A Fundamental Misunderstanding

© 6/1/2018 Declan Traill B.Sc.

declan@netspace.net.au

Quantum Mechanics claims that particles can become entangled such that there is a correlation in the detected results from EPR type experiments that cannot be explained by Classical Physics. This paper shows that the result can be fully explained by Classical Physics, and that the correlation curve for different angles between the two detectors can by reproduced when modeled this way. The model can even explain the results of the most recent supposed loophole-free Quantum Steering experiments – giving a clear violation of the Steering Inequality.

Introduction

Most of Physics can be understood and makes sense in terms of Classical Physics that people can understand and is what is termed “Local and Real”. We can ‘see’ what is going on and can build a model of it.

Quantum Mechanics, on the other hand, has some strange counterintuitive aspects, with entanglement being the main one that appears to be “spooky action at a distance” as Einstein famously called it. It asserts that particles can cooperate across vast distances instantaneously as if by magic.

Experiments (usually termed EPR experiments, based on the original thought experiment devised by Einstein, Podolsky and Rosen) have been done that appear to confirm that the QM entanglement is actually occurring; however, due to the practical difficulties in performing the experiment there are possible ways that the correlation found in the experiment can be explained Classically, these are called ‘loopholes’. Thus, entanglement may not be occurring at all.

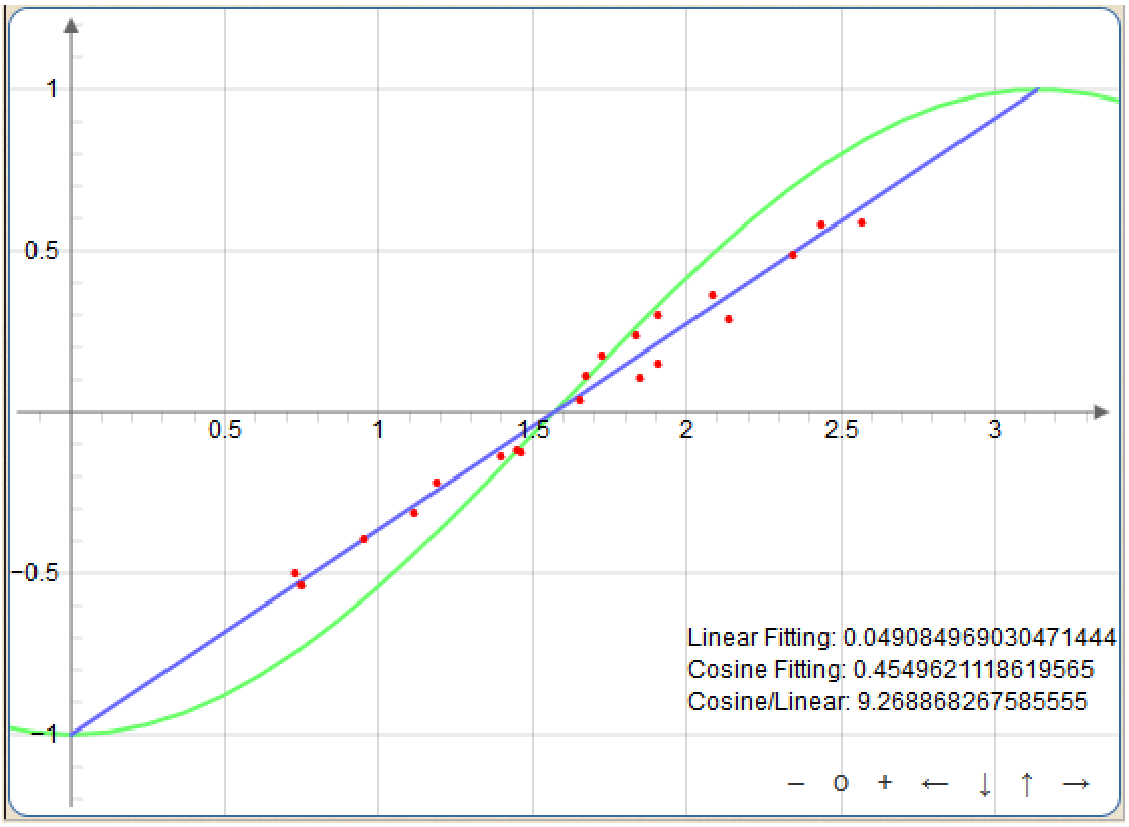
In order to solve this situation, recently supposed ‘loophole-free experiments have been devised in order to settle the answer once and for all. In these experiments, one of the loopholes termed the detection loophole is attempted to be closed by using a Steering Inequality rather than the usual Bell Test or CHSH Inequality that we’re used to determine if entanglement is occurring in the original EPR experiments. The Steering Inequality includes non-detects in the statistical calculations so that detector inefficiency cannot be used as a means to explain the experimental result Classically.

My work shows that this loophole has not been closed, and that even when the Steering Inequality is used, the experimental results can be explained Classically. As a result, the main problem with modeling the world in a ‘Local and Real’ way is overcome, and entanglement is shown not to be occurring. This may then allow the other aspects of QM to be unified with Classical Physics and Relativity.

