Developing a reference Library of Hub Test statistics

Richard Shurtleff *

Abstract

Many measurements of astronomical objects involve transverse vectors, directions tangent to the Celestial Sphere, such as polarization vectors, jets, major and minor axes, to name a few. Applying the Hub Test to a sample of transverse vectors yields measures of the correlations among the vectors' directions. How well do the directions aim toward points on the sphere, *i.e.* do they convergence? Do the directions avoid some point on the sphere, *i.e.* do they divergence? Judging significance requires developing statistics of samples of randomly directed transverse vectors, which is what this paper describes. Many artificial samples with randomly directed transverse vectors are created and the Hub Test is applied to each. The many results make distributions that are fit by suitable formulas. Thus probability distributions can be recovered by knowledge of the parameters, a great reduction in storage space. The collection of parameters for the many distributions makes the reference Library, a compact archive of statistical information. Having such a Library streamlines the process of finding the significance of Hub Test results. Two computer programs are provided in the Appendices so that creating the distribution parameter data can be repeated.

Keywords: Alignment; Hub Test; Transverse Vectors; Polarization

Comments: 8 page article followed by 25 pages of computer programs in two Appendices; 8 figures

*Department of Sciences, Wentworth Institute of Technology, 550 Huntington Avenue, Boston, MA, USA, 02115, orcid.org/0000-0001-5920-759X, e-mail addresses: shurtleffr@wit.edu, momentummatrix@yahoo.com

TABLE OF CONTENTS

1. Introduction

Many of the astronomical quantities observed are or are related to vectors pointing perpendicular to the line-of-sight, tangent to the Celestial Sphere. We have, for example, the polarization direction of electromagnetic radiation and the direction of asymmetries like jets. It is natural to ask if the directions measured for a sample of sources are correlated, aligned. But 'aligned' can have a number of meanings.

The Hub Test of alignment extends the transverse directions, making Great Circle geodesics on the Celestial Sphere. The transverse directions are perfectly aligned if they intersect at some point H on the sphere. The directions are well-aligned when the Great Circles converge in a small area near some point $H_{\min}$. The Hub Test can find correlations for samples with hubs $H_{\min}$ that are near the sources as well as the distant Hubs that other alignment tests would also detect.

The Hub Test differs from other tests that evaluate the correlations of transverse vectors dispersed over the Celestial Sphere. The notion of alignment is different. The idea that transverse vectors can align with a point on the sphere, focused as in Fig. 3, differs from the requirement that transverse vectors align because they point in the same direction. 'Pointing in the same direction' means having the same position angle, parallel. Being parallel is not required with the Hub Test when, as in Fig. 3, the transverse vectors focus on a nearby point. The `S' and `Z' tests, and similar tests, find alignments by comparing polarization position angles. Comparing position angles directly at different points on the curved surface of a sphere is complicated by parallel transfer. Such complications have been faced and resolved and the `S' and `Z' tests are well documented and reliably detect that type of alignment. Refs. 1-5.

This article describes a way to produce statistics for the Hub Test. The Hub Test is described in more detail in Ref. 4. However, any quantity obtained by calculation from observations has a significance determined by the statistics of identical procedures applied to random data. The goal here is to create a 'Library' of statistics for a large number of artificial cases. Then one is able to estimate the significance of an observed sample by reference to the 'Library' of artificial cases, for use when such an estimate is sufficient. Avoiding the process of creating statistics for each observed sample may be a convenient time-saver.

The process is split into several parts. First, a computer program generates many 'random runs', with each 'run' an application of the Hub Test to a sample with randomly directed transverse vectors. The data is collected and stored in 'runData' data files. A discussion appears in Sec. 2 and the FORTRAN-90 program in Appendix A performs the tasks.

Next, by the Mathematica program in Appendix B, the outcomes of those random runs form distributions which can be fit by suitable functions. The process is discussed in Sec. 3. Those fitting functions have three parameters, the scale constant, the location of the peak and the distribution width. By storing the parameters in a data file, 'fitData', one can reconstitute the distributions and make probability distributions. For the probability distributions, the scale constant is fixed by the requirement that a probability distribution must integrate to unity. Thus the probability distributions are fixed by the values of two parameters, the location of the peak and the half-width.

Once the fitData file is built, one can then apply the Library to observed samples whose alignments have been measured with the Hub Test. For an experimentally observed sample, one can interpolate the Library's cases. One gets a probability distribution for random data by interpolating the Library data with the observed sample by comparing two properties, the number of sources and the root-mean-square sample radius. Or, another path is possible. One can fit the Library data with fitting functions for the peak and half-width. Then substitute the observed sample's properties to find a candidate probability distribution for the observed sample.

This article deals with the first and second steps, generating random runs and fitting the resulting distributions. Applying the Library to find the significance of Hub Test results for observed samples is a topic for another article, Ref. 7.

2. Generating Random Runs

The Hub Test, Ref. 6, is based on the alignment of transverse vectors with points on the Celestial Sphere. In Fig. 1, the "alignment angle" η of polarization direction ψ with the point H on the sphere is the acute angle η between two great circles at S, $0° \le \eta \le 90°$. The alignment angle η measures how well the polarization direction $\hat{v}_\psi$ matches the direction $\hat{v}_H$ toward the point H. Perfect alignment occurs when $\eta = 0°$ and the two great circles overlap. Perpendicular great circles, $\eta = 90°$, indicates maximum "avoidance" of the polarization direction $\hat{v}_\psi$ with the point H on the sphere.

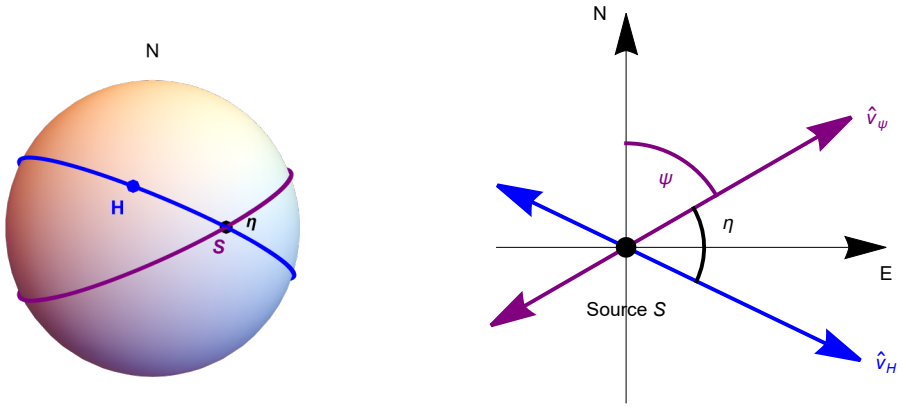

Figure 1: The Celestial sphere is pictured on the left and on the right is the plane tangent to the sphere at the source S. The linear polarization direction $\hat{v}_\psi$ lies in the tangent plane and determines the purple great circle on the sphere. A point H on the sphere together with the point S determine a second great circle, the blue circle drawn on the sphere. Clearly, H and S must be distinct in order to determine a great circle. The angle η measures the alignment of the polarization direction ψ with the point H.

With N sources S_i, $i = 1, ..., N$, there are N alignment angles η_{iH} at each point H . One can calculate an average alignment angle $\overline{\eta}$ at H,

$$\overline{\eta}(H) = \frac{1}{N} \sum_{i=1}^{N} \eta_{iH} \,, \tag{1}$$

where

$$\cos(\eta_{iH}) = |\, \hat{v}_\psi . \hat{v}_H \,| \,, \tag{2}$$

$$\cos(\eta_{iH}) = \left| \frac{\cos\delta_S \cos\psi \sin\delta_H + \cos\delta_H [\sin(\alpha_H - \alpha_S) \sin\psi - \cos(\alpha_H - \alpha_S) \cos\psi \sin\delta_S]}{\sqrt{1 - (\cos(\alpha_H - \alpha_S) \cos\delta_H \cos\delta_S + \sin\delta_H \sin\delta_S)^2}} \right| \,, \tag{3}$$

where α and δ are the longitude and latitude of the i^{th} source S and the point H. The notation 'α' and 'δ' may evoke an association with Right Ascension and Declination, but the formulas work with the longitudes and latitudes of any astronomical coordinate system. Each angle η_{iH} is taken to be the acute angle solving (2) or (3). If the η_{iH} are acute, then the average alignment angle $\overline{\eta}(H)$ at the point H must also be acute. The alignment angle $\overline{\eta}(H)$ is a function of position H on the sphere. It is symmetric across diameters, $\overline{\eta}(H) = \overline{\eta}(-H)$, because H and $-H$ lie on the same great circles through the sources S_i.

For random polarization directions, the average $\overline{\eta}(H)$ should be near 45°, since each alignment angle η_{iH} is acute, $0° \le \eta_{iH} \le 90°$, and random polarization directions should not favor large values or small values of η_{iH}, and, therefore, average to about 45°. But, recalling Brownian Motion, the sum in Eq. (1) on the right is a sum over random angles η_{iH} scattered above and below 45°. By Random Walk theory, the sum of (η_{iH} - 45°) should go like $N^{1/2}$, so $\overline{\eta}(H)$ should differ from 45° by an amount proportional to $N^{-1/2}$. Therefore as N grows larger for randomly directed samples, the average $\overline{\eta}(H)$ should approach nearer to 45°. Those expectations are confirmed by the data. See Figs. 8, A1, and A3 in Ref. 7.

Points H where the average alignment angle $\overline{\eta}(H)$ is smaller than 45°, the great circles tend to converge and where the angle $\overline{\eta}(H)$ is larger than 45°, the great circles can be said to diverge. The extremes of the function $\overline{\eta}(H)$ measure extreme convergence and extreme divergence of the great circles determined by the polarization directions.

In this article and notebook, we often use "min" to label the smallest alignment angle $\overline{\eta}_{\text{min}}$, the minimum value of the function $\overline{\eta}(H)$, Eq. (1). We have $\overline{\eta}_{\text{min}} = \overline{\eta}(\pm H_{\text{min}}) \le \overline{\eta}(H)$, for all H. The associated points on the Celestial Sphere are the "alignment hubs" H_{min} and $-H_{\text{min}}$. Thus "min" is associated with convergence of the polarization directions. The hubs H_{min} and $-H_{\text{min}}$ are points on the Celestial Sphere where the polarization directions are best aimed. For divergence, the "avoidance hubs" H_{max} and $-H_{\text{max}}$ locate places where the polarization directions most avoid, as indicated by the largest alignment angle $\overline{\eta}_{\text{max}}$, the maximum value of the function $\overline{\eta}(H)$. We have $\overline{\eta}_{\text{max}} = \overline{\eta}(\pm H_{\text{max}}) \ge \overline{\eta}(H)$, for all H. And, almost always, an avoidance related quantity is labelled with "max".

To avoid clumping, we create an artificial sample in the form of a square array. It follows that the number of sources N must be a perfect square, $N = m^2$ for some integer m. One expects a square array to be uniformly spaced over the region. Even though the array is on the surface of a sphere with its inherent curvature, we call it a 'square array' or sometimes a 'square matrix'.

The square array of sources is meant to approximate a circle of nominal radius on the sphere. See Fig. 2. The square is midway between being inscribed within and being superscribed about the circle. The radius of the circle is called the 'nominal radius' ρNOM, the arc distance from the center of the circle to a point on the circle. The sample array is rotated to avoid coincidences with the orientation of the grid.

