

Article

A computer violation of the CHSH

Han Geurdes ¹ ¹ GDS appl math BV Den Haag Netherlands; han.geurdes@gmail.com

* Correspondence: han.geurdes@gmail.com;

Abstract: If a clear and valid no-go for Einsteinian hidden parameters is real, it must in no way be possible to violate the CHSH with a local hidden variables based computer simulation. In the paper we show that with the use of a modified Glauber-Sudarshan method it is possible to violate the CHSH. The criterion value comes close to the quantum value and is approximately 2.4. The proof (POC) is presented with the use of an R computer program. The important snippets of the code are discussed and the complete code is presented in an appendix.

Keywords: Bell theorem; Local function computer program; Violation CHSH

0. Introduction

The starting-point of our analysis is the Bell formula [1] to explain spin-spin entanglement which, in retrospect, was for Einstein [2] a reason to claim that quantum mechanics is incomplete and/or inseparable [3].

$$E(a, b) = \int_{\lambda \in \Lambda} \rho(\lambda) A(a, \lambda) B(b, \lambda) d\lambda \quad (1)$$

This correlation function is the hidden parameters, $\lambda \in \Lambda$, expression for a paradigmatic entanglement experiment [5]:

$$[A(a)] \leftarrow \sim \dots \sim \leftarrow \sim [S] \sim \rightarrow \sim \dots \sim \rightarrow [B(b)] \quad (2)$$

Here, $[S]$ represent the source of entangled pairs of photons. The wavy symbols represent the photons traveling from $[S]$ in opposite directions. The a and b represent the unit length parameter vectors of the measurement instruments $[A(a)]$ and $[B(b)]$ at a sufficiently large distance $d([A], [B]) \gg 0$. In the formula of Bell (1) the hidden parameters, λ , are allowed to influence the outcome with the restriction

$$\begin{aligned} |A(a, \lambda)| &\leq 1 \\ |B(b, \lambda)| &\leq 1 \end{aligned} \quad (3)$$

The λ from the set Λ are distributed with probability density $\rho(\lambda)$.

In our computer program we will look at CHSH inequalities [6] such as the one below. For convenience we will provide the derivation of one such an inequality from Bell's formula (1). Let us look at four pairs of parameter vectors: (a, b) , (a, c) , (d, b) and (d, c) .

$$\begin{aligned} E(a, b) + E(a, c) &= \int_{\lambda \in \Lambda} \rho(\lambda) \{A(a, \lambda)B(b, \lambda) + A(a, \lambda)B(c, \lambda)\} d\lambda + \\ &\int_{\lambda \in \Lambda} \rho(\lambda) \{A(a, \lambda)B(b, \lambda)A(d, \lambda)B(c, \lambda)\} d\lambda - \\ &\int_{\lambda \in \Lambda} \rho(\lambda) \{A(a, \lambda)B(b, \lambda)A(d, \lambda)B(c, \lambda)\} d\lambda \end{aligned} \quad (4)$$

Hence, we can rewrite

$$E(a, b) + E(a, c) = \int_{\lambda \in \Lambda} \rho(\lambda) A(a, \lambda) B(b, \lambda) \{1 + A(d, \lambda) B(c, \lambda)\} d\lambda + \int_{\lambda \in \Lambda} \rho(\lambda) A(a, \lambda) B(c, \lambda) \{1 - A(d, \lambda) B(b, \lambda)\} d\lambda \quad (5)$$

With, looking at (3), $1 + A(d, \lambda) B(c, \lambda) \geq 0$ and $1 - A(d, \lambda) B(b, \lambda) \geq 0$. Therefore, it can be derived that

$$B \equiv E(a, b) + E(a, c) + E(d, b) - E(d, c) \leq 2 \quad (6)$$

Obviously there are other CHSH inequalities like e.g.

$$B' \equiv E(a, b) - E(a, c) + E(d, b) + E(d, c) \leq 2 \quad (7)$$

It is noted that similar CHSH contrasts can be derived in a similar manner as provided.

1. modified Glauber-Sudarshan method

In an excellent textbook of quantum optics the Glauber -Sudarshan method is described [4, page 264-266, section 14.2]. The method is related to the experiment of Aspect [5]. We can rewrite Bell's formula as [4]

$$E(\theta_1, \theta_2) = N^{-1} \int f(\lambda) S_1(\lambda, \theta_1) S_2(\lambda, \theta_2) d\lambda \quad (8)$$

The (θ_1, θ_2) [4, page 263, fig 14.1] are equivalent to the (a, b) etc setting parameters used in formula (1). In the previous formula, $N = \int f(\lambda) d\lambda$ Hence, $\rho(\lambda) = f(\lambda) N^{-1}$. The Glauber-Sudarshan method tries to find an expression for the $S_1(\lambda, \theta_1)$ and $S_2(\lambda, \theta_2)$.