The following URL is for a website (Ref 1) that allows for different models to be evaluated (using JavaScript) to try and match the results as measured and explained by Quantum Mechanics:

http://fmoldove.blogspot.com.au/2013_08_01_archive.html

The following screenshot is from that website, showing the Classical prediction in Blue, and the Quantum Mechanical result in Green. The goal is to try and match the QM (Green) line.



Linear Fitting: 0.049084969030471444
Cosine Fitting: 0.4549621118619565
Cosine/Linear: 9.268868267585555

Figure 1

The Evidence

We can fully explain the QM result using only Classical Physics. The plot below, Figure 2, is generated as a result of non-detect events by either Alice or Bob (the two detectors in the experiment). There is a high probability that Alice or Bob fail to detect either a +1 or -1 result at their detectors for photons incident on their detector at angles at or near 90 degrees from the detector's polarization axis. The probability of a non-detect event is proportional to the square of the cosine of the angle between the photon polarization axis and the detector polarization axis – such that the greater the angle difference, the more likely a non-detect will occur. This dependence on angle changes the shape of the correlation curve from linear (predicted to be the case for a Classical interpretation) to that predicted by Quantum Mechanics (QM). As you can see, the plotted curve is an exact match of the QM (Green) curve in Figure 1.

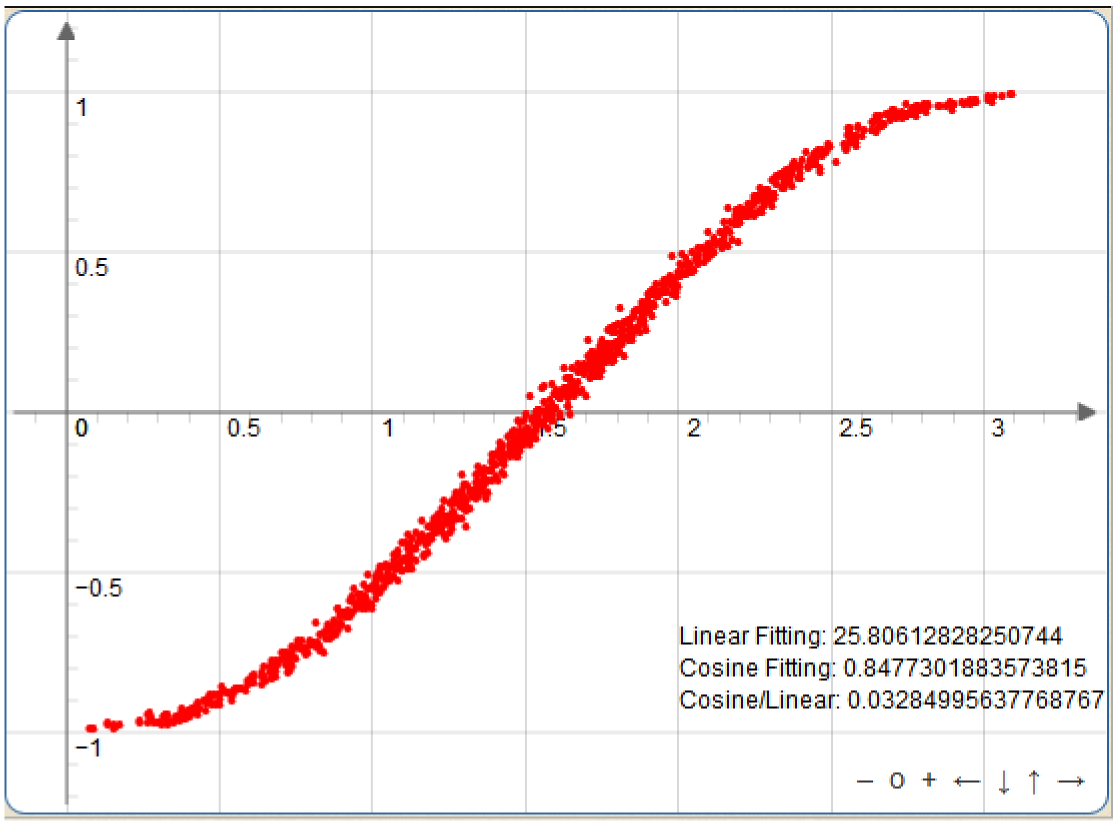


Figure 2

In normal EPR experiments (i.e. not the latest loophole-free experiments), failure of either Alice or Bob to detect a +1 or -1 result would cause that event to be discarded as there is not a complete set of results to record. This type of bias is referred to as the detection loophole, and has been a recognized weakness of EPR type experiments done in the past; so recently a new class of supposed loophole-free experiments were devised. The aim of these new experiments was to close off possible Classical explanations of the experimental results and prove that Quantum Entanglement was definitely occurring. The new experiments use what is termed a Steering Inequality that regards Bob as a trusted source (his non-detects are not recorded) and Alice, the untrusted partner, whose non-detect events are recorded and form part of the statistical calculations on the experimental results. This inclusion of non-detects on Alice's part is supposed to close the detection loophole so that detector efficiency cannot be used as a means to explain the correlation Classically.