For bookkeeping purposes, the samples are listed by the number of sources N and the nominal radius ρNom, (N, ρNOM). A great many random runs are processed for each sample (N, ρNOM). At the time of writing, there are 294 samples (N, ρNom), the grid has 1° spacing, and each sample is processed with 10,000 random runs.

For calculations and comparison with observed samples, the root-mean-square radius ρRMS, which is also shown in Fig. 2, is considered a better representation of the sample's size. It is straightforward to calculate the rms radius of any sample, so using ρRMS for comparisons is convenient.

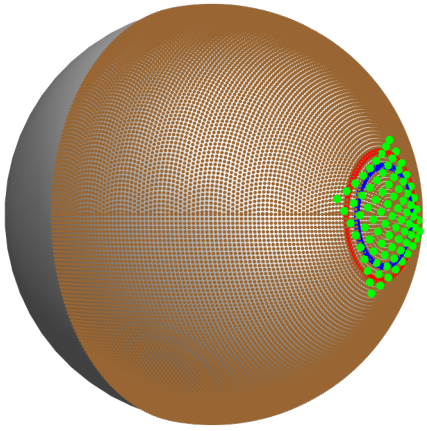

Figure 2: Grid, source array, and circles. An artificial sample of $N = 81$ sources (green) arranged in a square array on the surface of the Celestial Sphere. The square approximates a circle (red) with a 'nominal' radius ρNom = 16°. The blue circle has the root-mean-square radius of the square array, here ρRMS = 12.5°. For each random run, the sources (green) are assigned random polarization directions (not shown). Then the alignment function $\overline{\eta}(H)$, Eq. (1), is calculated at the 21,400 grid points H (brown).

The Fortran-90 computer program in App. A generates the random runs by creating the grid and the square array of sources. Then the program assigns random polarization directions to the sources and evaluates the function $\overline{\eta}$(H) on the grid. The maximum and minimum values of $\overline{\eta}(H)$ are saved, as well as some other quantities of interest, such as the hubs H_{min} and H_{max}, which are the grid points where the extreme values occur on the sphere. Each saved data file has the random run data associated with just one number of sources N but all the region radii ρNom, (one N, all ρNom). The data files are saved as plain text and are available via the links in Ref. 8.

3. Fitting the Distributions of Random Run Quantities

Next, we download, one at a time, the files containing the random run data. Each file has the random run data for a given number of sources N and all the nominal radii ρNOM. The data is analyzed by the Mathematica notebook in App. B, one value of N at a time.

The random run data includes the smallest alignment angle $\overline{\eta}_{\min}$ for alignment and the largest alignment angle $\overline{\eta}_{\max}$ for avoidance, as well as other results. For each case (N, ρNom), the random run values for $\overline{\eta}_{\min}$ and $\overline{\eta}_{\max}$ are collected in bins, forming histograms. As illustration, two of the histograms are displayed in Figs. 3 and 4.

An examination of the histogram for $\overline{\eta}_{\min}$ in Fig. 3 and the histogram for $\overline{\eta}_{\max}$ in Fig. 4 reveals that the distributions are steeper on the side toward $\eta \to 45°$. For $\overline{\eta}_{\min}$, the histogram is steeper on the high η side, while, for $\overline{\eta}_{\max}$, the histogram is steeper on the low η side. One finds that all the distributions of $\overline{\eta}_{\min}$ and $\overline{\eta}_{\max}$ show steep slopes toward 45°. See, for example, Figs. B2 and B3. This behavior appears to be an inherent property of the statistics.

For fitting random data distributions, the most convenient fitting function would be Gaussian. But Gaussians are symmetric about the peak, so we must entertain some other function. Fortunately, we find that the steep slope can be accommodated by multiplying the Gaussian by a step function, making a 'Step-Gaussian'. For $\overline{\eta}_{\min}$, the step function is unity below the peak and vanishes above the peak, while, for $\overline{\eta}_{\max}$, the step function is unity above the peak and vanishes below the peak.

For $\overline{\eta}_{\min}$ in Fig. 3, since the step function vanishes to the right of the peak, that makes the right side of the fitting function descend rapidly to zero as η increases past the peak. One can say that the less populated right side pushes probability to the left.

However, on the left, the step function is nearly unity. And the left side is more important than the right side, because the left side is where the smallest alignment angles $\overline{\eta}_{\min}$ for well-aligned samples are found. Thus, for small $\overline{\eta}_{\min}$, the tail of the curve drops off like a Gaussian, but with a probability gain of some 20% over a symmetric Gaussian. The normalization factor can be calculated directly, see Ref. 7.

Appropriately revised comments apply for the distribution of the largest avoidance angle $\overline{\eta}_{\max}$ in Fig. 4. For the measure of avoidance $\overline{\eta}_{\max}$, the distribution occurs above 45°, with the steep side on the left of the peak, the side toward 45°. See Fig. 4. The appropriate step function vanishes to the left and is unity to the right of the peak. Again, the important side, the right side, has a Gaussian dropoff for large η, but with a 20% increase over a symmetric Gaussian distribution.

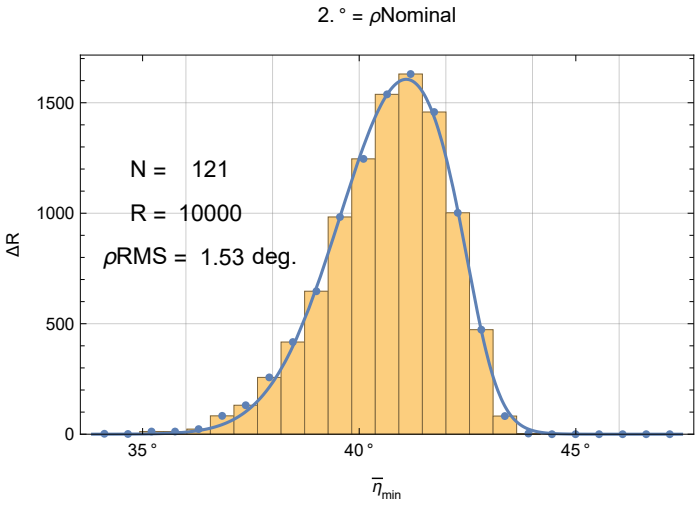

Figure 3: The histogram and fit for $N = 121$ and ρNom $= 2°$, the distribution of the $R = 10{,}000$ values of the smallest alignment angle $\overline{\eta}_{\min}$ from the random runs. As noted in the text, the histogram approximates the shape of the probability distribution, aside from a normalizing scale factor. These distributions are fit by a Gaussian multiplied by a step function, a 'Step-Gaussian'. Thus, the $R = 10{,}000$ values can be represented by the two parameters, the location of the peak and half-width, needed to determine the fitting function.

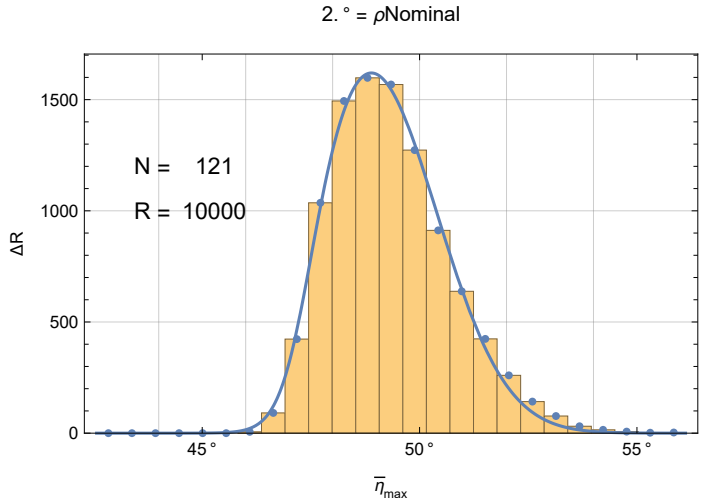

Figure 4: The histogram and fit for $N = 121$ and $\rho\text{Nom} = 2°$, the distribution of the $R = 10{,}000$ values of the largest avoidance angle $\overline{\eta}_{max}$ from the random runs. These distributions are fit by a Gaussian multiplied by a step function, a 'Step-Gaussian'.

Consider the histogram of the smallest alignment angles $\overline{\eta}_{min}$ from R = 10,000 runs that is displayed in Fig. 3. The height of a bar in Fig. 3 is the number ΔR of the random runs that gave a value of $\overline{\eta}_{min}$ in the interval $\delta\eta$, the width of a bin. Thus, the quantity ΔR/R is the likelihood that random runs give a value of $\overline{\eta}_{min}$ in the bin. The histogram has the shape of the probability distribution $P_{min}(\eta)$ for $\overline{\eta}_{min}$, differing by a normalization factor. Instead of integrating to 10,000, a probability distribution is normalized and integration yields unity.

The probability distribution is normalized, $1 = \int P_{min}\, d\eta$. Since the sum $\Sigma\, \Delta R = R$, it follows that $1 = \Sigma \frac{\Delta R}{R \delta\eta}\, \delta\eta$ and that $P_{min}(\eta) \approx \frac{\Delta R}{R \delta\eta}$. Therefore, by fitting the histogram, we have also found the probability distribution $P_{min}(\eta)$.

As noted above, there is a complication. In each figure, Figs. 3 and 4, the side toward $\overline{\eta}_{min} \to 45°$ has a steeper slope than the other side. The steeper slope is on the right in Fig. 3 and on the left in Fig. 4. Thus, to accommodate this behavior, the fitting curve shown in blue in Fig. 3 combines a Gaussian with a unit step-function, one that is unity well to the left of the peak, and zero well to the right. We choose the Step-Gaussian distribution function

$$f_{min}(\eta) = a \left(1 + e^{4 \frac{(\eta - \eta 0 - b)}{b}}\right)^{-1} e^{-\frac{1}{2}\left(\frac{\eta - \eta 0}{b}\right)^2}, \qquad (4)$$

To fit the histogram for $\overline{\eta}_{max}$ in Fig. 4, we have another step function, but with a couple of sign changes,

$$f_{max}(\eta) = a \left(1 + e^{-4 \frac{(\eta - \eta 0 + b)}{b}}\right)^{-1} e^{-\frac{1}{2}\left(\frac{\eta - \eta 0}{b}\right)^2}, \qquad (5)$$

where the scale factors a, half-widths b, and values $\eta 0$ at the peaks are varied to fit the $\overline{\eta}_{min}$ and $\overline{\eta}_{max}$ histograms like those in Figs. 3 and 4. These equations appear again in Sec. B5 of App. B, Eqs. (B4, B5), where the histograms are fit.

There are two free variables in Eq. (4) and two more in Eq. (5) that can be chosen so that a scaled version fits random run histograms like the ones shown in Figs. 3 and 4. We call η_0 the 'peak' value, the value of η at the peak, and b is called the 'half-width' usually denoted σ. The presence of the step functions $\left(1 + e^{\pm 4 \frac{(\eta - \eta 0 \mp \sigma)}{\sigma}}\right)^{-1}$ moves the peak and half-width a little from their Gaussian values without the step function. The fitting function has two parameters just like a Gaussian, namely the peak η_0 and the half-width σ.

Note that the step function is introduced without an associated parameter. Adding adjustable parameters was deemed ineffective at producing better fits. It is true that, with additional parameters, the step function can be adjusted to give better fits than without such parameters. One could reasonably argue that the '4' in the step functions could be replaced by some parameter 'c' and varied. However, we stay with '4' because '4' gives comparable fits to those of the best fits found when 'c' was varied.

