Michel Fodje's epr-simple simulation translated from
Python to Mathematica by John Reed 13 Nov 2013
Modified for QM Local Complete States 30 Jul 2019
No Zeroes in A and B outputs

Set run time parameters, initialize arrays

In[163]:=
```
trialcs = 8 000 000;
trialDeg = 360;
```

In[165]:=
```
CS = ConstantArray[0, trialcs];
CS1 = Table[{0, 0, 0, 0}, trialcs];
CS2 = Table[{0, 0, 0, 0}, trialcs];
```

In[168]:=
```
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];
```

Complete States Selection

```
(*This section is redacted pending patent submission.
  Tell me what you want to do and I will send you a complete states table.*)
```

Generate Particle Data

In[178]:=
```
trials = Length[CS2];
aliceDeg = ConstantArray[0, trials];
bobDeg = ConstantArray[0, trials];
a1 = ConstantArray[0, trials];
b1 = ConstantArray[0, trials];
ss = ConstantArray[0, trials];
aa = ConstantArray[0, trials];
bb = ConstantArray[0, trials];
ss3 = ConstantArray[0, trials];
aliceDet = ConstantArray[0, trials];
bobDet = ConstantArray[0, trials];
aa = CS2[[All, 1]]; bb = CS2[[All, 2]]; ss3 = CS2[[All, 3]];
Do[a = aa[[j]]; b = bb[[j]]; s3 = ss3[[j]];
 x1 = Part[a, 1]; y1 = Part[a, 2];
 aliceDeg[[j]] = ArcTan[x1, x2];
 a1[[j]] = a;
 x2 = Part[b, 1]; y2 = Part[b, 2];
 bobDeg[[j]] = ArcTan[x2, y2];
 b1[[j]] = b;
 λ = RandomChoice[{-1, 1}];
 If[a.s3 > 0, s1 = a, s1 = -a]; (*Polarizer Functions*)
 If[b.s3 > 0, s2 = b, s2 = -b];
 (*Measurement Functions*)
 A = λ/2 ((1 0).(PauliMatrix[1] * a[[1]] + PauliMatrix[2] * a[[2]] + PauliMatrix[3] * a[[3]]).
     (PauliMatrix[1] * (-s1[[1]]) + PauliMatrix[2] * (-s1[[2]]) +
       PauliMatrix[3] * (-s1[[3]])).(1/0) +
    (0 1).(PauliMatrix[1] * a[[1]] + PauliMatrix[2] * a[[2]] + PauliMatrix[3] * a[[3]]).
     (PauliMatrix[1] * (-s1[[1]]) + PauliMatrix[2] * (-s1[[2]]) +
       PauliMatrix[3] * (-s1[[3]])).(0/1));

 aliceDet[[j]] = Extract[Flatten[Re[A]], 1];
 B = λ/2 ((1 0).(PauliMatrix[1] * s2[[1]] + PauliMatrix[2] * s2[[2]] + PauliMatrix[3] * s2[[3]]).
     (PauliMatrix[1] * b[[1]] + PauliMatrix[2] * b[[2]] + PauliMatrix[3] * b[[3]]).(1/0) +
    (0 1).(PauliMatrix[1] * s2[[1]] + PauliMatrix[2] * s2[[2]] + PauliMatrix[3] * s2[[3]]).
     (PauliMatrix[1] * b[[1]] + PauliMatrix[2] * b[[2]] + PauliMatrix[3] * b[[3]]).(0/1));
 bobDet[[j]] = Extract[Flatten[Re[B]], 1],
 {j, trials}]
```

Statistical Analysis of Particle Data

In[191]:=
```
Do[
 If[(aliceDeg[[j]] * bobDeg[[j]]) > 0, theta = ArcCos[a1[[j]].b1[[j]]] * 180 / π,
  theta = -ArcCos[a1[[j]].b1[[j]]] * 180 / π + 360];
 θ = Round[theta];
 aliceD = aliceDet[[j]]; bobD = bobDet[[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, trials}]
```