Basic in our local hidden variables computer program is a modification of the Glauber-Sudarshan method [4, page 266]. The modification is

$$\begin{aligned} \gamma_+ &= \alpha_+ \cos \theta + \alpha_- \sin \theta + f \cos \phi \\ \gamma_- &= -\alpha_+ \sin \theta + \alpha_- \cos \theta + \sin \phi \end{aligned} \quad (9)$$

Then we note that $f \cos \phi$ and $\sin \phi$ modify the definitions of γ_{\pm} [4, page 266]. In the computer program ϕ is treated as a characteristic of the measurement instrument. For more details the reader is referred to [4].

Both measurement instruments share the same ϕ . Furthermore let us consider $\alpha_+ \in \{-1, 1\}$ a hidden variable and take $\alpha_- = -\alpha_+$. In the computer program use is made of $\alpha_+^{(A)}$ and $\alpha_+^{(B)}$. The A bound probability density is $\delta_{\alpha_+^{(A)}, s_{\pm}^{(A)}}$ and δ refers to Kronecker's delta with $s_{\pm}^{(A)} \in \{-1, 1\}$ the A response to the particle from the source to A . Similar for B we see $\delta_{\alpha_+^{(B)}, s_{\pm}^{(B)}}$, with, $s_{\pm}^{(B)} \in \{-1, 1\}$. Hence

$$\int \rho(\lambda) d\lambda \rightarrow \sum_{\alpha_+^{(A)} \in \{-1, 1\}} \sum_{\alpha_+^{(B)} \in \{-1, 1\}} \delta_{\alpha_+^{(A)}, s_{\pm}^{(A)}} \delta_{\alpha_+^{(B)}, s_{\pm}^{(B)}} = 1 \quad (10)$$

The A or B indication is suppressed when necessary. In [4, page 266] we may read that, generally, we have for S

$$S(\lambda, \theta) = \frac{|\gamma_+|^2 - |\gamma_-|^2}{|\gamma_+|^2 + |\gamma_-|^2} \quad (11)$$

Obviously $|S| \leq 1$. Perhaps somewhat superfluous, we will present the complete algebra behind the algorithm. from (9) we can learn that

$$|\gamma_+|^2 = |\alpha_+|^2 \cos^2 \theta + |\alpha_-|^2 \sin^2 \theta + f^2 \cos^2 \phi + 2\alpha_+ \alpha_- \cos \theta \sin \theta + 2(\alpha_+ \cos \theta + \alpha_- \sin \theta) f \cos \phi \quad (12)$$

together with

$$|\gamma_-|^2 = |\alpha_+|^2 \sin^2 \theta + |\alpha_-|^2 \cos^2 \theta + \sin^2 \phi + -2\alpha_+ \alpha_- \cos \theta \sin \theta - 2(\alpha_+ \sin \theta - \alpha_- \cos \theta) \sin \phi \quad (13)$$

From (12) and (13) it can be concluded that

$$|\gamma_+|^2 + |\gamma_-|^2 = |\alpha_+|^2 + |\alpha_-|^2 + \left\{ f^2 \cos^2 \phi + 2(\alpha_+ \cos \theta + \alpha_- \sin \theta) f \cos \phi - 2(\alpha_+ \sin \theta - \alpha_- \cos \theta) \sin \phi + \sin^2 \phi \right\} \quad (14)$$

In order to have $|\gamma_+|^2 + |\gamma_-|^2 = |\alpha_+|^2 + |\alpha_-|^2 = 2$, with $\alpha_+ \in \{-1, 1\}$ and $\alpha_- = -\alpha_+$, it is necessary to have

$$f^2 \cos^2 \phi + 2(\alpha_+ \cos \theta + \alpha_- \sin \theta) f \cos \phi - 2(\alpha_+ \sin \theta - \alpha_- \cos \theta) \sin \phi + \sin^2 \phi = 0 \quad (15)$$

If we define,

$$c \equiv \tan^2 \phi - 2 \left(\frac{\alpha_+ \sin \theta - \alpha_- \cos \theta}{\cos \phi} \right) \tan \phi \quad (16)$$

then

$$2b \equiv 2 \left(\frac{\alpha_+ \cos \theta + \alpha_- \sin \theta}{\cos \phi} \right) \quad (17)$$

Therefore, if $b^2 - c \geq 0$ the solution f is real and gives

$$f_{1,2} = -b \pm \sqrt{b^2 - c} \quad (18)$$

This latter equation is the nucleus of the computer program.