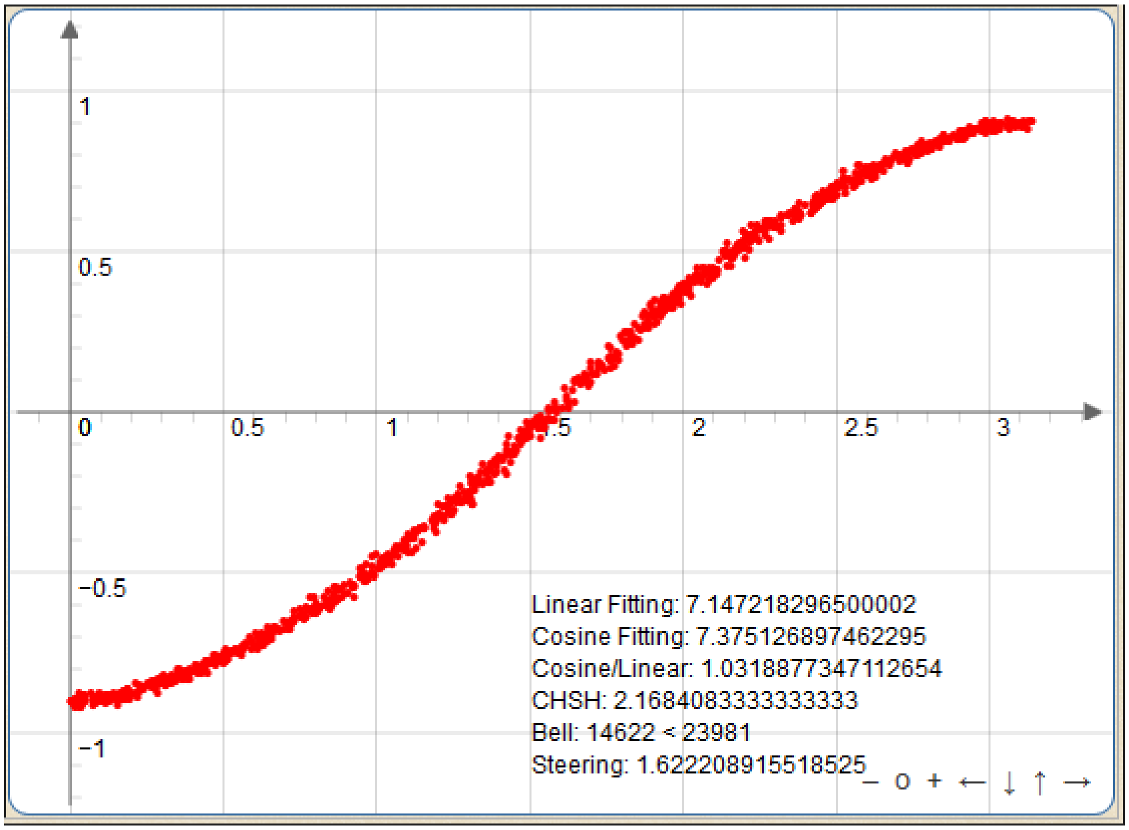
However, if the experiment is modeled based on a Steering Inequality calculation (See Appendix A; such as the calculation done in Ref 2) and the following functions for Alice and Bob (shown below) are used to determine their results (each can return -1, +1 or 0), then the new Steering Inequality can be shown to be violated, again just using a Classical model.

```
function GenerateAliceOutputFromSharedRandomness(direction, sharedRandomness3DVector)
{
        var dot = Dot(direction, sharedRandomness3DVector);
        var random_number = Math.random()*0.5;      // a random number between 0 and 0.5

        if (random_number > Math.pow(dot, 2)) {
                return 0;
        }

        // Now return +1 or -1 based on which Hemisphere the detector
        // is in relative to the photon's polarization axis
        if (dot > 0)
                return +1;
        else
                return -1;
};

function GenerateBobOutputFromSharedRandomness(direction, sharedRandomness3DVector)
{
        var dot = Dot(direction, sharedRandomness3DVector);
        var random_number = Math.random()*0.5;      // a random number between 0 and 0.5

        if (random_number > Math.pow(dot, 2)) {
                return 0;
        }

        // Now return +1 or -1 based on which Hemisphere the detector
        // is in relative to the photon's polarization axis
        if (dot > 0)
                return -1;
        else
                return +1;
};
```

Note:

1. Dot is the dot product – i.e. the projection of one vector onto the other, or the cosine of the angle between them.
2. Pow is a power function. So, a power of 2 of the dot value is the cosine squared.