We did not mention that 'a' in Eq. (4) and (5) is a free variable. The scale factors 'a' are largely ignored because we are interested in the probability distributions. With a probability distribution the factor 'a' is replaced by a function of σ that is determined by the requirement that the likelihood of all possible outcomes be unity.

Upon completion of these steps, we have a set of four parameters, $\eta_0^{\min}$, $\sigma^{\min}$, $\eta_0^{\max}$, and $\sigma^{\max}$, for many samples (N, ρNom). The file containing the output, the fitData table, is available via the link in Ref. 9.

Our work here is done. Based on the samples treated here, the parameters for an experimentally observed sample can be estimated and a value for the significance of the Hub Test results can be found. As mentioned previously, the process of applying the results of this article to real data is treated in another article, Ref. 7.

4. Conclusion

To conclude this work, let us give a brief overview of the rest of the story, completed in Ref. 7. Consider only the case of alignment, with the value of the smallest alignment angle $\overline{\eta}_{\min}$ of the sample polarization directions with any point on the sphere. Correlations with the largest avoidance angle $\overline{\eta}_{\max}$ have a similar story.

Given the Step-Gaussian function in Eq. (4), the problem of determining the significance of an observed $\overline{\eta}_{\min}$ reduces to estimating the values of two parameters, η_0 and σ, since those two values determine the probability distribution $P_{\min}(\eta)$. The distribution $P_{\min}(\eta)$ is the same as $f_{\min}(\eta)$ in Eq. (4) except for normalization, we must change the scale factor 'a' so the integral of $P_{\min}(\eta)$ over all η is unity.

By knowing the observed values of N and ρRMS of an experimentally observed sample, we can estimate the parameters $\eta_0^{\min}$ and $\sigma^{\min}$ by comparing the observed N and ρRMS with the values of (N, ρRMS) of the collection of $\eta_0^{\min}$ and $\sigma^{\min}$ found in this article. The comparison yields an estimate of the probability distribution $P_{\min}(\eta)$ of the smallest alignment angle $\overline{\eta}_{\min}$ for the observed sample.

Once determined, the probability distribution for an observed sample can be integrated to obtain the significance of an observed $\overline{\eta}_{\min}$. For example, to find the significance, or p-value, of the smallest alignment angle $\overline{\eta}_{\min}$, one finds the likelihood of smaller random run values by integrating $P_{\min}(\eta)$ from below,

$$p(\overline{\eta}_{\min}) = \int_{-\infty}^{\overline{\eta}_{\min}} P_{\min}(\eta)\, d\eta \quad . \qquad (6)$$

Thus, by Eqs. (4) and (6) the significance p of the correlated behavior indicated by the smallest alignment angle $\overline{\eta}_{\min}$ rests on finding the two probability distribution parameters $\eta_0^{\min}$ and $\sigma^{\min}$ in (4), where these were written $\eta 0$ and b. For details, consult Ref. 7.

Quantitative measures, like $\overline{\eta}_{\min}$ and $\overline{\eta}_{\max}$, require statistics to figure out how likely it is that their values reflect some unusual process or characteristic that is in need of explanation. Results that are usual or commonplace contribute in bulk to the overall picture. But it is the occasional and unlikely outcomes that signal special attention. Observing bulk random linear polarization directions from a number of sources would compel only an explanation of why those sources are linearly polarized. When the directions point significantly to some specific location on the Celestial Sphere, then additional explanation is expected. This paper lays the foundational drudge work that contributes to the goal of distinguishing usual outcomes from those that are special.

References

1. Hutsemekers, D., Evidence for very large-scale coherent orientations of quasar polarization vectors, Astronomy and Astrophysics 332, pp. 410-428 (1998).
2. Jain, P., Narain, G., and Sarala, S., Large-scale alignment of optical polarizations from distant QSOs using coordinate-invariant statistics, MNRAS 347, pp. 394-402,
doi = {10.1111/j.1365-2966.2004.07169.x} (2004).
3. Pelgrims, V. and Cudell, J. R., A new analysis of quasar polarization alignments, MNRAS 442, pp. 1239-1248, doi = {10.1093/mnras/stu966} (2014).
4. Bietenholz, M. F., Determining dependences between directional quantities and position on a sphere, AJ 91, pp. 1249-1252, doi = {10.1086/114100} (1986).
5. Pelgrims, V., Cosmic anisotropies from quasars: from polarization to structural-axis alignments, PhD thesis, https://arxiv.org/abs/1604.05141 (2016).
6. Shurtleff, R., "Indirect polarization alignment with points on the sky, the Hub Test" , https://vixra.org/abs/2011.0026 (2020).
7. Shurtleff, R., "Alternate methods for judging the significance of Hub Test results", https://vixra.org/abs/2112.0109 (2022).
https://www.dropbox.com/s/06c370jwow8aaut/20220512InterpolateAndFormula7.nb?dl=0 (2022);
https://www.dropbox.com/s/06c370jwow8aaut/20220512InterpolateAndFormula7.nb?dl=0 (2022);
8. Shurtleff, R., links to random run data files,
ReadMe.txt: https://www.dropbox.com/s/rwyn4iu900nut8c/ReadMe.txt?dl=0
N = 9: https://www.dropbox.com/s/nlp0w6r65fw50hp/2205randomV5N9.dat?dl=0
N = 16: https://www.dropbox.com/s/4mj60dpuxxewvwz/2205randomV5N16.dat?dl=0
N = 25: https://www.dropbox.com/s/ggzjz9thouuvqil/2205randomV5N25.dat?dl=0
N = 36: https://www.dropbox.com/s/pmatw0ur12cuhsq/2205randomV5N36.dat?dl=0
N = 49: https://www.dropbox.com/s/86znuf54vak1erj/2205randomV5N49.dat?dl=0
N = 64: https://www.dropbox.com/s/q5q7w9ce9fgh361/2205randomV5N64.dat?dl=0
N = 81: https://www.dropbox.com/s/gsf5zs6hzyktgae/2205randomV5N81.dat?dl=0
N = 100: https://www.dropbox.com/s/xx2dzny2orksa1y/2205randomV5N100.dat?dl=0
N = 121: https://www.dropbox.com/s/ghzazka7ry7xtb1/2205randomV5N121.dat?dl=0
N = 225: https://www.dropbox.com/s/isv7efom34p30fo/2205randomV5N225.dat?dl=0
N = 256: https://www.dropbox.com/s/o075nbsfml75gay/2205randomV5N256.dat?dl=0
N = 324: https://www.dropbox.com/s/psg0g02loyzsgcb/2205randomV5N324.dat?dl=0
N = 625: https://www.dropbox.com/s/wxr99fxm91uut87/2205randomV5N625.dat?dl=0
N = 900: https://www.dropbox.com/s/4ipvinbgb6e0pnv/2205randomV5N900.dat?dl=0 (all 2022)
9. Shurtleff, R., the link to the Library fitting function data table fitData: https://www.dropbox.com/s/5058eky24o815tn/20220514-fitData21Rgns14Nsv5.dat?dl=0 (2022)
10. Shurtleff, R., this article and its Mathematica notebook are available online with the ready-to-run Mathematica version of Appendix B. The links:
https://www.dropbox.com/s/247ossmn1cqwg61/20220210FortranRunDataToMMAToFitData.nb?dl=0 (2022)
https://www.wolframcloud.com/obj/shurtleffr/Published/20220210FortranRunDataToMMAToFitData.nb (2022)
11. Shurtleff, R., the FORTRAN-90 program in Appendix A can be downloaded in 'plain text' via the following link: https://www.dropbox.com/s/v7trg7sedi6oc9a/20220515RunDataGenerator6.f90?dl=0 (2022)
12. Gfortran, GNU Fortran project, Fortran 95/2003/2008/2018 compiler for the GNU Compiler Collection, https://gcc.gnu.org/wiki/GFortran (2018).
13. Wolfram Research, Inc., Mathematica, Version 13.0.1.0, Champaign, IL (2022).

Notes:

(1) This article is available online with a ready-to-run Mathematica version of Appendix B. Follow one of the links in Ref. 10. The various files and versions in Ref. 10 and 11 may be updated out-of-synch with one-another.

(2) The FORTRAN-90 code in Appendix A can be downloaded in plain .txt from a link in Ref. 11. The FORTRAN-90 program is consistent with gfortran — the GNU Fortran compiler, part of GCC, Ref. 12.