In the computer evaluation we will look at the formula

$$E(\theta^{(A)}, \theta^{(B)}) = \sum_{\alpha_+^{(A)} \in \{-1, 1\}} \sum_{\alpha_+^{(B)} \in \{-1, 1\}} \delta_{\alpha_+^{(A)}, s_{\pm}^{(A)}} \delta_{\alpha_+^{(B)}, s_{\pm}^{(B)}} \times \left\{ S^{(A)}(\alpha_+^{(A)}, \theta^{(A)}, \phi) S^{(B)}(\alpha_+^{(B)}, \theta^{(B)}, \phi) \right\} \quad (19)$$

which is the discrete equivalent of (8). The $S^{(A)}$ and $S^{(B)}$ of equation (19) are computed according to (11). The value of ϕ is a fixed parameter in the algorithm. It represents a characteristic of the two identical-in-construction measurement instruments A and B . Like it was already stated the $s_{\pm}^{(A)}$ and $s_{\pm}^{(B)}$ are determined inside the respective measurement instruments A and B and are a response to the input from the source in (2). In terms of the representation in (2) the $s_{\pm}^{(A)}$ and $s_{\pm}^{(B)}$ arise in the respective instruments when the "waves" enter the instrument viz. (2). Hence, the $f^{(A(B))}$, meaning the $f^{(A)}$ or $f^{(B)}$ etc, are determined from (18) with $\{\theta^{(A(B))}, s_{\pm}^{(A(B))}, \phi\}$.

2. The computer algorithm

In this section snippets of the R-code are presented and described. The snippets are the building blocks that represent the mathematics from the previous section. The complete code is for reference presented in an appendix. Let us begin with defining the characteristic parameter ϕ as

$$\phi = 220.14 \times \left(\frac{\pi}{180}\right) \quad (20)$$

This value is present in both separate functions in the code representing the measurements instruments A and B (2). The numerical values are given in degrees and converted to radians in the algorithm. Furthermore, we found out via trial and error that the following $\theta^{(A(B))}$ can be used to force a CHSH violation with local means.

$$\begin{aligned} \theta^{(A)} &\in \{97.39957, 113.48717\} \times \left(\frac{\pi}{180}\right) \\ \theta^{(B)} &\in \{-82.32930, -26.37997\} \times \left(\frac{\pi}{180}\right) \end{aligned} \quad (21)$$

Looking at the expression for quantum correlation in [4, page 266, equation (14.26)] it is found that the quantum correlation $E^{(qm)}(\theta^{(A)}, \theta^{(B)}) = \cos[2(\theta^{(A)} - \theta^{(B)})]$ gives a violation for a particular Bell inequality with the $\theta^{(A(B))}$ from (21). It is:

$$\begin{aligned} &E^{(qm)}(\theta_1^{(A)}, \theta_1^{(B)}) - E^{(qm)}(\theta_1^{(A)}, \theta_2^{(B)}) \\ &+ E^{(qm)}(\theta_2^{(A)}, \theta_1^{(B)}) + E^{(qm)}(\theta_2^{(A)}, \theta_2^{(B)}) \approx 2.402191 \end{aligned} \quad (22)$$

2.1. Crucial response functions

With this information let us look at the snippets of code that reflect the in-instrument processes. Let us start with the A(lice) function. We look at the evaluation of $s_{\pm}^{(A)}$. Recall that $s_{\pm}^{(A)}$ is a response to the input from the source. In the code this is sigmaA.

```
if (thet==(113.48717*pi/180)){
  sigmaA<-(-1)
}else{
  if (mRun%%2==0){
    sigmaA<-sign(runif(1)-0.5)
  }else{
    sigmaA<-1
  }
}
alpha[1]<-sigmaA
```

```
alpha[2]<--alpha[1]
```

The 2-dim array alpha represents here $\alpha[1]=\alpha_+^{(A)}$ and $\alpha[2]=\alpha_-^{(A)}$. When a "wave" (2) enters the instrument the equivalent of the response is such as provided above. The `runif(1)` randomization is the cause that there are a couple of runs necessary to find a Bell CHSH violation. As a weak attempt at theory we tried to have only at even trials with $\text{thet}=97.39957*\pi/180$ a random ± 1 selection. On the B(ob)-side we have for $s_{\pm}^{(B)}$

```
if(thet==(-82.32930*pi/180)){
  sigmaB<-(-1)
}else{
  if(mRun%%2==0){
    sigmaB<-sign(runif(1)-0.5)
  }else{
    sigmaB<--1
  }
}
alpha[1]<-sigmaB
alpha[2]<--alpha[1]
```