Here is the correlation curve from the Steering Inequality model:



Linear Fitting: 7.147218296500002
Cosine Fitting: 7.375126897462295
Cosine/Linear: 1.0318877347112654
CHSH: 2.1684083333333333
Bell: 14622 < 23981
Steering: 1.622208915518525

Figure 3

As you can see, the QM correlation curve is again obtained, and the Steering Inequality value is violated (it is supposed to be less than or equal to 1 in a Classical model). You can also see that the other, more traditional, correlation inequality tests: The original Bell Test (Ref 6), and the CHSH Test (Ref 7) are also both violated.

This clearly demonstrates that the supposed QM Entanglement correlation can be explained Classically, even taking into account the recent loophole-free experiments.

Conclusion

The consequences of this are profound. Not only does it call into question Quantum Mechanics and its interpretation of the world, but it may have real world security implications, as industry relies on – and has been informed as such – that these QM communications are secure pathways to exchange encryption keys as they rely on entanglement! So, these results strongly suggest that the prediction of Quantum Mechanics, that particles can become entangled, may well be a Fundamental Misunderstanding about how the Universe works at its most basic, lowest level.

Appendix A

```
//Dot is the scalar product of 2 3D vectors
function Dot(a, b)
{
 return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
};

//Norm computes the norm of a 3D vector
function GetNorm(vect)
{
 return Math.sqrt(Dot(vect, vect));
};

//Normalize generates a unit vector out of a vector
function Normalize(vect)
{
 //declares the variable
 var ret = new Array(3);

 //computes the norm
 var norm = GetNorm(vect);

 //scales the vector
 ret[0] = vect[0]/norm;
 ret[1] = vect[1]/norm;
 ret[2] = vect[2]/norm;
 return ret;
};

//RandomDirection create a 3D unit vector of random direction
function RandomDirection()
{
 //declares the variable
 var ret = new Array(3);

 //fills a 3D cube with coordinates from -1 to 1 on each direction
 ret[0] = 2*(Math.random()-0.5);
 ret[1] = 2*(Math.random()-0.5);
 ret[2] = 0;//2*(Math.random()-0.5);
 //excludes the points outside of a unit sphere (tries again)
 if(GetNorm(ret) > 1)
  return RandomDirection();

 return Normalize(ret);
};

var bob_0_sum = 0;
var bob_neg_sum = 0;
var bob_pos_sum = 0;
var bob_total = 0;
var N1 = 0;
var N2 = 0;
var N3 = 0;
var N4 = 0;
var NE1 = 0;
var NE2 = 0;
var NE3 = 0;
var NE4 = 0;
var Bell_NU1 = 0;
var Bell_NE2 = 0;
var Bell_NU3 = 0;
var E1 = 0;
var E2 = 0;
var E3 = 0;
var E4 = 0;
var a_zero_b_pos = new Array(3);
var a_zero_b_neg = new Array(3);
var a_neg_b_pos = new Array(3);
var a_neg_b_neg = new Array(3);
var a_pos_b_pos = new Array(3);
var a_pos_b_neg = new Array(3);
var n_total = new Array(3);

var generateData = function()
{
 a_zero_b_pos[0] = 0;
 a_zero_b_neg[0] = 0;
 a_neg_b_pos[0] = 0;
 a_neg_b_neg[0] = 0;
 a_pos_b_pos[0] = 0;
 a_pos_b_neg[0] = 0;
```

```
a_zero_b_pos[1] = 0;
a_zero_b_neg[1] = 0;
a_neg_b_pos[1] = 0;
a_neg_b_neg[1] = 0;
a_pos_b_pos[1] = 0;
a_pos_b_neg[1] = 0;

a_zero_b_pos[2] = 0;
a_zero_b_neg[2] = 0;
a_neg_b_pos[2] = 0;
a_neg_b_neg[2] = 0;
a_pos_b_pos[2] = 0;
a_pos_b_neg[2] = 0;

n_total[0] = 0;
n_total[1] = 0;
n_total[2] = 0;

  clearBoard();
clearOutput();
//gets the data
var angMom = new Array();
var t = document.getElementById('in_data').value;
var data = t.split('\n');
for (var i=0;i<data.length;i++)
{
   var vect = data[i].split(',');
   if(vect.length == 3)
  angMom[i] = data[i].split(',');
  }

  var newTotAngMom = angMom.length;
  clearBoard();

var varianceLinear = 0;
var varianceCosine = 0;
var totTestDirs = document.getElementById('totTestDir').value;


var abDirections = new Array();
var AliceDirections = new Array();
var BobDirections = new Array();
var t2 = document.getElementById('in_test').value;
var data2 = t2.split('\n');
for (var k = 0; k < data2.length; k++)
{
        var vect2 = data2[k].split(',');
        if (vect2.length == 6)
 {
      abDirections[k] = data2[k].split(',');
      AliceDirections[k] = data2[k].split(',');
      BobDirections[k] = data2[k].split(',');

      AliceDirections[k][0] = abDirections[k][0];
      AliceDirections[k][1] = abDirections[k][1];
      AliceDirections[k][2] = abDirections[k][2];
      BobDirections[k][0]   = abDirections[k][3];
      BobDirections[k][1]   = abDirections[k][4];
      BobDirections[k][2]   = abDirections[k][5];
 }
}

var TempOutput = "";
var alice_result = 0;
var bob_result = 0;

//computes the output
for(var j=0; j<totTestDirs; j++)
{
 var a = AliceDirections[j];
 var b = BobDirections[j];
 var rand = 0;
 var a_dot;
 var b_dot;
 var a_angle;
 var b_angle;

 bob_0_sum = 0;
 bob_neg_sum = 0;
 bob_pos_sum = 0;
 bob_total = 0;

 for(var i=0; i<newTotAngMom; i++)
```

```
{
    angMom[i] = RandomDirection();
    rand = Math.random();