(3) Mathematica, Ref.13, has the option of hiding cells. We hide some cells. For example, cells that contain only "print" statements are hidden. To see a hidden cell, follow a link in Ref. 10, open the file in Mathematica, and change the cell's properties.

```
! Appendix A
! A Fortran-90 Computer Program for Generating Random Runs
! CONTENTS
! A1. Introduction
! A2.  Begin the program, INTERFACE blocks
! A3.  Type Declarations
! A4.  Set parameters to control the program
! A5.  Construct the grid
! A6.  Construct samples and apply the Hub Test
! A7.  Save results to files, end program
! A8.  External functions
!234567891123456789212345678931234567894123456789512345678961234567897123456789812345678991234567 89
! Preface
! We use standard right-handed Cartesian coordinates x,y,z. For spherical coordinates,
! the latitude is -Pi/2 at the South Pole (x,y,z) = (0,0,-1) and +P/2 at the North Pole (0,0,+1).
! At the Equator, lat = 0 . Angles are in radians unless stated otherwise.
! This file: 20220515RunDataGenerator6.f90, the sixth version.
! To download a copy, follow the link in Ref. 11.
!
! A1. Introduction
!      The Hub Test evaluates the correlation of transverse vectors on a sphere such as the
! polarization directions of light from astronomical sources. Statistics are
! needed to judge the significance of the Hub Test results. This program applies
! the Hub Test to randomly directed transverse vectors. Each application of the Hub Test is
! called a 'random run.'
!       The program starts with INTERFACE blocks in Sec. A2, because there are external programs
! defined in Sec. A8 after the END PROGRAM statement. The INTERFACE blocks are followed
! by Type Declarations in Sec. A3.
!       Some parameters controlling the program are set in Sec. A4 Settings. In Sec. A5, we define
! a grid of evenly spaced points on the Celestial Sphere. These grid points are the points H on the
! sphere where the function Eta Bar is evaluated. It is only necessary to cover one hemisphere because
! the main Hub Test function is symmetric across diameters. The grid covers the hemisphere centered
! on the positive y-axis.
!    Artificial samples are constructed and evaluated in Sec. A6. See Fig. 2 in the text. Each sample is a
! a square array, so the number of sources N must be a perfect square. The number N is called
! nSrc in the program. The size of the array is intended to approximate a circular region with a
! nominal radius RhoNom.  By the 'radius' of a circular region on a sphere, we mean the angular distance
! from the center to the edge. The two quantities N and RhoNom determine a single case.
!    Next, the Hub Test is applied, still in Sec. A6. Given one of the artificial samples, we calculate the
! average alignment angle function EtaBar at the grid points. The maximum and minimum of the alignment
! function are collected along with the grid points where the extreme values occur. The location of these
! extrema are called 'Hubs'. The program generates many random runs for each case of  (N,RhoNom).
! Another program, in Appendix B, takes the random runs and determines Hub Test statistics.
!     In Sec. A7, the collected data is written to files. Each file name contains the number of sources,
! one N for each file. The results for all the region radii RhoNom are included in the file with the given N.
! Thus, there is one file for N = 9 sources, one for N = 16 sources, etc, with each file containing all the
! array sizes RhoNom. Then the program ends.
!     In Sec. A8, some external functions are defined for use in the program.
!
```

```
program RunData
        ! A2. INTERFACE Blocks
!
IMPLICIT NONE
        ! The functions are defined below, after the program ends.
INTERFACE  ! er, radial unit vector to a point on the Celestial Sphere at a given longitude, latitude
  FUNCTION er(Lon, Lat) RESULT  (w) !Lon, Lat = longitude, latitude
    REAL, DIMENSION(3) :: w
    REAL, INTENT(IN) :: Lon,Lat  ! longitude and latitude
  END FUNCTION
END INTERFACE
INTERFACE  ! eN, local North at the point on the sphere with the given longitude, latitude
  FUNCTION eN(Lon, lat) RESULT  (w)
    REAL, DIMENSION(3) :: w
    REAL, INTENT(IN) ::Lon,Lat  ! longitude and latitude
  END FUNCTION
END INTERFACE
INTERFACE  ! eE, local East at the point on the sphere with the given longitude, latitude
  FUNCTION eE(Lon, lat) RESULT  (w)
    REAL, DIMENSION(3) :: w
    REAL, INTENT(IN) ::Lon,Lat  ! longitude and latitude
  END FUNCTION
END INTERFACE
INTERFACE  ! the longitude at the point on the sphere with the given radial vector r
  FUNCTION LonFromR(r) RESULT (w)
    REAL :: w
    REAL, INTENT(IN) :: r(3)   ! the given radial vector
  END FUNCTION LonFromR
END INTERFACE
INTERFACE  ! LatFromR,  the latitude at the point on the sphere with the given radial vector r
  FUNCTION LatFromR(r) RESULT (w)
    REAL :: w
    REAL, INTENT(IN) :: r(3)   ! the given radial vector
  END FUNCTION LatFromR
END INTERFACE
INTERFACE  ! Rotation matrix for a rotation about the x-axis through an angle theta
  FUNCTION rotAboutX(theta) RESULT  (w)
    REAL, DIMENSION(3,3) :: w
    REAL, INTENT(IN) :: theta   ! the rotation angle
  END FUNCTION
END INTERFACE
INTERFACE  ! Rotation matrix for a rotation about the y-axis through an angle theta
  FUNCTION rotAboutY(theta) RESULT  (w)
    REAL, DIMENSION(3,3) :: w
    REAL, INTENT(IN) :: theta   ! the rotation angle
  END FUNCTION
END INTERFACE
INTERFACE  ! Rotation matrix for a rotation about the z-axis through an angle theta
  FUNCTION rotAboutZ(theta) RESULT  (w)
    REAL, DIMENSION(3,3) :: w
    REAL, INTENT(IN) :: theta   ! the rotation angle
  END FUNCTION
END INTERFACE
INTERFACE  ! The cross product r0xr1 of two vectors r0 and r1
  FUNCTION cross(r0,r1) RESULT  (w)
```

```
      REAL, DIMENSION(3) :: w
      REAL, INTENT(IN) ::  r0(3), r1(3)   ! the two vectors
   END FUNCTION
END INTERFACE
!
                ! A3. Type declarations
!
!   General
REAL, PARAMETER ::  PI  = 4* ATAN(1.)    ! the constant pi, 3.14159...
REAL ::  rotZtoY(3,3)
INTEGER :: i,j,k,m,n,ir, iN ! dummy indices
REAL  ::  x,y,z     ! dummy variables, usually Cartesian coordinates
REAL, DIMENSION(3) :: r, r0, r1, r2, r3, r4, r5, r6   ! dummy vectors
REAL :: theta   ! generic angle
!   Grid
INTEGER, PARAMETER ::   MaxDimGrid = 30000    ! maximum number of grid points
INTEGER :: nGrid    ! the number of grid points
REAL  :: LonH, latH, LonGrid0(MaxDimGrid), LatGrid0(MaxDimGrid) ! initial longitude and latitude of the grid points
REAL  :: LonGrid(MaxDimGrid), LatGrid(MaxDimGrid) ! longitude and latitude after the grid is rotated
REAL ::  Rgrid(MaxDimGrid,3)    ! unit radial vectors to the grid points
REAL ::  LonRand, LatRand ! Random longitude and latitude
!   Samples.
!   Each sample is identified by (nSrc, RhoNom), the number of sources and the region's nominal radius
!   Number of sources
INTEGER, PARAMETER ::   MaxNumSources = 1600    ! maximum number of sources
INTEGER, PARAMETER ::   MaxNumOfnSrc = 25    ! maximum number of different values of nSrc
INTEGER  ::   nSrcList(MaxNumOfnSrc) ! list of the numbers of sources evaluated in the program
INTEGER  ::   nSrc ! number of sources
INTEGER  ::   rootNsrc    ! square root of the number of sources
INTEGER  ::   NumberOfnSrc  ! number of distinct nSrc cases considered
!   Nominal radius of the sample's region
INTEGER, PARAMETER ::   MaxNumRgnRadii = 25    ! maximum number of nominal radii considered
REAL  ::  RhoNomList(MaxNumRgnRadii)  ! the list of nominal radii evaluated in the program
INTEGER ::  numberOfRgnRadii    ! the number of region radii evaluated in the program
!   The square array of sources
REAL  ::    LonS, latS, LonSource0(MaxNumSources), LatSource0(MaxNumSources) ! initial source longitudes
!         and latitudes
REAL  ::    LonSource(MaxNumSources), LatSource(MaxNumSources) ! final longitude and latitude after moving
REAL ::     RSource(MaxNumSources,3) ! unit radial vectors to the sources from the origin
REAL ::     NSource(MaxNumSources,3),ESource(MaxNumSources,3) ! local North, local East at each source, nit vectors
REAL ::     rCenterSrc0(3), rCenterSrc(3),LonCenterSrc,LatCenterSrc ! unit radial vectors to the center before and
!                 after moving the sources,  longitude, latitude
REAL, DIMENSION(3) ::  rSrcAve   ! unit radial vectors to the center, approx. equal to but calculated differently
!                                          from rCenterSrc
REAL ::     aSM, daSM  ! arc length of one side of the square array of sources, nearest neighbor distance
REAL ::     dotRSrcWithAve    ! dot product of each source to the center of the sample
REAL ::     rSrcSum(3)   ! the sum of the sources' unit radial vectors
REAL ::     RhoRMSsquared, RhoRMS    ! root mean square radius squared, root mean square radius
REAL  ::    psiSource0(MaxNumSources),psiSource(MaxNumSources)  ! polarization position angle
!   Mixed quantities, Grid with sources, grid with hubs, source with polarization, etc
REAL    :: rSrcxrGrid(MaxNumSources,MaxDimGrid,3) !  SxH, unit vector, cross product of source S and grid point H
INTEGER, PARAMETER ::   minGridCenterToHmin = 2  ! minimum number of grid spaces between hub Hmin
!                       and sources' center
INTEGER, PARAMETER ::   minGridCenterToHmax = 2 ! minimum number of grid spaces between hub Hmax
```

```
!                           and sources' center
REAL :: rSrcxPsiSource(MaxNumSources,3) ! the cross product of the radial source vector with the polarization direction,
!            a unit vector
!   We split the random runs into units of random runs each.
INTEGER ::   nRunMax       ! the number of random runs in one unit
INTEGER ::   kiloRunMax    ! the number of units of random runs generated for each sample (nSrc,RhoNom)
INTEGER  ::  nRun, kiloRun  ! run # counter, kilorun # counter
!   Results
REAL :: sumForjEtaBar !    total of the alignment angles Etaij from the sources to the grid point j
REAL :: jEtaBarToGrid(MaxDimGrid) ! average of the alignment angles eta from the sources to each grid point j
REAL :: EtaBarMin, EtaBarMax !    minimum, maximum of the average alignment values jEtaBarToGrid
REAL :: EtaBarMin1, EtaBarMax1 !  copies of minimum, maximum of the average alignment values jEtaBarToGrid
!                    needed for logic
REAL :: jEtaBarMin(2), jEtaBarMax(2)    ! (jmin, EtaBarMin) and (jmax, EtaBarMax), grid point ID# j and extreme
!             alignment angle value
REAL :: SourceCenterToH ! angle from rSrcAve to grid point j calculated for all j, saved only for the min eta hub jmin
!       and max eta hub jmax
LOGICAL :: okMin, okMax ! As average alignment angles are found these are TRUE for the smallest (largest) average
!                  alignment angle so far
REAL :: OutRunData(MaxNumSources,24) !  output data to be saved in a file
CHARACTER (len=20) :: file_name ! variable file name for output data
!   Set the grid parameters and the source square array parameter ('SM' for Source Matrix)
REAL, PARAMETER ::  dTheta1  = 1.0*(2*Pi/360) ! Grid spacing in radians
REAL, PARAMETER ::  gridExtent  = 1.6 ! the angle in radians from the pole to the grid edge, Pi/2 is a
!                             hemisphere, Pi is a sphere
REAL, PARAMETER ::  thetaSM  = 0.015 ! Rotate the source array to avoid coincidences with the coordinate axes
PRINT *,    'The nearest neighbor grid spacing is  ', dTheta1, ' radians and ', dTheta1*(360./(2.*Pi)),' degrees. '
PRINT *,    'The Grid extends from the pole by an arc of  ', gridExtent, ' radians, Pi/2 would cover a hemisphere.'
