A computer violation of the CHSH

Han Geurdes
GDS CvdLijnstr 164 DenHaag Netherlands
(Dated: October 22, 2019)

If a clear no-go for Einsteinian hidden parameters is real, it must be in no way possible to violate the CHSH with local hidden variable 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 > 2. The proof 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.

I. 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].

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

This correlation function is the hidden parameters, $\lambda \in \Lambda$, expression for a pardigmatic entanglement experiment:

$$[A(a)] \leftarrow\sim \ldots \sim\leftarrow\sim [S] \sim\rightarrow\sim \ldots \sim\rightarrow [B(b)] \tag{2}$$

Here,$[S]$ represent the source of entangled pairs of photons. The wavy symbols represent the photons traveling fro $[S]$ 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 $d([A],[B])$. In the formula of Bell (1) the hidden parameters, λ, are allowed to influence the outcome with

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

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

In our computer program we will look at the following CHSH inequality. Let us look at four pairs of parameter vectors: $(a,b),(a,c),(d,b)$ and (d,c).

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

Hence, we can rewrite

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

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

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

Obviously there are other CHSH inequalities like e.g.

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

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

II. MODIFIED GLAUBER-SUDARSHAN METHOD

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

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

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 the modification of the Glauber-Sudarshan method. We refer to [4, page 266]. The modificaation is

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

Then we note that $f\cos\phi$ and $\sin\phi$ modify the definitions of $\gamma_\pm$ [4, page 266]. Further 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 to B we see $\delta_{\alpha_+^{(B)},s_\pm^{(B)}}$, with, $s_\pm^{(B)} \in \{-1,1\}$. Hence

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

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

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

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

$$\begin{aligned}|\gamma_+|^2 = |\alpha_+|^2\cos^2\theta + |\alpha_-|^2\sin^2\theta + f^2\cos^2\phi + \\ 2\alpha_+\alpha_-\cos\theta\sin\theta + \\ 2\left(\alpha_+\cos\theta + \alpha_-\sin\theta\right) f\cos\phi\end{aligned} \tag{12}$$

together with

$$\begin{aligned}|\gamma_-|^2 = |\alpha_+|^2\sin^2\theta + |\alpha_-|^2\cos^2\theta + \sin^2\phi + \\ -2\alpha_+\alpha_-\cos\theta\sin\theta - \\ 2\left(\alpha_+\sin\theta - \alpha_-\cos\theta\right)\sin\phi\end{aligned} \tag{13}$$

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

$$\begin{aligned}|\gamma_+|^2 + |\gamma_-|^2 = |\alpha_+|^2 + |\alpha_-|^2 + \\ \left\{f^2\cos^2\phi + 2\left(\alpha_+\cos\theta + \alpha_-\sin\theta\right) f\cos\phi\right. \\ \left. -2\left(\alpha_+\sin\theta - \alpha_-\cos\theta\right)\sin\phi + \sin^2\phi\right\}\end{aligned} \tag{14}$$

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

$$f^2\cos^2\phi + 2\left(\alpha_+\cos\theta + \alpha_-\sin\theta\right) f\cos\phi - 2\left(\alpha_+\sin\theta - \alpha_-\cos\theta\right)\sin\phi + \sin^2\phi = 0 \tag{15}$$

If we define,

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

then

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

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

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

This latter equation is the nucleus of the computer program. In the evaluation we will look at the formula

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

which is the discrete equivalent of (8). The $S^{(A)}$ and $S^{(B)}$ are computed according to (11). The value of ϕ is a fixed parameter in the algorithm. It represents a chracteristic 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 repective intruments* 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 $\left\{\theta^{(A(B))}, s_{\pm}^{(A(B))}, \phi\right\}$.

III. 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

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

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.

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

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

$$E^{(qm)}(\theta_1^{(A)}, \theta_1^{(B)}) - E^{(qm)}(\theta_1^{(A)}, \theta_2^{(B)}) + E^{(qm)}(\theta_2^{(A)}, \theta_1^{(B)}) + E^{(qm)}(\theta_2^{(A)}, \theta_2^{(B)}) \approx 2.402191 \tag{22}$$

A. Crucial response functions

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

```
if(thet==(113.48717*pi/180)){
  sigmaA<-(-1)
}else{
  sigmaA<-sign(runif(1)-0.5)
}
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 cuase that there are a couple of runs necessary to find a Bell CHSH violation. On the B(ob)-side we have for $s_{\pm}^{(B)}$

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

In both cases runif(1) draws a random number. The sign gives a -1 if that number is less than 0.5 and 1 when larger 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 100 pairs of "photons", the two snippets will warrant a substantial violation close to the quantum CHSH-criterium value (22). This must be completely impossible if the CHSH is really waterproof.

B. 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
  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))
    }
  }
  return(S1)
}
```

as part of the A side evaluation. The B side is similar. 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) 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 III A would be impossible. It is not. In this section it is crystal clear that $|S1| \leq 1$.