    alice_result = GenerateAliceOutputFromSharedRandomness(a, angMom[i], rand);
    bob_result = GenerateBobOutputFromSharedRandomness(b, angMom[i], rand);

            if (bob_result == 0) {
                    i--;
                    continue;
            }

            var dot_ab = Dot(a, b);
            var angle = Math.acos(dot_ab);

            if (angle >= 0 ) {
                        if ((angle > Math.PI*(3/4 - 1/200)) && (angle < Math.PI*(3/4 + 1/200))) {
                                    if (alice_result == 0) {
                                            if (bob_result == +1) a_zero_b_pos[0]++; else a_zero_b_neg[0]++;
                                    }

                                    if (alice_result == -1) {
                                            if (bob_result == +1) a_neg_b_pos[0]++; else a_neg_b_neg[0]++;
                                    }

                                    if (alice_result == +1) {
                                            if (bob_result == +1) a_pos_b_pos[0]++; else a_pos_b_neg[0]++;
                                    }

                                    n_total[0]++;
                        }
                        else if (angle > Math.PI*(1/2 - 1/100)) {
                                    if (alice_result == 0) {
                                            if (bob_result == +1) a_zero_b_pos[1]++; else a_zero_b_neg[1]++;
                                    }

                                    if (alice_result == -1) {
                                            if (bob_result == +1) a_neg_b_pos[1]++; else a_neg_b_neg[1]++;
                                    }

                                    if (alice_result == +1) {
                                            if (bob_result == +1) a_pos_b_pos[1]++; else a_pos_b_neg[1]++;
                        }

                        n_total[2]++;
                }
                else if ((angle > Math.PI*(1/4 - 1/200)) && (angle < Math.PI*(1/4 + 1/200))) {
                        if (alice_result == 0) {
                                if (bob_result == +1) a_zero_b_pos[2]++; else a_zero_b_neg[2]++;
                        }

                        if (alice_result == -1) {
                                if (bob_result == +1) a_neg_b_pos[2]++; else a_neg_b_neg[2]++;
                        }

                        if (alice_result == +1) {
                                if (bob_result == +1) a_pos_b_pos[2]++; else a_pos_b_neg[2]++;
                        }

                        n_total[1]++;
                }
        }

var angle = Math.acos(Dot(a, b));

        if (angle <= Math.PI/100) {
          N1++;
          if (alice_result == bob_result) {
                NE1++;
               }
        }

        if ( (angle >= (Math.PI/8 - Math.PI/100)) && (angle <= (Math.PI/8 + Math.PI/100) ) ){
          N2++;
          if (alice_result == bob_result) {
                NE2++;
               }
               else {
                Bell_NU1++;
               }
        }

        if ( (angle >= (Math.PI/4 - Math.PI/100)) && (angle <= (Math.PI/4 + Math.PI/100) ) ){
          N3++;
```

```
                 if (alice_result == bob_result) {
                        NE3++;
                        Bell_NE2++;
                       }
                }

                if ( (angle >= (3*Math.PI/8 - Math.PI/100)) && (angle <= (3*Math.PI/8 + Math.PI/100) ) ){
                 N4++;
                 if (alice_result == bob_result) {
                        NE4++;
                       }
                       else {
                        Bell_NU3++;
                       }
                }

                E1=2*NE1/N1 - 1;
                E2=2*NE2/N2 - 1;
                E3=2*NE3/N3 - 1;
                E4=2*NE4/N4 - 1;

         TempOutput = TempOutput + (j+1);
         TempOutput = TempOutput + ",";
         TempOutput = TempOutput + (i+1);
         TempOutput = TempOutput + ",";
         TempOutput = TempOutput +
         (alice_result);
         TempOutput = TempOutput + ",";
         TempOutput = TempOutput +
         (bob_result);
     if(i != newTotAngMom-1 || j != totTestDirs-1)
          TempOutput = TempOutput + " \n";
    }
   }

   apendResults(TempOutput);
  };

  var plotData = function()
  {
     clearBoard();
    boardCorrelations.suspendUpdate();
```

```
//gets the data
var angMom = new Array();
var t = document.getElementById('in_data').value;
var data = t.split('\n');
for (var i=0;i<data.length;i++)
{
   var vect = data[i].split(',');
   if(vect.length == 3)
  angMom[i] = data[i].split(',');
  }

  var newTotAngMom = angMom.length;

var varianceLinear = 0;
var varianceCosine = 0;
var totTestDirs = document.getElementById('totTestDir').value;

//extract directions
var abDirections = new Array();
var AliceDirections = new Array();
var BobDirections = new Array();
var t2 = document.getElementById('in_test').value;