PRINT *,    'The Source Array is rotated by an angle of ', thetaSM, ' radians.'
!
                ! A4. Set parameters to control the program
!
!   Set the number of sources N. Note that nSrc = N must be a perfect square, 9, 16, 25, ...
NumberOfnSrc = 14 ! The number of values of nSrc processed
nSrcList(1:NumberOfnSrc) =  (/9,16,25,36,49,64,81,100,121,225,256,324,625,900/)
!   Set the nominal region radii RhoNom. Note that the radii are in radians
numberOfRgnRadii = 21   ! The number of nominal region radii processed
RhoNomList(1:numberOfRgnRadii) =   (/1./7,1./6,1./5,1./4,1./3,1./2,1.,3./2,2.,3.,4.,6.,8.,10., &
        & 15.,16.,24.,32.,42.,52.,64./)*2.*Pi/360. ! nominal region radii in radians
PRINT *,    'The list of the numbers of sources evaluated is  ', nSrcList(1:NumberOfnSrc)
PRINT *,    'The nominal region radii in radians are ', RhoNomList(1:numberOfRgnRadii)
PRINT *,    'The nominal region radii in degrees  are ', RhoNomList(1:numberOfRgnRadii)*(360./(2.*Pi))
! The number of random runs to be generated for each case (N, RhoNom)
! Usually, I set nRunMax = 1000.  Then, for each case (N,RhoNom), we have # = kiloRunMax*nRunMax.
nRunMax = 1000  !The number of random runs in one unit
kiloRunMax =  10  !The number of units of random runs for each case (N,RhoNom)
PRINT *,    'The number of cases (N,RhoNom) is ', NumberOfnSrc*numberOfRgnRadii
PRINT *,    'The number of random runs for each case (N,RhoNom) is ', kiloRunMax*nRunMax
PRINT *,    'Combining all cases, the total number of random runs is ', numberOfRgnRadii*kiloRunMax*nRunMax
!
CALL RANDOM_SEED()  ! Needed for pseudo-random number generator
                ! A5.  Construct the Grid, an array of points, uniformly spaced in longitude and latitude
!
! When gridExtent = Pi/2 = 1.57, the points on the sphere form a grid covering one hemisphere.
```

```
! Initially, the grid is constructed centered on the North Pole, for convenience.
! The grid has points separated by a fixed amount, dTheta1,  in longitude and latitude.
! Once set up, the grid is rotated to the positive y-axis, so the grid covers the hemisphere with y > = 0.
! The sample is constructed very near to the y-axis. Thus, the center point of the sample is close to
! yHAT = (0,1,0) and the center of the grid is located at the point (0,1,0).
!
!                    Initially, build the grid centered on the North Pole, (0,0,1)
   n=0   ! n is the counter for the grid points
   j=1   ! j is the counter for latitude
   i=1   ! i is the counter for longitude
DO  WHILE (j < gridExtent/dTheta1)   ! latitudes run from PI/2 to PI/2 - gridExtent
   latH = PI/2. - j*dTheta1 - dTheta1/SQRT(2.)  ! latitudes start at the pole and work away
   DO  WHILE (i < CEILING(2.*Pi*COS(latH)/dTheta1))    ! longitudes run from 0 to 2Pi, spacing depends on latitude
      LonH = i*dTheta1/(COS(latH))    ! longitude
      LonGrid0(n)=LonH        ! save the longitude in a table
      LatGrid0(n)=LatH        ! save the latitude in a table
      n=n+1   ! that grid point is done, move on
      i=i+1    ! that longitude is done, move on
   END DO
      i=1   ! reset longitude index to its initial value
      j=j+1 ! that latitude is done, move on to the next latitude
END DO
   nGrid = n  ! the number of grid points
PRINT *,    'The number of grid points is ', nGrid
!
!   rotation matrix: rotate about the x-axis so that zHAT goes to yHAT, yHAT goes to -zHAT, xHAT fixed
rotZtoY = RESHAPE((/ 1,0,0,0,0,-1,0,1,0 /), SHAPE(rotZtoY))
!   Check the rotation:
      !x=1.
      !y=10.
      !z=100.
! PRINT *, 'rot.(1,10,100) = ', matmul(rotZtoY,(/x,y,z/))    !Check zHAT goes to yHAT, etc.
! rotate the grid
      k=1  ! k is the counter for grid points
DO  WHILE(k<=nGrid)
      r0 = er( LonGrid0(k),LatGrid0(k) )  ! 3D coordinates of the grid point
      r = MATMUL(rotZtoY,r0)              ! 3D coordinates after being rotated z to y
         Rgrid(k,1) = r(1)       ! x-coordinate of the grid point after the grid is rotated z to y
         Rgrid(k,2) = r(2)       ! y-coordinate
         Rgrid(k,3) = r(3)       ! z-coordinate
      LonGrid(k) = LonFromR(r)    ! longitude of the grid point after the rotation
      LatGrid(k) = LatFromR(r)    ! latitude of the grid point after the rotation
         k=k+1   ! grid point k is done, move on to k+1
END DO
!
               ! A6. Construct samples and apply the Hub Test
!
! The samples are square arrays of sources, so the number of sources for a given sample must be a
! perfect square, N = m^2. Currently there are 14 cases of N running from 9 to 900, May 2022.
! The samples are constructed centered very close to the y-axis, near the point (0,1,0) on the sphere.
! The square array that is meant to uniformly cover a circle with a nominal radius, 'squaring the circle.'
! The size of the array is adjusted to approximate a circular area with a nominal radius, the angular
! distance from the center to the edge. There are 21 nominal radii from 1/7 degree to 64 degrees,
! With 14 cases of N, each with 21 different sizes of array, there are 14*21 = 294 samples.
```

```
!
! Select N, one N at a time
DO 2000 iN = 1,NumberOfnSrc     ! index iN picks the number of sources N from the list nSrcList
      nSrc = nSrcList(iN) ! the number of sources, nSrc = N
PRINT *,    'For this set of cases with different RhoNom, the number of sources is nSrc = N = ', nSrc
         rootNsrc = NINT( SQRT( REAL(nSrc)  ) )  ! the integer square root of the number of sources
! PRINT *,    'n^1/2 = ', rootNsrc
!
               ! A6a. Create the square array of sources, 'SM' stands for Source Matrix
!
!   Center the source array a little off the y-axis to avoid resonances with coordinate axes and grid points
CALL RANDOM_NUMBER(r1)  ! a random 3-vector with (x,y,z) values between 0 and 1
! In the following step, the random displacement from yHAT is small, with coordinates between -dTheta/2 and +dTheta/2
      rCenterSrc0 = (/0.,1.,0./) + dTheta1*(r1-(/0.5,0.5,0.5/) )  ! a small displacement from yHAT
      rCenterSrc = rCenterSrc0/SQRT(DOT_PRODUCT(rCenterSrc0,rCenterSrc0)) ! Make it a unit vector
! PRINT *,    'The center of the square array  has x,y,z coordinates ', rCenterSrc
      LonCenterSrc = LonFromR(rCenterSrc) ! longitude of the center of the square array of sources
      LatCenterSrc = LatFromR(rCenterSrc) ! latitude of the center of the square array of sources
! PRINT *,    '(longitude, latitude) of rCenterSrc = ', LonCenterSrc, LatCenterSrc
!
! Apply the Hub Test for each nominal radius, one radius at a time
      kiloRun = 0     ! counter for the number of random run units nRunMax processed
DO 1000 ir = 1,numberOfRgnRadii     ! index ir picks the nominal region radius RhoNom from the list RhoNomList
!Create square array of sources
      aSM = RhoNomList(ir)*((SQRT(2.)+2.)/2.) ! length of a side is about 1.7 times the nominal radius
      daSM = aSM/(rootNsrc-1)             ! The space between neighboring sources along a row or column
!
! Determine the longitudes and latitudes of the sources
      n=1  ! n is the counter for sources
DO 120  j = 1,rootNSrc  ! j is the counter for latitudes of sources
      latS = LatCenterSrc  -((REAL(rootNsrc) + 1.)/2.)*daSM + j*daSM
      DO 100 i = 1, rootNSrc  ! i is the counter for the longitudes of the sources
         LonS = LonCenterSrc +(-((REAL(rootNsrc) + 1.)/2.)*daSM + i*daSM)/(COS(latS))
         LonSource0(n)=LonS      ! save the longitude in a table
         LatSource0(n)=LatS      ! save the latitude in a table
         n=n+1       ! on to the next source
      100  CONTINUE
120  CONTINUE
!
! Rotate the square array about the y-axis by some angle thetaSM, where 'SM' indicates Source Matrix
! The rotation reassures that the square array is out of alignment with the grid.
DO 140 k = 1,nSrc   ! k is the counter for the sources
      r0 = er( LonSource0(k),LatSource0(k) ) ! Initial 3D coordinates of the source
      r = MATMUL(rotAboutY(thetaSM),r0)      ! Final 3D coordinates of the source
      RSource(k,1) = r(1)     ! x-coordinate of the source k after the array is rotated about y
      RSource(k,2) = r(2)     ! y-coordinate of the source k
      RSource(k,3) = r(3)     ! z-coordinate of the source k
      LonSource(k) = LonFromR(r)  ! longitude of source k after the square array is rotated about y
      LatSource(k) = LatFromR(r)  ! latitude  of source k after the square array is rotated about y
      r = eN( LonSource(k),LatSource(k) ) ! 3D components of local North at the source
      NSource(k,1) = r(1)     ! x component of local North at the source
      NSource(k,2) = r(2)     ! y component of local North at the source
      NSource(k,3) = r(3)     ! z component of local North at the source
      r = eE( LonSource(k),LatSource(k) ) ! 3D components of local East at the source
```

```
      ESource(k,1) = r(1)     ! x component of local East at the source
      ESource(k,2) = r(2)     ! y-component
      ESource(k,3) = r(3)     ! z-component
140  CONTINUE
!
! Calculate the RMS radius of the source matrix:
      rSrcSum = (/0.,0.,0./)  ! Initially the sum is zero
DO 160  n=1,nSrc    ! n is the counter for sources
      rSrcSum = rSrcSum + RSource(n,1:3)  ! the sum of the sources' unit radial vectors
160 CONTINUE
      rSrcAve = rSrcSum/SQRT(DOT_PRODUCT(rSrcSum,rSrcSum)) ! Unit vector to sources' center
! The two ways to get the center of the array should agree:
!PRINT *,    'The difference should be small. rSrcAve - rCenterSrc  = ', rSrcAve - rCenterSrc
      RhoRMSsquared = 0.  ! the square of the root-mean-square radius of the source array, initially zero
DO 180 n = 1,nSrc   ! n is the counter for sources
      dotRSrcWithAve=MIN(1.,DOT_PRODUCT(RSource(n,1:3),rSrcAve)) ! dot product of the source with the array's center.
! Note: Since RSource and rSrcAve are unit vectors, the dot product must be less than unity.
! However, when RSource and rSrcAve are the same, imprecision may push the dot product past unity.
      RhoRMSsquared = RhoRMSsquared +((ACOS(dotRSrcWithAve ))**2)/nSrc    ! Sum the squares of the distance
!                           to the center
180 CONTINUE
      RhoRMS = SQRT(RhoRMSsquared) ! root-mean-square distance of sources from the source array's center, rSrcAve
!PRINT*,     'number of sources nSrc, nominal radius RhoNom, RMS radius RhoRMS:', nSrc, RhoNomList(ir)*180/Pi,
!               RhoRMS*180/Pi
!
               ! A6b. Start the Hub Test by calculating the alignment angle function EtaBar at the grid points
!
! For each of the 294 samples, the sources are assigned randomly directed polarization directions.
! Then the function Eta Bar is calculated at the grid points.
! The maximum and minimum values of Eta Bar and their locations on the sphere are collected.
! The results for the samples are collected and saved in files, one file for each sample number N.
! Currently, there are 10,000 records for each sample (N,RhoNom), so 2,940,000 records in total.
! For the purpose of users to check this program, there is an option to reduce the number of cases to a small number.
! To run with 294 samples and 10,000 random runs each see Sec. A3 Settings.
!
! Calculate the unit vector along the cross product of the radial vectors, source i with each grid point j
DO 230 i = 1,nSrc  ! i is the counter for sources
      DO  220 j = 1,nGrid ! j is the counter for grid points
         r0 = RSource(i,1:3) ! 3D radial unit vector to the source i
         r1 = RGrid(j,1:3)   ! 3D radial unit vector to the grid point j
         r2 = Cross(r0,r1)   ! r0xr1, cross product of source and grid point