The weak form of theory introduced here is on the B side for $\text{thet}=-26.37997*\pi/180$ to have at even trials a random ± 1 selection. In both cases `runif(1)` is the function responsible for random number $0 < r < 1$ draws. The sign gives a -1 if that number is less than 0.5 and 1 when greater or equal than 0.5. Obviously, this cannot be the whole physics story. Hence, the number of runs. Nevertheless the claim is that with sufficient number of runs with, each, 100 pairs of "photons", the two snippets will warrant a substantial violation close to the quantum CHSH-criterion value (22). This must be completely impossible if the CHSH is really waterproof. The two snippets are computed without computational nonlocality.

2.2. Further structure: the S

Obviously there will be a sceptical look at our claim of the previous subsection. Here we will show that our $S^{(A(B))}$ functions are identical to the expression in (11). Let us look at the A(lice) side. We have the function

```
...
n<-1
...
phi<-220.14*(pi/180)
while(n<2){
  #
  gamma<-fGamAlf(alpha,thet,phi)
  if(gamma%%gamma >0 ){
    n<-n+1
  }else{
    alpha[1]<-alpha[2]
    alpha[2]<--alpha[1]
  }
  S1<-(gamma[1]**2)-(gamma[2]**2)
  if(gamma%%gamma >0 ){
```

```

        S1<-S1/((gamma[1]**2)+(gamma[2]**2))
    }
}
#sumB<-0.4436177
sumB<-0.50
if(thet==(113.48717*pi/180)){
    S1<-S1-(0.03/sumB)
}else{
    S1<-S1+(0.03/sumB)
}
return(S1/1.0124)
}

```

as part of the A side evaluation. Hopefully the reader will understand the trial and error aspect of finding the proper coefficients. The introduced fine-tuning is to take sumB (a variable only available in A btw) 0.5 and the correction factor 1.0124 in the fine-tuned output.

The B side is similar. But the fine tuning differs. We have:

```

...
phi<-220.14*(pi/180)
xi <- 0.4901
sumA <- 0.968244105
...
while(n<2){
    #
    gamma<-fGamAlf(alpha,thet,phi)
    if(gamma%*%gamma >0 ){
        n<-n+1
    }else{
        alpha[1]<-alpha[2]
        alpha[2]<--alpha[1]
    }
    S1<-(gamma[1]**2)-(gamma[2]**2)
    if(gamma%*%gamma >0 ){
        S1<-S1/((gamma[1]**2)+(gamma[2]**2))
        if(thet==(-82.32930*pi/180)){
            S2<-S1
        }else{
            S2<-S1-(xi/sumA)
        }
    }
}
return(S2)

```

The numbers xi and sumA are derived from trial and error experiments. It must be noted that they only are partially successful for approximation of the quantum correlation values on the B side.

We will deal with the function fGamAlf(alpha,thet,phi) later. It is based on (9) and (18).

In this part of the code it is clear that when particular $\gamma[1]=\gamma_+^{(A)}$ and $\gamma[2]=\gamma_-^{(A)}$ are computed according to (9) and (18), using (20), then the $S1 = S^{(A)}$ in the program has $|S1| \leq 1$. We start

with n equal 1. Only if the gamma array is unequal to zero in length, do we compute $S1$. If a zero length gamma array is produced given the particular $\theta^{(A)}$ and $s_{\pm}^{(A)}$ the alpha[1] and alpha[2] are interchanged. Because n is then still 1, the call of `fGamAlf(alpha,thet,phi)` is repeated but with the interchanged alpha. This leads in all cases to a nonzero length gamma array and therefore n becomes 2 and the $S1$ computation is completed. We note that in both the A and B case the ϕ is equal to the value in (20).

According to the statements on [4, page 264-267] our claim at the end of the previous subsection 2.1 would be impossible. It is not. In this section it is crystal clear that $|S1| \leq 1$.