var data2 = t2.split('\n');
for (var k = 0; k < data2.length; k++)
{
        var vect2 = data2[k].split(',');
        if (vect2.length == 6)
 {
            abDirections[k] = data2[k].split(',');
            AliceDirections[k] = data2[k].split(',');
  BobDirections[k] = data2[k].split(',');

      AliceDirections[k][0] = abDirections[k][0];
      AliceDirections[k][1] = abDirections[k][1];
      AliceDirections[k][2] = abDirections[k][2];
      BobDirections[k][0]   = abDirections[k][3];
      BobDirections[k][1]   = abDirections[k][4];
      BobDirections[k][2]   = abDirections[k][5];
 }
}
```

```
var tempLine = new Array();
var Data_Val = document.getElementById('out_measurements').value;
var data_rows = Data_Val.split('\n');

var directionIndex = 1;
var beginNewDirection = false;

     var a = new Array(3);
a[0] = AliceDirections[0][0];
a[1] = AliceDirections[0][1];
a[2] = AliceDirections[0][2];
      var b = new Array(3);
b[0] = BobDirections[0][0];
b[1] = BobDirections[0][1];
b[2] = BobDirections[0][2];
var sum = 0;

for (var ii=0;ii<data_rows.length;ii++)
{
 //parse the input line
   var vect = data_rows[ii].split(',');
   if(vect.length == 4)
     tempLine = data_rows[ii].split(',');

 //see if a new direction index is starting
 if (directionIndex != tempLine[0])
 {
  beginNewDirection = true;
 }

 if(!beginNewDirection)
 {
  var sharedRandomnessIndex = tempLine[1];
  var sharedRandomness = angMom[sharedRandomnessIndex];
  var aliceOutcome = tempLine[2];
  var bobOutcome = tempLine[3];
  sum = sum + aliceOutcome*bobOutcome;
 }


 if (beginNewDirection)
 {
  //finish computation
  var epsilon = sum/newTotAngMom;
  var angle = Math.acos(Dot(a, b));

  boardCorrelations.createElement('point', [angle,epsilon],{size:0.1,withLabel:false});

  var diffLinear = epsilon - (-1+2/Math.PI*angle);
  varianceLinear = varianceLinear + diffLinear*diffLinear;
  var diffCosine = epsilon + Math.cos(angle);
  varianceCosine = varianceCosine + diffCosine*diffCosine;

  //reset and start a new cycle
  directionIndex = tempLine[0];
  a[0] = AliceDirections[directionIndex-1][0];
  a[1] = AliceDirections[directionIndex-1][1];
  a[2] = AliceDirections[directionIndex-1][2];
  b[0] = BobDirections[directionIndex-1][0];
  b[1] = BobDirections[directionIndex-1][1];
  b[2] = BobDirections[directionIndex-1][2];
  sum = 0;
  var sharedRandomnessIndex = tempLine[1];
  var sharedRandomness = angMom[sharedRandomnessIndex];
  var aliceOutcome = tempLine[2];
  var bobOutcome = tempLine[3];
  sum = sum + aliceOutcome*bobOutcome;
  beginNewDirection = false;
 }

  }
//finish computation for last element of the loop above
var epsilon = sum/newTotAngMom;
var angle = Math.acos(Dot(a, b));

boardCorrelations.createElement('point', [angle,epsilon],{size:0.1,withLabel:false});

var diffLinear = epsilon - (-1+2/Math.PI*angle);
varianceLinear = varianceLinear + diffLinear*diffLinear;
var diffCosine = epsilon + Math.cos(angle);
varianceCosine = varianceCosine + diffCosine*diffCosine;

var S1=Math.abs(E1+E2+E3-E4);
var S2=Math.abs(E1+E2-E3+E4);
var S3=Math.abs(E1-E2+E3+E4);
var S4=Math.abs(-E1+E2+E3+E4);
var S=Math.max(S1,S2,S3,S4);

var total_a_zero = new Array(3);
var total_a_neg = new Array(3);
var total_a_pos = new Array(3);
var total_a = new Array(3);
var prob_a_zero = new Array(3);
var prob_a_neg = new Array(3);
var prob_a_pos = new Array(3);

total_a_zero[0] = a_zero_b_pos[0] + a_zero_b_neg[0];
total_a_neg[0] = a_neg_b_pos[0] + a_neg_b_neg[0];
total_a_pos[0] = a_pos_b_pos[0] + a_pos_b_neg[0];
total_a[0] = total_a_zero[0] + total_a_neg[0] + total_a_pos[0];
if ( total_a[0] == 0 ) total_a[0] = 1;
prob_a_zero[0] = total_a_zero[0] / total_a[0];
prob_a_neg[0] = total_a_neg[0] / total_a[0];
prob_a_pos[0] = total_a_pos[0] / total_a[0];

total_a_zero[1] = a_zero_b_pos[1] + a_zero_b_neg[1];
total_a_neg[1] = a_neg_b_pos[1] + a_neg_b_neg[1];
total_a_pos[1] = a_pos_b_pos[1] + a_pos_b_neg[1];
total_a[1] = total_a_zero[1] + total_a_neg[1] + total_a_pos[1];
if ( total_a[1] == 0 ) total_a[1] = 1;
prob_a_zero[1] = total_a_zero[1] / total_a[1];
prob_a_neg[1] = total_a_neg[1] / total_a[1];
prob_a_pos[1] = total_a_pos[1] / total_a[1];

total_a_zero[2] = a_zero_b_pos[2] + a_zero_b_neg[2];
total_a_neg[2] = a_neg_b_pos[2] + a_neg_b_neg[2];
total_a_pos[2] = a_pos_b_pos[2] + a_pos_b_neg[2];
total_a[2] = total_a_zero[2] + total_a_neg[2] + total_a_pos[2];
if ( total_a[2] == 0 ) total_a[2] = 1;
prob_a_zero[2] = total_a_zero[2] / total_a[2];
prob_a_neg[2] = total_a_neg[2] / total_a[2];
prob_a_pos[2] = total_a_pos[2] / total_a[2];

// Prevent divide by zero errors
if ( total_a_zero[0] == 0 ) total_a_zero[0] = 1;
if ( total_a_neg[0] == 0 ) total_a_neg[0] = 1;
```

```
 if ( total_a_pos[0] == 0 ) total_a_pos[0] = 1;

 if ( total_a_zero[1] == 0 ) total_a_zero[1] = 1;
 if ( total_a_neg[1] == 0 ) total_a_neg[1] = 1;
 if ( total_a_pos[1] == 0 ) total_a_pos[1] = 1;

 if ( total_a_zero[2] == 0 ) total_a_zero[2] = 1;