         r3 = r2/SQRT(DOT_PRODUCT(r2,r2)) ! Unit vector in direction of the cross product
         rSrcxrGrid(i,j,1) = r3(1)   ! x component of unit vector along cross product
         rSrcxrGrid(i,j,2) = r3(2)   ! y component of unit vector along cross product
         rSrcxrGrid(i,j,3) = r3(3)   ! z component of unit vector along cross product
      220  CONTINUE
230  CONTINUE
!
! Make the random runs:
   EtaBarMin = Pi  ! Set the initial minimum value of the alignment angle function. A high value on purpose.
   EtaBarMax = -Pi ! Initial maximum value of the alignment angle function. A low value on purpose.
301 CONTINUE
DO 350   nRun = 1,nRunMax    ! nRun is the counter for random runs
! Assign random polarization directions psi
```

```
      CALL RANDOM_NUMBER(psiSource0) ! populate the array psiSource0 with N random numbers 0 to 1.
      psiSource = Pi*psiSource0      ! N random polarization directions, with values 0 to PI.
! Calculate SxPsi, the unit vector along the cross product of the radial vector to the Source i and its polarization vector Psi
      DO 330 i = 1,nSrc   ! i is the counter for sources
        r = SIN(psiSource(i))*NSource(i,1:3) - COS(psiSource(i))*ESource(i,1:3) ! r is SxPsi, by an easily derived formula
        rSrcxPsiSource(i,1) = r(1)  ! x-component of cross product SxPsi
        rSrcxPsiSource(i,2) = r(2)  ! y-component of cross product SxPsi
        rSrcxPsiSource(i,3) = r(3)  ! z-component of cross product SxPsi
      330  CONTINUE
! Calculate the angle from the center of the source array to each grid point j
      DO 340  j = 1,nGrid  ! j is the counter for grid points
        SourceCenterToH =  ACOS(DOT_PRODUCT(rCenterSrc,rGrid(j,1:3))) ! angle from the center of the sources
!                            to grid point j
        EtaBarMin1 = EtaBarMin  ! dummy parameter to assist with LOGIC
        EtaBarMax1 = EtaBarMax  ! dummy parameter to assist with LOGIC
! Sum the alignment angles Etaij from each source i to the grid point j and divide by N to get the average.
! The average is the alignment angle function EtaBar(j) evaluated at grid point j.
! EtaBar(j) is the fundamental function of the Hub Test.
        sumForjEtaBar = 0.  ! Sum over sources at grid point j, initially zero
        DO  345  i = 1,nSrc ! i is the counter for the sources
          sumForjEtaBar = sumForjEtaBar + ACOS(ABS(DOT_PRODUCT(rSrcxPsiSource(i,1:3),rSrcxrGrid(i,j,1:3))))
! where we sum the angles Etaij
        345 CONTINUE
        jEtaBarToGrid(j) = sumForjEtaBar/nSrc ! find average alignment angle from sources to the grid point j
! Find the extreme values of the alignment angle function EtaBar, the minimum and maximum of the EtaBar(j)
! okMin, okMax are LOGIC conditions to get minimum and maximum EtaBar at any grid point
        okMin = (( jEtaBarToGrid(j) < EtaBarMin ).AND.(SourceCenterToH > minGridCenterToHmin*dTheta1))
!                  where the second clause keeps Hmin away from the sources
        IF (okMin .EQV. .TRUE.)  THEN
        EtaBarMin1 = jEtaBarToGrid(j)   ! replace previous min with new min if the new EtaBar(j) is smaller
        END IF
        IF (okMin .EQV. .TRUE.)  THEN
        jEtaBarMin = (/REAL(j), EtaBarMin1/) ! The combination (j,EtaBarMin) to identify the grid point j and the
!        current min EtaBar
        END IF
        IF (okMin .EQV. .TRUE.)  THEN   ! Logic can be obtuse
        EtaBarMin = EtaBarMin1
        END IF
        okMax = (( jEtaBarToGrid(j) > EtaBarMax ).AND.(SourceCenterToH > minGridCenterToHmax*dTheta1))
!                  where the second clause keeps Hmax away from the sources
        IF (okMax .EQV. .TRUE.)  THEN
        EtaBarMax1 = jEtaBarToGrid(j)   ! replace previous max with new max if the new EtaBar(j) is larger
        END IF
        IF (okMax .EQV. .TRUE.)  THEN
        jEtaBarMax = (/REAL(j), EtaBarMax1/)    ! The combination (j,EtaBarMax) to identify the grid point j
!                    and the current max EtaBar
        END IF
        IF (okMax .EQV. .TRUE.)  THEN   ! Logic can be obtuse
        EtaBarMax = EtaBarMax1
        END IF
      340 CONTINUE
!
            ! A7. Save the results to files, end program
!
! The maximum and minimum of the alignment function Eta Bar are collected, along with the grid
```

```
! point data where the extreme values occur.
! The collected data is written to a file, one file for one value of the number N of sources;
! Thus there is one file for 9 sources, one for 16 sources, etc. Each file contains all the array sizes.
!
! Prepare the output
!PRINT *,    'A random run determines the following quantities that are to be saved:'
!PRINT *,    'nRun, rCenterSrc, RhoRMS, longitude, latitude, unit radial vector to Hmin, minimum alignment angle,'
!PRINT *,    'longitude, latitude, unit radial vector to Hmax, maximum avoidance angle, nSrc, RhoNominal'
!PRINT *,    'Compare rCenterScr, rSrcAve', rCenterSrc(1:3), rSrcAve(1:3)
      OutRunData(nRun,1) = nRun + nRunMax*kiloRun - (ir-1)*nRunMax*kiloRunMax
      OutRunData(nRun,2) =  rCenterSrc(1) ! x coord. rCenter
      OutRunData(nRun,3) =  rCenterSrc(2) ! y coord. rCenter
      OutRunData(nRun,4) =  rCenterSrc(3) ! z coord. rCenter
      OutRunData(nRun,5) = jEtaBarMin(1)  ! ID# for grid point Hmin where EtaBar is Min
      OutRunData(nRun,6) =  RhoRMS        ! RMS radius of source array
      OutRunData(nRun,7) =  -999.         !  Not used
      OutRunData(nRun,8) = lonGrid(jEtaBarMin(1)) ! longitude for Hmin
      OutRunData(nRun,9) =  latGrid(jEtaBarMin(1)) ! latitude for Hmin
      OutRunData(nRun,10) = rGrid(jEtaBarMin(1),1)  ! x-coord of Hmin
      OutRunData(nRun,11) = rGrid(jEtaBarMin(1),2)  ! y-coord of Hmin
      OutRunData(nRun,12) = rGrid(jEtaBarMin(1),3)  ! z-coord of Hmin
      OutRunData(nRun,13) = jEtaBarMin(2)          ! EtaBarMin
      OutRunData(nRun,14) = jEtaBarMax(1)  ! ID# for grid point Hmax where EtaBar is Max
      OutRunData(nRun,15) = -999.         !  Not used
      OutRunData(nRun,16) = -999.         !  Not used
      OutRunData(nRun,17) = lonGrid(jEtaBarMax(1)) ! longitude for Hmax
      OutRunData(nRun,18) = latGrid(jEtaBarMax(1))  ! latitude for Hmax
      OutRunData(nRun,19) = rGrid(jEtaBarMax(1),1)  ! x-coord of Hmax
      OutRunData(nRun,20) = rGrid(jEtaBarMax(1),2)  ! y-coord of Hmax
      OutRunData(nRun,21) = rGrid(jEtaBarMax(1),1)  ! z-coord of Hmax
      OutRunData(nRun,22) = jEtaBarMax(2)          ! EtaBarMax
      OutRunData(nRun,23) = nSrc                   ! number of sources
      OutRunData(nRun,24) = RhoNomList(ir)             ! Nominal radius of the sample array
      EtaBarMin = Pi  ! reset min EtaBar for the next random run
      EtaBarMax = -Pi ! reset max EtaBar for the next random run
350  CONTINUE
!
! Append the random run data to a file
IF(kiloRun < 0.9) THEN              ! Initialize the file to accept the first unit of random runs
   WRITE (file_name,"('2205randomV6N',i0,'.dat')")nSrc  ! The file name includes the number of sources
! PRINT*,"file name is ",trim(file_name)
   OPEN(Unit=5,file=file_name)
   WRITE(5,4000)   TRANSPOSE(OutRunData(1:nRunMax,1:24))
   CLOSE(5)
   ELSE IF(kiloRun > 0.9) THEN         ! Append succeeding units of random runs to the file
      WRITE (file_name,"('2205randomV6N',i0,'.dat')")nSrc  ! The file name includes the number of sources
! PRINT*,"file name is ",trim(file_name)
      OPEN(Unit=5,file=file_name,STATUS='OLD', POSITION='APPEND')
      WRITE(5,4000)   TRANSPOSE(OutRunData(1:nRunMax,1:24))
      CLOSE(5)
END IF
4000 FORMAT(24E16.7,/)   !The FORMAT statement for the output files, 24 real numbers per record
   kiloRun = kiloRun + 1   ! that unit of random runs is done, start the next batch unless following LOGIC says no
IF(kiloRun < ir*kiloRunMax) THEN ! Repeat until done with random runs for RhoNom(ir)
   GoTo 301
```

```
END IF
!
! Print*, 'The nominal radius and the number of random runs done with that radius are ', RhoNomList(ir), nRun
1000 CONTINUE ! one nominal radius processed, move to the next nominal radius if there is one
!
PRINT *,    'The number of random runs generated for this N is',   numberOfRgnRadii*kiloRunMax*nRunMax
2000 CONTINUE ! one number of sources N processed, move to the next N if there is one
!
END PROGRAM RunData
!
                ! A8. External functions
!
! er, radial unit vector to a point on the Celestial Sphere at a given longitude, latitude
FUNCTION er(Lon, lat)   RESULT (w)  ! er, radial unit vector
   IMPLICIT NONE
   REAL, DIMENSION(3) :: w
   REAL, INTENT(IN) ::Lon,lat  ! longitude and latitude
w = (/COS(Lon)*COS(lat),SIN(Lon)*COS(lat),SIN(lat)/)
END FUNCTION
FUNCTION eN(Lon, lat)   RESULT (w)   ! eN, local North at the point on the sphere with a given longitude, latitude
   REAL, DIMENSION(3) :: w
   REAL, INTENT(IN) ::Lon,lat      ! longitude and latitude
w = (/-COS(Lon)*SIN(lat),-SIN(Lon)*SIN(lat),COS(lat)/)
END FUNCTION
FUNCTION eE(Lon, lat)   RESULT (w)  ! eE, local East at the point on the sphere with a given longitude, latitude
   REAL, DIMENSION(3) :: w
   REAL, INTENT(IN) ::Lon,lat      ! longitude and latitude
w = (/-SIN(Lon),COS(Lon),0./)
END FUNCTION
FUNCTION LonFromR (r) RESULT (w)   ! the longitude of the point on the sphere with the given radial vector r
IMPLICIT NONE
REAL :: w
REAL, PARAMETER ::  PI  = 4* ATAN(1.)
REAL, INTENT(IN) :: r(3)        ! the given radial vector
IF ((r(1) > 0.).AND.(r(2) >= 0.)) THEN
  w = ATAN(ABS(r(2)/r(1)))
ELSE IF ((r(1) < 0.) .AND.(r(2) >= 0.)) THEN
  w = PI - ATAN(ABS(r(2)/r(1)))
ELSE IF ((r(1) < 0.).AND.(r(2) < 0.)) THEN
  w = PI + ATAN(ABS(r(2)/r(1)))
ELSE IF((r(1) > 0.).AND.(r(2) < 0.)) THEN
  w = 2.*PI - ATAN(ABS(r(2)/r(1)))
ELSE IF((r(1) == 0.).AND.(r(2) >= 0.)) THEN
  w = PI/2.
ELSE IF((r(1) == 0.).AND.(r(2) < 0.)) THEN
  w = 3*PI/2.
END IF
END FUNCTION LonFromR
FUNCTION LatFromR (r) RESULT (w)   ! the latitude of the point on the sphere with the given radial vector r
IMPLICIT NONE
REAL :: w
REAL, PARAMETER ::  PI  = 4* ATAN(1.)
REAL, INTENT(IN) :: r(3)        ! the given radial vector
IF (SQRT(r(1)**2 + r(2)**2) > 10.**(-4)) THEN
```

```
  w = ATAN(r(3)/(SQRT(r(1)**2 + r(2)**2)))
ELSE IF (Sqrt(r(1)**2 + r(2)**2) < 10.**(-4) ) THEN
  w = SIGN(1.,r(3))*Pi/2. ! sign(1.,r(3))*
END IF
END FUNCTION LatFromR
FUNCTION rotAboutX(theta) RESULT  (w) ! Rotation matrix for a rotation about the x-axis through an angle theta
      REAL, DIMENSION(3,3) :: w
      REAL, INTENT(IN) :: theta      ! rotation angle
      w = RESHAPE((/ 1.,0.,0.,0.,COS(theta),SIN(theta),0.,-SIN(theta),COS(theta) /), shape(w))
  END FUNCTION
FUNCTION rotAboutY(theta) RESULT  (w) ! Rotation matrix for a rotation about the y-axis through an angle theta
      REAL, DIMENSION(3,3) :: w
      REAL, INTENT(IN) :: theta      ! rotation angle
      w = RESHAPE((/ COS(theta),0.,-SIN(theta),0.,1.,0.,SIN(theta),0.,COS(theta) /), shape(w))
  END FUNCTION
FUNCTION rotAboutZ(theta) RESULT  (w) ! Rotation matrix for a rotation about the z-axis through an angle theta
      REAL, DIMENSION(3,3) :: w
      REAL, INTENT(IN) :: theta     ! rotation angle
      w = RESHAPE((/ COS(theta),SIN(theta),0.,-SIN(theta),COS(theta),0.,0.,0.,1. /), shape(w))
  END FUNCTION
FUNCTION cross(r0,r1) RESULT  (w) ! The cross product r0xr1 of two vectors r0 and r1
      REAL, DIMENSION(3) :: w
      REAL, INTENT(IN) ::  r0(3), r1(3)   ! the two given vectors
      w(1) = r0(2)*r1(3) - r0(3)*r1(2)    ! x-component of the cross product
      w(2) = r0(3)*r1(1) - r0(1)*r1(3)    ! y-component of the cross product
      w(3) = r0(1)*r1(2) - r0(2)*r1(1)    ! z-component of the cross product
  END FUNCTION
```