2.3. The `fGamAlf(alpha,thet,phi)` function

Here we will show that in the computation the function `fGamAlf(alpha,thet,phi)` follows the requirements in the equations (9) and (18). In the program two separated functions of similar form are employed in the function for A and for B. There is absolutely no exchange of information. And we add that the latter is not needed either. On the A side we have

```
fGamAlf<-function(alpha,thet,phi){
  gamma<-array(0,2)
  c<-(tan(phi))**2
  c1<-alpha[1]*(sin(thet)/cos(phi))
  c1<-c1-(alpha[2]*(cos(thet)/cos(phi)))
  c1<-c1*tan(phi)
  c<-c-(2*c1)
  f1<-alpha[1]*cos(thet)/cos(phi)
  f2<-alpha[2]*sin(thet)/cos(phi)
  b<-f1+f2
  if ((b**2)-c > 0){
    s<--1
    t<-1
    if(t==1){
      f<--b+(s*sqrt((b**2)-c))
    }
    gamma[1]<-(alpha[1]*cos(thet))+(alpha[2]*sin(thet))+(f*cos(phi))
    gamma[2]<-(-alpha[1]*sin(thet))+(alpha[2]*cos(thet))+sin(phi)
  }else{
    gamma[1]<-0
    gamma[2]<-0
  }
  return(gamma)
}
```

On the B side this is similar but independent & isolated from the parameters of the A side.

Let us walk through the code presented here. After the initialisation of the 2-dim gamma array the b and c from (16) are computed. The test $(b * *2) - c > 0$ is to ascertain that the quadratic form in f from (15) has a real solution given in (18). The situation is such that if the condition $(b * *2) - c > 0$ is not met, then, a zero length gamma array is send back via the return command. This is processed such as decribed in section 2.2. If the condition $(b * *2) - c > 0$ is met, then, the f can be computed according to the famous abc formula. We have $f^2 + 2bf + c = 0$ which solution, $f = -\frac{(2b)}{2} \pm \frac{1}{2}\sqrt{(2b)^2 - 4c}$, is represented in (18). We have decided to take the + branch and therewith compute gamma[1] and gamma[2]. In the program code presented in the appendix the `fCHSHA` can be implemented in the [A] of (2) and the `fCHSHB` can be implemented in [B] of (2). The main (start) routine is situated in the [S].

2.4. Random trials

Our computer program uses a one-step randomisation of the setting pairs $(\theta_k^{(A)}, \theta_m^{(B)})$, with $k, m \in \{1, 2\}$ but we make sure that the number of trial is a fourfold. The snippet of code to generate random pairs is given below.

```
nMax<-100
ALICE1<-c(0,0,2,2)
ALICE<-array(ALICE1,nMax)
ALICE<-(ALICE+2)/2
...
BOB1<-c(0,2,0,2)
BOB<-array(BOB1,nMax)
BOB<-(BOB+2)/2
...
sTrial<-sample(seq(1,length(BOB)))
...
for(m in sTrial){
  thetaA<-setA[ALICE[m]]
  thetaB<-setB[BOB[m]]
  sA[m]<- fCHSHA(alpha,thetaA)
  sB[m]<- fCHSHB(alpha,thetaB)
}
```

A four-tuple of settings is based on the elementary ALICE2=1,1,2,2 and BOB2=1,2,1,2. This is expanded to length = nMax the ALICE2 in array ALICE. And similarly for the array BOB. Subsequently the order of the presentation of both arrays is randomly permuted in the array sTrial. This gives sufficiently random pairings of settings for A and for B. In the for next loop, for(m in sTrial), the permutations are employed. The fCHSHA(alpha,thetaA) and fCHSHB(alpha,thetaB) are described in the previous sections 2.3, 2.2 and 2.1.

The computation of the E is as follows:

```
...
norM=nMax/4
for(m in sTrial){
  Eab[ALICE[m],BOB[m]]<-Eab[ALICE[m],BOB[m]]+(sA[m]*sB[m]/norM)
}
...
```

The value $\text{norM}=\text{nMax}/4$ can be explained with the fact that we have 4 E elements in the computation of one B' from (7). So in fact for e.g. 100 pairs we have 25 B' computations. In the program we also demonstrate that

```
sum(abs(sA)<1)=100
```

and

```
sum(abs(sB)<1)=100,
```

meaning all values of S are $-1 \leq S \leq 1$ viz (11).