 if ( total_a_neg[2] == 0 ) total_a_neg[2] = 1;
 if ( total_a_pos[2] == 0 ) total_a_pos[2] = 1;

 var E = 0;

 E = E + prob_a_zero[0] * Math.pow((a_zero_b_pos[0] - a_zero_b_neg[0]) / total_a_zero[0], 2) +
             prob_a_neg[0] * Math.pow((a_neg_b_pos[0] - a_neg_b_neg[0]) / total_a_neg[0], 2) +
             prob_a_pos[0] * Math.pow((a_pos_b_pos[0] - a_pos_b_neg[0]) / total_a_pos[0], 2);

 E = E + prob_a_zero[1] * Math.pow((a_zero_b_pos[1] - a_zero_b_neg[1]) / total_a_zero[1], 2) +
                   prob_a_neg[1] * Math.pow((a_neg_b_pos[1] - a_neg_b_neg[1]) / total_a_neg[1], 2) +
                   prob_a_pos[1] * Math.pow((a_pos_b_pos[1] - a_pos_b_neg[1]) / total_a_pos[1], 2);

 E = E + prob_a_zero[2] * Math.pow((a_zero_b_pos[2] - a_zero_b_neg[2]) / total_a_zero[2], 2) +
                   prob_a_neg[2] * Math.pow((a_neg_b_pos[2] - a_neg_b_neg[2]) / total_a_neg[2], 2) +
                   prob_a_pos[2] * Math.pow((a_pos_b_pos[2] - a_pos_b_neg[2]) / total_a_pos[2], 2);

 //display total fit
 boardCorrelations.createElement('text',[1.5, -0.6, 'Linear Fitting: ' + varianceLinear],{});
 boardCorrelations.createElement('text',[1.5, -0.7, 'Cosine Fitting: ' + varianceCosine],{});
 boardCorrelations.createElement('text',[1.5, -0.8, 'Cosine/Linear: ' + varianceCosine/varianceLinear],{});
 boardCorrelations.createElement('text',[1.5, -0.9, 'CHSH: ' + S],{});
 boardCorrelations.createElement('text',[1.5, -1.0, 'Bell: ' + (Bell_NU3 + Bell_NE2) + ' < ' + Bell_NU1],{});
 boardCorrelations.createElement('text',[1.5, -1.1, 'Steering: ' + E],{});

 boardCorrelations.unsuspendUpdate();
};


var clearBoard = function()
{
 JXG.JSXGraph.freeBoard(boardCorrelations);
 boardCorrelations =
JXG.JSXGraph.initBoard('jxgboxCorrelations',{boundingbox:[-0.20, 1.25,
3.4, -1.25],axis:true,
showCopyright:false});
 boardCorrelations.create('functiongraph', [function(t){ return
-Math.cos(t); }, -Math.PI*10, Math.PI*10],{strokeColor:

"#66ff66", strokeWidth:2,highlightStrokeColor: "#66ff66",
highlightStrokeWidth:2});
 boardCorrelations.create('functiongraph', [function(t){ return
-1+2/Math.PI*t; }, 0, Math.PI],{strokeColor: "#6666ff",

strokeWidth:2,highlightStrokeColor: "#6666ff", highlightStrokeWidth:2});
};

var clearInput = function()
{
 document.getElementById('in_data').value = '';
};

var clearTestDir = function()
{
 document.getElementById('in_test').value = '';
};

var clearOutput = function()
{
 document.getElementById('out_measurements').value = '';
};

var generateTestDir = function()
{
   clearBoard();
 var totTestDir = document.getElementById('totTestDir').value;
 var testDir = new Array(totTestDir);
 var strData = "";
 for(var i=0; i<totTestDir; i++)
 {
  //first is Alice, second is Bob
  testDir[i] = RandomDirection();
  strData = strData + testDir[i][0] + ", " + testDir[i][1] + ", " +
testDir[i][2]+ ", " ;
  testDir[i] = RandomDirection();
  strData = strData + testDir[i][0] + ", " + testDir[i][1] + ", " +
testDir[i][2] + '\n';
 }

 document.getElementById('in_test').value = strData;
};

var generateRandomData = function()
{
   clearBoard();
 var totAngMoms = document.getElementById('totAngMom').value;
 var angMom = new Array(totAngMoms);
 var strData = "";
 for(var i=0; i<totAngMoms; i++)
 {
  angMom[i] = RandomDirection();
  strData = strData + angMom[i][0] + ", " + angMom[i][1] + ", " +
  angMom[i][2] + '\n';
 }

 document.getElementById('in_data').value = strData;
};

var apendResults= function(newData)
{
 var existingData = document.getElementById('out_measurements').value;
 existingData = existingData + newData;
 document.getElementById('out_measurements').value = existingData;
};

function GenerateAliceOutputFromSharedRandomness(direction, sharedRandomness3DVector, rand) {
            var dot = Dot(direction, sharedRandomness3DVector);
            var rand2 = Math.random()*0.5;

            if (rand2 > Math.pow(dot, 2)) {
                        return 0;
            }

            // Now return +1 or -1 based on which Hemisphere the detector is in relative to the chosen
electron spin axis
            if (dot > 0)
                        return +1;
            else
```

```
                    return -1;
};

function GenerateBobOutputFromSharedRandomness(direction, sharedRandomness3DVector, rand) {
          var dot = Dot(direction, sharedRandomness3DVector);
          var rand2 = Math.random()*0.5;

          if (rand2 > Math.pow(dot, 2)) {
                     return 0;
          }

          // Now return +1 or -1 based on which Hemisphere the detector is in relative to the chosen
electron spin axis
          if (dot > 0)
                    return -1;
          else
                    return +1;
};

var boardCorrelations = JXG.JSXGraph.initBoard('jxgboxCorrelations',
{axis:true, boundingbox: [-0.25, 1.25, 3.4, -1.25],
showCopyright:false});

clearBoard();
generateRandomData();
generateTestDir();
generateData();
plotData();
```

