ = a1[[j]] - b1[[j]] + 360;  (*All angles are shifted by 2π since θ is an index*)
 theta[[j]] = θ;
 aliceD = A1[[j]]; bobD = B1[[j]];
 If[aliceD == 1, nAP[[θ]] ++];
 If[bobD == 1, nBP[[θ]] ++];
 If[aliceD == -1, nAN[[θ]] ++];
 If[bobD == -1, nBN[[θ]] ++];
 If[aliceD == 1 && bobD == 1, nPP[[θ]] ++];
 If[aliceD == 1 && bobD == -1, nPN[[θ]] ++];
 If[aliceD == -1 && bobD == 1, nNP[[θ]] ++];
 If[aliceD == -1 && bobD == -1, nNN[[θ]] ++], {j, trials2}]
```

Calculate Mean Values and Plot

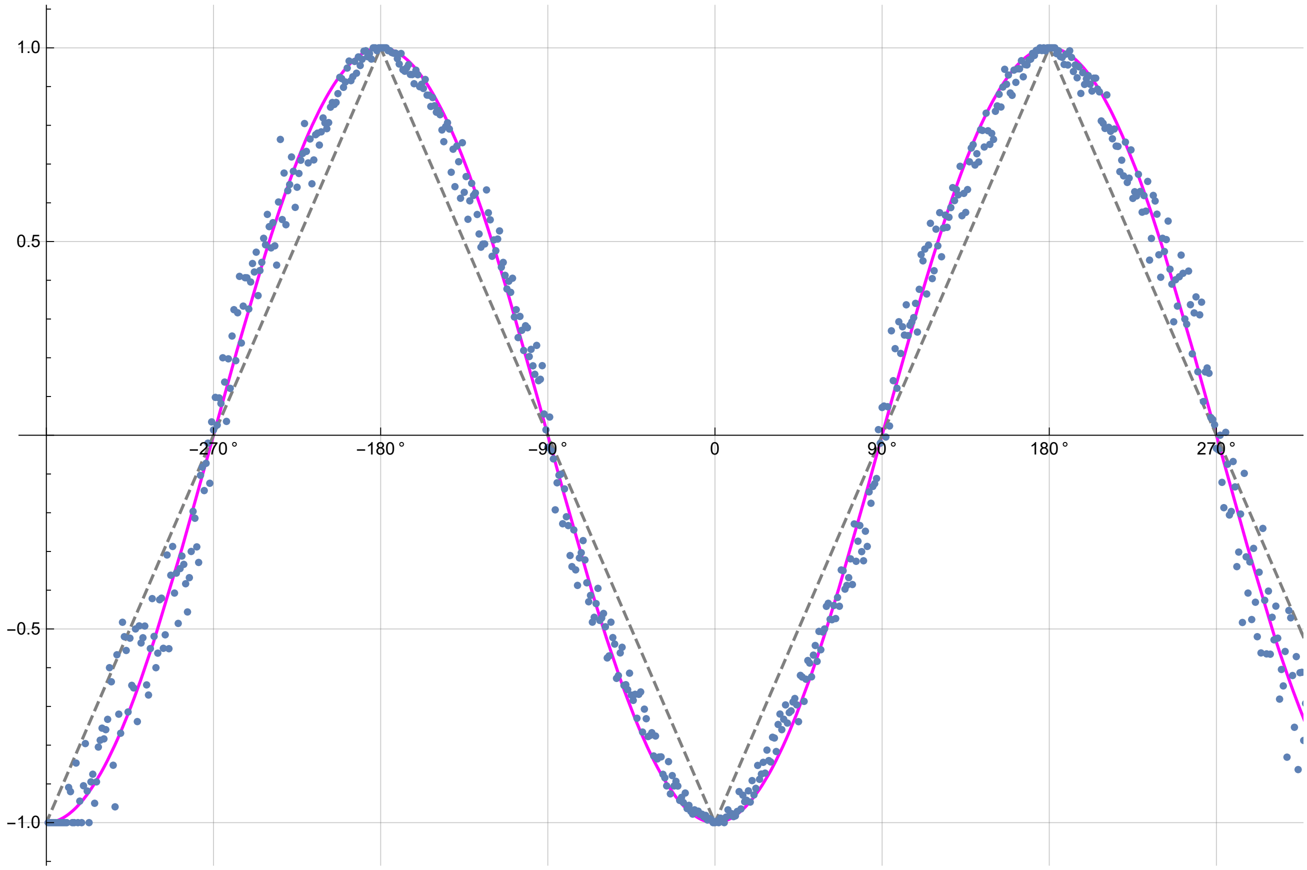
```
In[1567]:= pPP = 0; pPN = 0; pNP = 0; pNN = 0;
mean = ConstantArray[0, trialDeg];
Do[sum = nPP[[i]] + nPN[[i]] + nNP[[i]] + nNN[[i]];
If[sum == 0, Goto[jump],
   {pPP = nPP[[i]] / sum;
    pNP = nNP[[i]] / sum;
    pPN = nPN[[i]] / sum;
    pNN = nNN[[i]] / sum;
mean[[i]] = pPP + pNN - pPN - pNP}];
 Label[jump], {i, trialDeg}]
```

In[1570]:=
```
simulation = ListPlot[mean, PlotMarkers → {Automatic, Tiny}];
negcos = Plot[-Cos[x Degree], {x, 0, 720}, PlotStyle → {Magenta},
   Ticks → {{{0, -360 °}, {90, -270 °}, {180, -180 °}, {270, -90 °}, {360, 0 °}, {450, 90 °},
      {540, 180 °}, {630, 270 °}, {720, 360 °}}, Automatic}, GridLines → Automatic];
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}];
p3 = Plot[-5 + 2 x Degree / π, {x, 360, 540}, PlotStyle → {Gray, Dashed}];
p4 = Plot[7 - 2 x Degree / π, {x, 540, 720}, PlotStyle → {Gray, Dashed}];
```

Compare mean values with -Cosine Curve and compute averages

In[1576]:=
```
Show[negcos, p1, p2, p3, p4, simulation]
```

Out[1576]=



In[1577]:=
```
AveA = N[Sum[A1[[i]], {i, trials2}] / trials2];
AveB = N[Sum[B1[[i]], {i, trials2}] / trials2];
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}]
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]
CHSH = Abs[N[mean[[315]]] - N[mean[[225]]] + N[mean[[405]]] + N[mean[[45]]]]
```

```
AveA = 0.00332

AveB = 0.00146

P(A+)= 0.50166

P(B+)= 0.50073
```

Out[1589]= 200 000

Out[1590]= 0.252015

Out[1591]= 0.249625

Out[1592]= 0.249645

Out[1593]= 0.248715

Out[1594]= 2.56102

In[1595]:= `trials2`

Out[1595]= 200 000