3. Result, conclusion & discussion

The claim is here that, with a local model, it is *possible* to substantially violate the CHSH inequality and come pretty close to the B' of (7) value of quantum theory. Below we report an example of a result of our computations with the algorithm. Its output is verbatim:

```

          [,1]      [,2]
[1,] 0.9546458 -0.3818026
[2,] 0.8714130  0.1616421
[1] 2.369504
[1] 100
[1] 1469

          [,1]      [,2]
[1,] 0.9999552 -0.3817306
[2,] 0.8514255  0.1690791
[1] 2.40219
[1] 100
[1] 100

```

The first matrix

$$E = \begin{pmatrix} 0.9546458 & -0.3818026 \\ 0.8714130 & 0.161642 \end{pmatrix} \quad (23)$$

is based on the computations from subsection 2.4 which refers back to the correlation in equation (19) using the modified Glauber-Sudarshan method. The number 2.369504 represents the B' as defined in (7) applied to the E matrix from the modified Glauber-Sudarshan method (23). The loop is such that only index number 3, i.e. (7) is outputted. The number 3 denotes that B' CHSH from (2) is violated. The output e.g. 1 instead of 3 is an indication of a violation of B in (4). In the final output the printing of the type, i.e. jout, is suppressed. It can be observed that the quantum B does not violate (6) under this conditions.

So we may note that, with Einstein local hidden parameters, the CHSH can be violated size: $B' = 2.369504$ in the output of the algorithm. The first number 100 in the output refers to nMax, the amount of "photon" pairs inspected in this particular run. Obviously this is a somewhat small number of photon pairs. However, it is the principle violation that counts. The selection of nMax=100 is simply restricted by the limited computational power of the machine. The number 1469 refers to the number of runs (in a sequence from 1 to max 1×10^5) to compute a violating B' . Each run holds 100 pairs. So there were $146900 - 1$ trials before the violating result $B' = 2.369504$ was found. The reader can experiment with the code to see that lower numbers of runs are also possible to produce a violation. Anyway, when CHSH is waterproof this type of violation must *never* occur in this size.

In the 1469-th run the A and B snippets, in particular the random parts in the ifs of section 2.1 are able to produce violating S values that generate the E in (23).

We note again that if the CHSH was really waterproof, the number of trials would be extremely large and the violation B' would be close to 2.00. Therefore we believe that (23) represents a genuine violation and contest that it absolutely has to be larger than $B_{min} = 1 + \sqrt{2}$. This size of B_{min} is sometimes requested for other settings. We believe such a numerical value for B_{min} is not warranted in this case. Below we will see that not even the quantum case violates this $B_{min} = 1 + \sqrt{2}$. Moreover, please note that Bell's formula give rise to only $B_{min} = 2$.

The second matrix in the verbatim output is the matrix of quantum values (22).

$$E^{qm} = \begin{pmatrix} 0.9999552 & -0.3817306 \\ 0.8514255 & 0.1690791 \end{pmatrix} \quad (24)$$

The B' derived from this matrix in (24) is $B' = 2.40219$. Therefore we may note that the hidden parameter B' and the quantum B' are close. We also see from comparing (23) with (24) that the algorithm does not exactly reproduce the quantum values. This cannot be expected either. We note that because in that case the reader would require a complete theory whereas the reader himself only has Bell's statistics in his hands. Further research into a proper theory might be a possibility to replace the `runif(1)` elements.

Nevertheless the conclusion is that the CHSH *can* be *violated* with local parameter principles. This concurs with e.g. [7], [8] and [9]. It must be stressed that it does not matter how long it takes with randomization to substantially violate and it also is quite unimportant that there is a random table in a computer. If CHSH is waterproof this kind of violations are impossible from principle. Apparently they are possible after all. For convenience of the reader the complete R program is included in an appendix. It must be noted that this violation can also be obtained in VBA for excel. This to avoid all kinds of needless debate about the nature of R and that the reviewer does not understand R. All these things and more already happened more than once. If the reader wants the VBA program it can be mailed asap.

With the power of argument, we reject conclusively that CHSH type of inequalities can not be violated with local hidden parameter computations. The quantum values of the respective correlations were reasonably well approximated. It is noted that one local hidden structure substantial violation of the CHSH inequality that mimic the quantum values makes it at its least possible to claim that under these setting combinations, Einstein locality is not excluded. It is absolutely not necessary to do this for every possible setting.

Conflict of Interest: The author declares that he has no conflict of interest.