Appendix B. A Computer Program for Fitting Random Run Distributions

CONTENTS

Preface

Appendix A created samples with randomly directed polarization directions, transverse vectors. The Hub Test was applied to a large number of times. In Appendix B, the results of the Hub Test from App. A are analyzed, resulting in formulas for the distributions of the App. A results. The parameters of the distribution formulas are saved and constitute a reference ‘Library’ that contains information sufficient to reconstruct the distributions. Furthermore, the Library can generate probability distributions for the Hub Test results from randomly directed samples. Such information is convenient for deducing the significance of Hub Test results from observed samples.

The following is a Mathematica notebook. To get a ready-to-run version follow one of the links in Ref. 10.

B1. Preliminary

Imagine the points are plotted on the Celestial sphere and we are looking down on the sphere from the outside. See Figs. 1 and 2, for example.

```
The date and time that this statement was evaluated:
```
Wed 22 Jun 2022 07:12:35 GMT−4

```
The computer time expended so far is 1.344 seconds.

The computer memory in use is 95 386 696 bytes.
```

Definitions:

mean the arithmetic average of a set of numbers, $\frac{1}{N}\sum_{i=1}^{N} n_i$

stanDev the standard deviation. Given a set of N numbers n_i with mean value m, the standard deviation is $\left(\frac{1}{N}\sum_{i=1}^{N}(n_i - m)^2\right)^{1/2}$, the square root of the average of the squares of the differences of the numbers with the mean. Note that we divide by N to get the average of the deviations squared.

In[4]:=
```
(*The 'home directory' has the notebook. *)
homeDirectory = NotebookDirectory[];
```

In[5]:=
```
mean[data_] := (1 / Length[data]) Sum[data[[i4]], {i4, Length[data]}];
(* arithmetic average, Eq. (B1) *)
stanDev[data_] := ((1 / Length[data]) Sum[(data[[i5]] - mean[data])^2, {i5, Length[data]}])^1/2
(*standard deviation, Eq. (B2)*)
```

B2. Upload the FORTRAN runData files

Upload and analyze the files saved in Appendix A one at a time. Each file contains the random run data, runData, for one value of N, the number of sources.
For example, "2205randomV5N9.dat" has the runData for $N = 9$ sources, including all 21 region radii.

Definitions:
fortranRunData: raw FORTRAN output
1. run # 2., 3., 4. x,y,z coordinates of the center of the sample 5. grid point ID for Hmin (not useful, unknown grid) 6. r.m.s radius of region 7. not used (-999.) 8., 9. longitude, latitude of Hmin 10., 11., 12. x,y,z coords. for Hmin 13. $\overline{\eta}_{\min}$ 14. grid point ID for Hmax (not useful, unknown grid) 15., 16. not used (-999.) 17.,18. longitude, latitude of Hmax 19., 20., 21. x,y,z coords. for Hmax 22. $\overline{\eta}_{\max}$ 23. nSrc, the number of sources 24. ρNom, nominal region radius

runData0: raw FORTRAN output organized into records each 24 items long

runData: runData0 reorganized for the Mathematica program
1. nRun 2. $\hat{r}$ at Region Center 3a. grid data for Hmin 3b. $\overline{\eta}_{\min}$ 4a. grid data for Hmax 4b. $\overline{\eta}_{\max}$ 5. nSrc 6. radius ρNom

runData in detail:
1. nRun 2. rCenterSrc 3a. grid data: {gridIDHmin, ρRMS, notUsed1, longHmin, latHmin, rHmin}, 3b. $\overline{\eta}_{\min}$ 4a. grid data: {gridIDHmax, notUsed2, notUsed3, longHmax, latHmax, rHmax}, 4b. $\overline{\eta}_{\max}$ 5. nSrc 6. nominal radius ρNom

nRun run #, from 0 to 10,000 for each case (nSrc, ρNom)
rCenterSrc 3D rectangular coords for the average of the source locations, the center of the sample
gridIDHmin ID # for the grid point at Hmin, meaningless without the grid
ρRMS root mean square radius of the sources' locations from the sample center

notUsed1 another unused real number, -999.
longHmin, latHmin longitude, latitude of the hub *Hmin* where $\overline{\eta}(H)$ is the minimum $\overline{\eta}_{\min}$
rHmin 3D rectangular coords of Hmin
ηBarmin the minimum $\overline{\eta}_{\min}$ of the alignment angle function $\overline{\eta}(H)$
gridIDHmax ID # for the grid point at Hmax, meaningless without the grid
ρRMS root mean square radius of the sources' locations from the sample center
notUsed2,3 more unused real number slots, -999.
longHmax, latHmax longitude, latitude of the hub Hmax where $\overline{\eta}(H)$ is the maximum $\overline{\eta}_{\max}$
rHmax 3D rectangular coords of Hmax
ηBarmax the maximum $\overline{\eta}_{\max}$ of the alignment angle function $\overline{\eta}(H)$
nSrc the number of sources in the sample
ρNom the nominal radius of the sample region

In[7]:=
```
SetDirectory[homeDirectory];
(*fortranRunData= ReadList["2205randomV5N9.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N16.dat",Real];*)
fortranRunData = ReadList["2205randomV5N25.dat", Real];(*Selected*)
(*fortranRunData= ReadList["2205randomV5N36.dat",Real];*)
(* fortranRunData= ReadList["2205randomV5N49.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N64.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N81.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N100.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N121.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N225.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N256.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N324.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N625.dat",Real];*)
(*fortranRunData= ReadList["2205randomV5N900.dat",Real];*)
```

```
The uploaded data file contains 5040000 numbers in 210000 records, each 24 numbers long.
```

B3. Organize the data for the Mathematica code

Definitions:
fortranRunData, runData0:
1. nRun 2,3,4. rCenterSrc x,y,z 5. gridID for Hmin (not useful, unknown grid) 6. ρRMS 7. -999. (notUsed1) 8., 9. longitude, latitude of Hmin 10,11,12. rHmin x,y,z 13. $\overline{\eta}_{\min}$ 14. gridID for Hmin (not useful, unknown grid) 15., 16. -999. (notUsed2,3) 17., 18. longitude, latitude of Hmin 19,20,21. rHmin x,y,z 22. $\overline{\eta}_{\max}$ 23. nSrc 24. nominal radius ρNom

In[10]:=
```
runData0 = Partition[fortranRunData, 24];(*Partition the data into 24-item records*)
Length[runData0];
Print["The first record: ", runData0[[1]]]
Print["The last record: ", runData0[[-1]]]
```

```
The first record: {1., 0.0000954966, 0.999984, -0.00568294, 37., 0.00212949, -999.,
  1.62929, -0.0276712, -0.0584373, 0.997908, -0.0276677, 0.703947, 104., -999., -999.,
  1.60266, 0.0943891, -0.0317185, 0.995043, -0.0317185, 0.860122, 25., 0.00249333}

The last record: {10000., 0.0000954966, 0.999984, -0.00568294, 8822., 1.04921,
  -999., 0.619663, 0.101757, 0.809863, 0.577757, 0.101581, 0.637981, 8722., -999.,
  -999., 2.30459, 0.679356, -0.521005, 0.577757, -0.521005, 0.982946, 25., 1.11701}
```

```
(*Some time ago, and not with the current data files,
MMA would not read the following number correctly. I am not sure why,
cosmic rays? The following statement fixed it. If not all the data in a file uploads,
you may need to search for similarly offending numbers.*)
(*runData0[[1,6]] = -999.*)
```

In[15]:=
```
nRun = Table[runData0[[n, 1]], {n, Length[runData0]}];(*run #, recycles for each ρNom*)
rCenterSrc = Table[{runData0[[n, 2]], runData0[[n, 3]], runData0[[n, 4]]}, {n, Length[runData0]}];
(*x,y,z coords. sample center*)
gridIDHmin = Table[runData0[[n, 5]], {n, Length[runData0]}];(*not used*)
ρRMS = Table[runData0[[n, 6]], {n, Length[runData0]}];
(*rms distance sources to sample center*)
notUsed1 = Table[runData0[[n, 7]], {n, Length[runData0]}];
longHmin = Table[runData0[[n, 8]], {n, Length[runData0]}];(*longitude Hmin*)
latHmin = Table[runData0[[n, 9]], {n, Length[runData0]}];(*latitude Hmin*)
rHmin = Table[{runData0[[n, 10]], runData0[[n, 11]], runData0[[n, 12]]}, {n, Length[runData0]}];
(*x,y,z Hmin*)
ηBarmin = Table[runData0[[n, 13]], {n, Length[runData0]}];(*the minimum η̄min*)
```

In[24]:=
```
gridIDHmax = Table[runData0[[n, 14]], {n, Length[runData0]}];(*not used*)
notUsed2 = Table[runData0[[n, 15]], {n, Length[runData0]}];
notUsed3 = Table[runData0[[n, 16]], {n, Length[runData0]}];
longHmax = Table[runData0[[n, 17]], {n, Length[runData0]}];(*longitude Hmax*)
latHmax = Table[runData0[[n, 18]], {n, Length[runData0]}];(*latitude Hmax*)
rHmax = Table[{runData0[[n, 19]], runData0[[n, 20]], runData0[[n, 21]]}, {n, Length[runData0]}];
(*x,y,z Hmax*)
ηBarmax = Table[runData0[[n, 22]], {n, Length[runData0]}];(*the maximum η̄max*)
```