References

[1] Moldoveanu Florin. “Elliptic Composability”. 2013

http://fmoldove.blogspot.com.au/2013_08_01_archive.html

[2] Smith DH, Gillet G, de Almeida MP, et al. “Conclusive quantum steering with superconducting transition-edge sensors”. Nat Commun. 2012; p2-4.

https://www.nature.com/articles/ncomms1628.pdf

[3] Traill DA, “A Classical Explanation for the Correlation of Entangled Quantum Particles Via the Detection Loophole”, 2017. http://vixra.org/abs/1707.0402

[4] Jackson PA. FQXi finalist essays; “The Intelligent Bit”, 2013. Do Bob & Alice have a future?” 2014. “The Red/Green Sock Trick”
2015. http://fqxi.org/community/forum/topic/2430

and “Classical Quantum Consciousness” 2016.
http://fqxi.org/community/forum/topic/2755

[5] Minkowski JS, Jackson PA, “Quasi-Classical Entanglement, Superposition and Bell Inequalities”.

DOI: 10.13140/RG.2.1.3754.1287. 2014.
https://www.academia.edu/9216615/Quasi-classical_Entanglement_Superposition_and_Bell_Inequalities._v2

[6] Wikipedia, “Bell’s Theorem”. https://en.wikipedia.org/wiki/Bell%27s_theorem

[7] Wikipedia, “CHSH inequality”. https://en.wikipedia.org/wiki/CHSH_inequality

[8] Wikipedia, “Waveplate”. https://en.wikipedia.org/wiki/Waveplate

[9] Hou Z., Xiang G.Y., Dong D., Li C.F., Guo G.C. “Realization of mutually unbiased bases for a qubit with only one wave plate: theory and experiment”
Opt. Express, 23 (2015), pp. 10018-10031
https://www.osapublishing.org/oe/abstract.cfm?uri=oe-23-8-10018