Calculate mean values and plot

In[192]:=
```
pPP = 0; pPN = 0; pNP = 0; pNN = 0;
```

In[193]:=
```
mean = ConstantArray[0, trialDeg];
```

In[194]:=
```
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}]
```

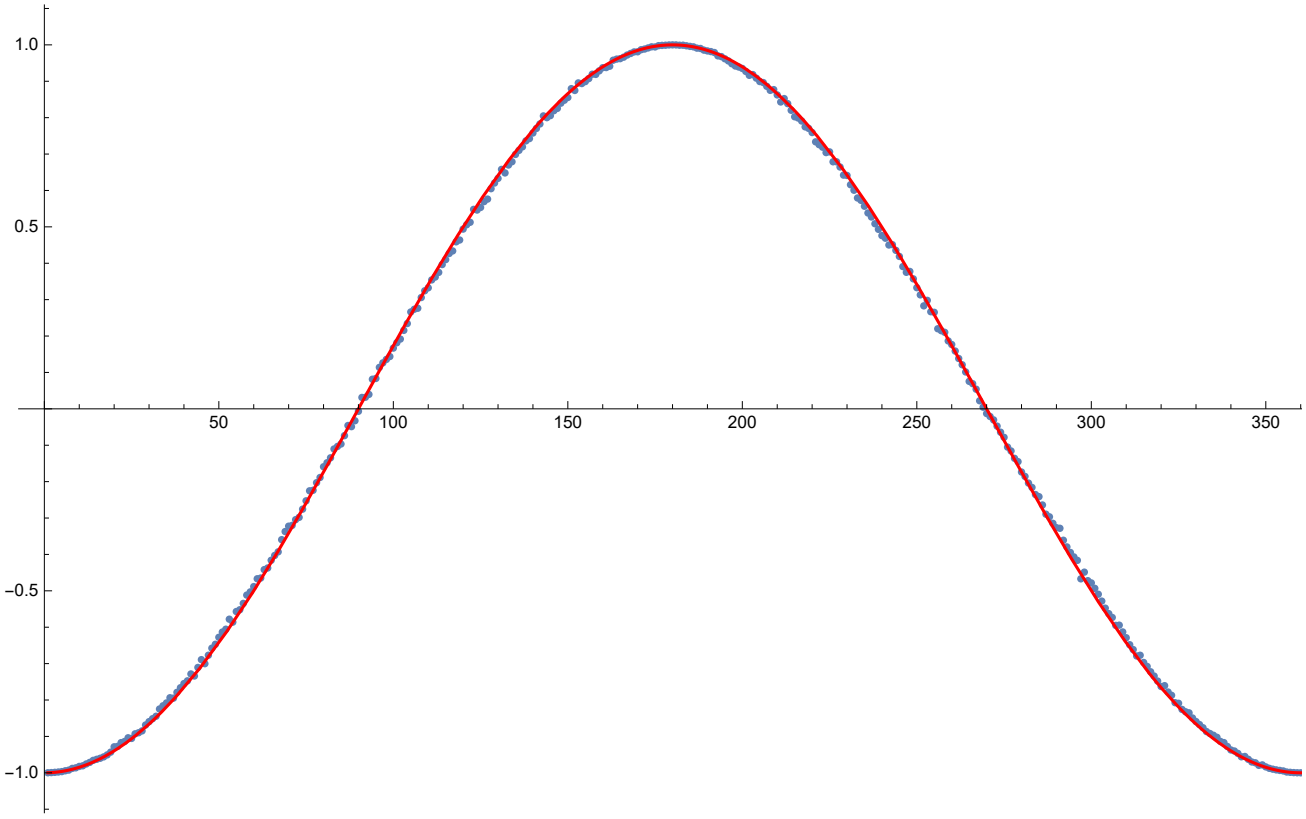
In[195]:=
```
simulation = ListPlot[mean, PlotMarkers → {Automatic, Tiny}];
```

In[196]:=
```
negcos = Plot[-Cos[x Degree], {x, 1, 361}, PlotStyle → {Red}];
```

Compare mean values with -Cosine Curve and compute averages

In[197]:=
```
Show[simulation, negcos]
AveA = N[Sum[aliceDet[[i]], {i, trials}] / trials];
AveB = N[Sum[bobDet[[i]], {i, trials}] / trials];
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]
```

Out[197]=




```
AveA = -0.000772142
AveB = -0.000594352
P(A+)= 0.499603
P(B+)= 0.499713
```

Out[211]= 0.250003

Out[212]= 0.250687

Out[213]= 0.2496

Out[214]= 0.24971