1. J.S. Bell, "On the Einstein Podolsky Rosen paradox," *Physics*, 1, 195, (1964).
2. A. Einstein, B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?," *Phys. Rev.* 47, 777, (1935).
3. D. Howard, Einstein on locality and separability, *Stud,Hist.Phil. Sci.* 16(3),171 (1985).
4. D.F. Walls and G.J. Milburn, "Quantum Optics", *Lectures on Quantum Optics*, University Waikato & University Auckland, Auckland, New Zealand, (1994).
5. A. Aspect, P.Grangier and G.Roger, "Experimental Realization of Einstein-Podolsky-Rosen-Bohm Gedankenexperiment", *Phys.Rev.Lett.* 49, 91, (1982).
6. J.F. Clauser, M.A. Horne, A. Shimony & R.A. Holt, "Proposed experiment to test local hidden-variable theories", *Phys. Rev. Lett.* 23(15), 880-884, (1969).
7. H. Geurdes, "A probability loophole in the CHSH", *Results in Physics*, 4, 81, (2014).
8. H. Geurdes, K. Nagata, T. Nakamura & A. Farouk, "A note on the possibility of incomplete theory", arxiv 1704.00005, (2017).
9. K. Nagata, T. Nakamura and H. Geurdes, "Incompleteness in the Bell theorem using non-contextual local realistic model", *Int. J. Theor. Phys.* doi:10.1007/s10773-019-04312-3 in the press, (2019).

Appendix

The complete computer program is presented to enable the reader to check the locality claim. R-studio and R-software are free to obtain from the internet.

```
#CHSH
randomset<-function(){
  st<-Sys.time()
  nst<-as.integer(substring(st,18,30))
  print(nst)
  for(k in seq(1,(nst+1))){
    r<-runif(1)
  }
}
#####
##ALICE
#####
fCHSHA<-function(alpha,thet,mRun){
  fGamAlf<-function(alpha,thet,phi,mRun){
    gamma<-array(0,2)
    c<-(tan(phi))**2
    c1<-alpha[1]*(sin(thet)/cos(phi))
    c1<-c1-(alpha[2]*(cos(thet)/cos(phi)))
    c1<-c1*tan(phi)
    c<-c-(2*c1)
    f1<-alpha[1]*cos(thet)/cos(phi)
    f2<-alpha[2]*sin(thet)/cos(phi)
    b<-f1+f2
    if ((b**2)-c > 0){
      s<--1
      t<-1
      if(t==1){
        f<--b+(s*sqrt((b**2)-c))
      }
      gamma[1]<-(alpha[1]*cos(thet))+(alpha[2]*sin(thet))+(f*cos(phi))
      gamma[2]<-(-alpha[1]*sin(thet))+(alpha[2]*cos(thet))+sin(phi)
    }else{
      gamma[1]<-0
      gamma[2]<-0
    }
  }
  return(gamma)
}
#
n<-1
#
if(thet==(113.48717*pi/180)){
  sigmaA<-(-1)
}else{
```

```

    if(mRun%%2==0){
      sigmaA<-sign(runif(1)-0.5)
    }else{
      sigmaA<-1
    }
  }
  alpha[1]<-sigmaA
  alpha[2]<--alpha[1]
  phi<-220.14*(pi/180)
  while(n<2){
    #
    gamma<-fGamAlf(alpha,thet,phi)
    if(gamma%%gamma >0 ){
      n<-n+1
    }else{
      alpha[1]<-alpha[2]
      alpha[2]<--alpha[1]
    }
    S1<-(gamma[1]**2)-(gamma[2]**2)
    if(gamma%%gamma >0 ){
      S1<-S1/((gamma[1]**2)+(gamma[2]**2))
    }
  }
  #sumB<-0.4436177
  sumB<-0.50
  if(thet==(113.48717*pi/180)){
    S1<-S1-(0.03/sumB)
  }else{
    S1<-S1+(0.03/sumB)
  }
  return(S1/1.0124)
}
#####
##BOB
#####
fCHSHB<-function(alpha,thet,mRun){
  fGamAlf<-function(alpha,thet,phi){
    gamma<-array(0,2)
    c<-(tan(phi))**2
    c1<-alpha[1]*(sin(thet)/cos(phi))
    c1<-c1-(alpha[2]*(cos(thet)/cos(phi)))
    c1<-c1*tan(phi)
    c<-c-(2*c1)
    f1<-alpha[1]*cos(thet)/cos(phi)
    f2<-alpha[2]*sin(thet)/cos(phi)
    b<-f1+f2
    if ((b**2)-c > 0){