C. 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))
    }
```

```
      else{
        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 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 III B. 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].

D. 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 nMax times in an array ALICE and an 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 III C, III B and III A.

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 $norM=nMax/4$ 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.

IV. RESULT, CONCLUSION & DISCUSSION

Like we already anticipated in previous sections we claim here that 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 a result of our computations with the algorithm. Its output is verbatim:

```
          [,1]      [,2]
[1,] 0.9092034 0.162797
[2,] 0.9394986 0.672403
[1] 2.358308
[1] 3
[1] 100
[1] 57
          [,1]       [,2]
[1,] 0.9999552 -0.3817306
[2,] 0.8514255  0.1690791
[1] 2.40219
```

The first matrix

$$E = \begin{pmatrix} 0.9092034 & 0.162797 \\ 0.9394986 & 0.672403 \end{pmatrix} \tag{23}$$

is based on the computations from subsection III D which refers back to the correlation in equation (19) using the modified Glauber-Sudarshan method. The number 2.358308 represents the B' as defined in (2) applied to the E matrix from the modified Glauber-Sudarshan method (23). The number 3 in the output denotes that B' CHSH from (2) is violated. So we may note that, with Einstein local hidden parameters, the CHSH can be violated size: $B' = 2.358308$ in the output of the algorithm. The number 100 in the output refers to nMax, the amount of "photon" pairs inspected. Obviously this is somewhat small. However, it is the principle violation that counts. This has got nothing to do with the number of photons and is simply restricted by the limited computational power of the machine. The number 57 refers to the number of runs (from 1 to max 1000) to compute a violating B'. In the 57-th run the A and B snippets, in particular the random parts in the ifs of section III A are able to produce violating S values that generate E in (23).

We note 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 $B_{min} = 2 + (1/\sqrt{2})$ such as sometimes is requested. 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 $B_{min} = 2 + (1/\sqrt{2})$.

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

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

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 that the algorithm does not reproduce the quantum values.

Nevertheless the conclusion is that the CHSH *can* be *violated* with local parameter principles. This concurs with e.g. [6], [7] and [8]. The complete R program is included in an appendix.

We acknowledge that the quantum values were not obtained. Perhaps that better algorithms for the generation of alpha can improve the match with quantum mechanics. However, with the power of argument, we reject conclusively that CHSH type of inequalities can not be violated with local hidden parameter computations.

---

[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] H. Geurdes, "A probability loophole in the CHSH", Results in Physics, 4, 81, (2014).
[7] H. Geurdes, K. Nagata, T. Nakamura & A. Farouk, "A note on the possibility of incomplete theory", arxiv 1704.00005, (2017).
[8] 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 to enable the reader to check the locality claim. R-studio and R-software are free to obtain from the internet.

```
#CHSH
###############
##ALICE
###############
fCHSHA<-function(alpha,thet){
  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))
      }
      else{
        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{
    sigmaA<-sign(runif(1)-0.5)
  }
  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))
    }
  }
  return(S1)
}
###############
##BOB
###############
fCHSHB<-function(alpha,thet){
  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))
      }
      else{
        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)
  #
  if(thet==(-82.32930*pi/180)){
    sigmaB<-(-1)
  }else{
    sigmaB<-sign(runif(1)-0.5)
  }
  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))
      S2<-S1
    }
  }
  return(S2)
}
#####
#main
#####
alpha<-array(0,2)
#
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
uLim<-2.28
jout<-0
sTrial<-sample(seq(1,length(BOB)))
while(Bout<uLim){
  Eab<-matrix(0,2,2)
  #
  nTel<-nTel+1
  for(m in sTrial){
    #print(n)
    thetaA<-setA[ALICE[m]]
    thetaB<-setB[BOB[m]]
    sA[m]<- fCHSHA(alpha,thetaA)
    sB[m]<- fCHSHB(alpha,thetaB)
  }
  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>1000){
    print(Eab)
    Bout<-99999
  }else{
    if (blTST){
      print(Eab)
    }
    Bout<-max(B)
    if(blTST){
      for (j in 1:4){
        if (B[j]==Bout){
          jout<-j
        }
      }
    }
  }
  print(Bout)
}
print(jout)
print(nMax)
print(nTel)
#plot(ALICE[sTrial],type='l')
#plot(BOB[sTrial],type='l')
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)
```