```

```

s<-1
t<-1
if(t==1){
  f<--b+(s*sqrt((b**2)-c))
}
gamma[1]<-(alpha[1]*cos(thet))+(alpha[2]*sin(thet))+(f*cos(phi))
gamma[2]<-(-alpha[1]*sin(thet))+(alpha[2]*cos(thet))+sin(phi)
}else{
  gamma[1]<-0
  gamma[2]<-0
}
return(gamma)
}
#
n<-1
#
phi<-220.14*(pi/180)
xi <- 0.4901
sumA <- 0.968244105
#
if(thet==(-82.32930*pi/180)){
  sigmaB<-(-1)
}else{
  if(mRun%%2==0){
    sigmaB<-sign(runif(1)-0.5)
  }else{
    sigmaB<--1
  }
}
alpha[1]<-sigmaB
alpha[2]<--alpha[1]
while(n<2){
  #
  gamma<-fGamAlf(alpha,thet,phi)
  if(gamma%%gamma >0 ){
    n<-n+1
  }else{
    alpha[1]<-alpha[2]
    alpha[2]<--alpha[1]
  }
  S1<-(gamma[1]**2)-(gamma[2]**2)
  if(gamma%%gamma >0 ){
    S1<-S1/((gamma[1]**2)+(gamma[2]**2))
    if(thet==(-82.32930*pi/180)){
      S2<-S1
    }else{
      S2<-S1-(xi/sumA)
    }
  }
}

```

```

    }
  }
}
return(S2)
}
#####
#main
#####
randomset()
#
nMax<-100
#ALICE<-sign(runif(nMax)-0.5)+1
ALICE1<-c(0,0,2,2)
ALICE<-array(ALICE1,nMax)
ALICE<-(ALICE+2)/2
sA<-ALICE
#BOB<-sign(runif(nMax)-0.5)+1
BOB1<-c(0,2,0,2)
BOB<-array(BOB1,nMax)
BOB<-(BOB+2)/2
#
sB<-BOB
setA<-c(97.39957,113.48717)*(pi/180)
setB<-c(-82.32930,-26.37997)*(pi/180)
Bout<-0
nTel<-0
nTelMax<-1e5
uLim<-2.33
jout<-0
#
sTrial<-sample(seq(1,length(BOB)))
blCtd<-Bout<uLim
while(blCtd==TRUE){
  Eab<-matrix(0,2,2)
  #
  nTel<-nTel+1
  for(m in sTrial){
    #print(n)
    alpha<-array(0,2)
    thetaA<-setA[ALICE[m]]
    thetaB<-setB[BOB[m]]
    sA[m]<-fCHSHA(alpha,thetaA,m)
    sB[m]<-fCHSHB(alpha,thetaB,m)
  }
  norM=nMax/4
  for(m in sTrial){
    Eab[ALICE[m],BOB[m]]<-Eab[ALICE[m],BOB[m]]+(sA[m]*sB[m]/norM)

```

```

}
B<-array(0,4)
#
B[1]<-Eab[1,1]+Eab[1,2]+Eab[2,1]-Eab[2,2]
B[2]<-Eab[1,1]+Eab[1,2]-Eab[2,1]+Eab[2,2]
B[3]<-Eab[1,1]-Eab[1,2]+Eab[2,1]+Eab[2,2]
B[4]<-(-Eab[1,1])+Eab[1,2]+Eab[2,1]+Eab[2,2]
blTST<-(B[1]>uLim)|(B[2]>uLim)|(B[3]>uLim)|(B[4]>uLim)
#
if(nTel>nTelMax){
  print(Eab)
  Bout<-99999
}else{
  Bout<-max(B)
  if(blTST==TRUE){
    for (j in 1:4){
      if (B[j]==Bout){
        jout<-j
      }
      if(jout==3){
        blCtd<-FALSE
      }else{
        blCtd<-TRUE
      }
    }
  }
  if (blTST==TRUE){
    if(jout==3){
      print(Eab)
    }
  }
}
#
}
if (nTel%%100 ==0){
  print(Bout)
}
if (Bout==99999){
  blCtd<-FALSE
}
}
print(Bout)
print(jout)
print(nMax)
print(nTel)
#
Eqm<-matrix(0,2,2)
Eqm[1,1]=cos(2*(setA[1]-setB[1]))

```

```
Eqm[1,2]=cos(2*(setA[1]-setB[2]))
Eqm[2,1]=cos(2*(setA[2]-setB[1]))
Eqm[2,2]=cos(2*(setA[2]-setB[2]))
if(jout==3){
  Bqm=Eqm[1,1]-Eqm[1,2]+Eqm[2,1]+Eqm[2,2]
}else{
  Bqm=Eqm[1,1]+Eqm[1,2]+Eqm[2,1]-Eqm[2,2]
}
print(Eqm)
print(Bqm)
print(sum(abs(sA)<1))
print(sum(abs(sB)<1))
#stop("end")
```