In[31]:=
```
nSrc = Table[runData0[[n, 23]], {n, Length[runData0]}];
(*the number of sources in the sample*)
ρNom = Table[runData0[[n, 24]], {n, Length[runData0]}];
(*the nominal radius of the sample region*)
```

runData same order as runData0, but reorganized for the Mathematica program
1. nRun 2. $\hat{r}$ at Region Center 3a. grid data for Hmin 3b. $\overline{\eta}_{\min}$ 4a. grid data for Hmax 4b. $\overline{\eta}_{\max}$ 5. nSrc 6. radius ρNom
In detail:
1. nRun 2. rCenterSrc 3a. grid data: {gridIDHmin, ρRMS, notUsed1, longHmin, latHmin, rHmin}, 3b. $\overline{\eta}_{\min}$ 4a. grid data: {gridIDHmax, notUsed2, notUsed3, longHmax, latHmax, rHmax}, 4b. $\overline{\eta}_{\max}$ 5. nSrc 6. nominal radius ρNom

In[33]:=
```
(*Reorganize the runData for the Mathematica code:*)
runData = Table[{ nRun[[n]], rCenterSrc[[n]], {{gridIDHmin[[n]],
       ρRMS[[n]], notUsed1[[n]], longHmin[[n]], latHmin[[n]], rHmin[[n]]}, ηBarmin[[n]]},
     {{gridIDHmax[[n]], notUsed2[[n]], notUsed3[[n]], longHmax[[n]], latHmax[[n]], rHmax[[n]]},
      ηBarmax[[n]]}, nSrc[[n]], ρNom[[n]] }, {n, Length[runData0]}];
runData[[1]];
```

B4 Investigate some of the data

Definitions:

allRgnRadii the list of nominal region radii ρNom for the samples, in degrees

runDataRgn[ir] a collection of all runData for the irth nominal radius ρNom. Recall that we consider only one nSrc at a time since only one nSrc data is uploaded at a time.

nRunDataRgn[ir] the number of records with the irth nominal radius ρNom

ratios root mean square radius to nominal radius ρRMS/ρNom

aveRATIO (min ratio + max ratio)/2, for plotting purposes only

flatPlaneRatio ρRMS/ρNom on a flat 2D plane, derived elsewhere, see Sec. A6a. for the construction on the sphere

In[35]:=
```
allRgnRadii = Union[Table[runData[[n, -1]], {n, Length[runData]}]] (360./(2. π));
(*the nominal region radii ρNom in degrees*)
```

```
Nominal Radii. Check for anomolies. Maybe two are nearly equal.

There are 21 different nominal radii ρNom in the runData table for N = 25. sources.

The nominal radii in degrees are {0.142857, 0.166667, 0.2, 0.25,
  0.333333, 0.5, 1., 1.5, 2., 3., 4., 6., 8., 10., 15., 16., 24., 32., 42., 52., 64.}
```

In[39]:=
```
Union[Table[{"Nom: ", ρNom[[n]] (360./(2. π)),
    " degrees, ratio RMS/Nom: ", ρRMS[[n]] / ρNom[[n]]}, {n, Length[ρNom]}]];
ratios = Union[Table[ρRMS[[n]] / ρNom[[n]], {n, Length[ρNom]}]];(*ρRMS/ρNom*)
aveRATIO = (Min[ratios] + Max[ratios]) / 2.;(*For plotting purposes.*)
flatPlaneRatio = ((1. + 2.^-1/2)/(6.^1/2)) ((nSrc[[1]]^1/2 + 1)/(nSrc[[1]]^1/2 - 1))^1/2 ; (*Eq. (B3),
like ratios = ρRMS/ρNom above, but on a flat 2D plane *)
```

In[43]:=
```
lpRMSStoNom = Show[{ListPlot[
    {Union[Table[{Log[10, ρNom[[n]] (360./(2. π))], ρRMS[[n]] / ρNom[[n]]}, {n, Length[ρNom]}]]},
   PlotStyle → {{PointSize[Medium], Black}, {PointSize[Medium], Green}},
   PlotStyle → PointSize[Medium], FrameTicks → Automatic,
   PlotLabel → " Ratio of RMS radius to nominal radius", FrameLabel →
    {"log10(ρNom)", "ρRMS/ρNom "}, GridLines → Automatic, Frame → True, Axes → False],
  Graphics[{{Green, Line[{{Log[10, ρNom[[1]] (360./(2. π))] - 0.1, flatPlaneRatio},
     {Log[10, ρNom[[Length[ρNom]]] (360./(2. π))] + 0.1, flatPlaneRatio}}], Black,
   Text[StyleForm[ "N = ", FontSize → 12, FontWeight → "Plain"], {0., aveRATIO}],
   Text[StyleForm[ ToString[nSrc[[1]], InputForm, NumberMarks → False],
     FontSize → 12, FontWeight → "Plain"], {0.3, aveRATIO}]}]}];
```

Out[44]=

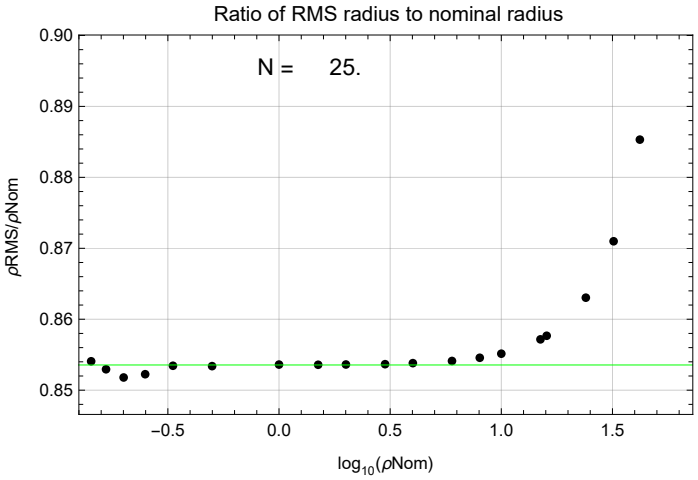

Figure B1. The ratio of the RMS radius to the nominal radius of the sample. See Fig. 2. The ratio depends on the size of the array, but only by a few percent. The green line is the ratio for square arrays constructed on a flat plane, without any distortions due to the curvature of the sphere and without the small random relocations of the array's center in Sec. A6a.

In[45]:=
```
(*The table runDataRgn[ir] is a collection
 of all runData for the ir^th nominal radius ρNom.*)
Table[runDataRgn[ir] = {}, {ir, Length[allRgnRadii]}];
For[ir = 1, ir ≤ Length[allRgnRadii], ir++,
 For[j = 1, j ≤ Length[runData], j++, If[(ρNom[[j]] - allRgnRadii[[ir]] (2. π)/360.)^2 < 10.^-10,
   AppendTo[runDataRgn[ir], runData[[j]]]]]]
nRunDataRgn[ir_] := Length[runDataRgn[ir]]
 (*the number of runData records that have the ir^th nominal radius ρNom*)
```

B5. Fit the random run distributions.

The fits compress the data. By fitting functions to the distributions of random run quantities $\overline{\eta}_{\min}$ and $\overline{\eta}_{\max}$ and keeping just the mean and standard deviation of other quantities, we keep the essence of the random run results without keeping their bulk. All the many runs downloaded in Sec. B2 for a specific number of sources N and a specific radius ρNom reduce to the handful of numbers needed to recreate the distributions from the fitting function formulas.

Definitions

rCenter[ir] $\hat{r}$ at Region Center
nSrcRgn[ir] number of sources
ρRgn[ir] nominal radius ρNom
ηBarMinData[ir], ηBarMaxData[ir] tables of $\overline{\eta}_{\min}$, $\overline{\eta}_{\max}$ for samples with nominal radius ρRgn[ir]
rHminT[ir], rHmaxT[ir] tables of radial unit vectors to Hmin, Hmax
sortηBarMin[ir], sortηBarMax[ir] tables of $\overline{\eta}_{\min}$, $\overline{\eta}_{\max}$, smallest first
η0B[ir], η0Bmax[ir] Guess the peak for the step-Gaussians
σB[ir], σBmax[ir] Guess the half-widthGuess the half-width
hl0[ir], hlMax0[ir] histograms of $\overline{\eta}_{\min}$, $\overline{\eta}_{\max}$ values
hl[ir], hlMax[ir] assign histogram heights to midpoints of bins
nlmB[ir], nlmBMax[ir] step-Gaussian fits to histograms of $\overline{\eta}_{\min}$, $\overline{\eta}_{\max}$ values
amin[ir], amax[ir] amplitudes for the fits
bmin[ir], bmax[ir] half-widths for the fits, 'b' becomes σ
x0min[ir], x0max[ir] value of x at the peaks of the fits, 'x0' becomes η0
damin[ir],dbmin[ir],dx0min[ir], damax[ir],dbmax[ir],dx0max[ir] standard errors calculated for the fits
anglerHminToCenter[ir], anglerHmaxToCenter[ir] tables of arc distance from hubs Hmin, Hmax to sample center
θrHminToCenter[ir], θrHmaxToCenter[ir], average (mean) arc distance hubs to sample center
$\sigma\theta$rHminToCenter[ir], $\sigma\theta$rHmaxToCenter[ir] standard deviation of those arc distances

fitData output data produced by Appendix B

fitData0, fitData Parameters of the alignment (min) and avoidance (max) random run distributions. Angles in radians.
1a. nSrcRgn[ir] Number of sources 1b. ρRgn[ir] Nominal sample radius 1c. ρRMS RMS sample radius 1d. nRunDataRgn[ir] number of random runs
2a. x0min[ir] (η0min) peak $\overline{\eta}_{\min}$ distribution 2b. dx0min[ir] standard error for η0
3a. bmin[ir] (σmin) half-width $\overline{\eta}_{\min}$ distr. 3b. dbmin[ir] standard error for σ
4a. amin[ir] amplitude of $\overline{\eta}_{\min}$ distribution 4b. damin[ir] standard error for amplitude
5a. x0max[ir] (η0max) peak $\overline{\eta}_{\max}$ distribution 5b. dx0max[ir] standard error for η0
6a. bmax[ir] (σmax) half-width $\overline{\eta}_{\max}$ distr. 6b. dbmax[ir] standard error for σ
7a. amax[ir] amplitude of $\overline{\eta}_{\max}$ distribution 7b. damax[ir] standard error for amplitude

In[48]:=
```
fitData = {}; t1 = TimeUsed[];
For[ir = 1, ir ≤ Length[allRgnRadii], ir++,
 rCenter[ir] = runDataRgn[ir][[1, 2]](*r̂ at Region Center*);
 nSrcRgn[ir] = runDataRgn[ir][[1, -2]](*number of sources*);
 ρRgn[ir] = runDataRgn[ir][[1, -1]](*nominal radius ρNom*);
 ηBarMinData[ir] = Table[runDataRgn[ir][[i1, 3, 2]], {i1, Length[runDataRgn[ir]]}]
 (* η̄min table*);
 ηBarMaxData[ir] = Table[runDataRgn[ir][[i1, 4, 2]], {i1, Length[runDataRgn[ir]]}]
 (* η̄max table*);
 rHminT[ir] = Table[ runDataRgn[ir][[i1, 3, 1, 6]] , {i1, Length[runDataRgn[ir]]}]
 (*radial unit vector to Hmin*);
 rHmaxT[ir] = Table[ runDataRgn[ir][[i1, 4, 1, 6]] , {i1, Length[runDataRgn[ir]]}]
```

```
(*radial unit vector to Hmax*);
sortηBarMin[ir] = Sort[ηBarMinData[ir]](* table of η̄min, smallest first*);
η0B[ir] = mean[ηBarMinData[ir] ] (*Guess the peak for the step-Gaussian. *);
σB[ir] = stanDev[ηBarMinData[ir] ](*Guess the half-width.*);
hl0[ir] = HistogramList[sortηBarMin[ir],
  {η0B[ir] - 5 σB[ir], η0B[ir] + 5 σB[ir], 0.4 σB[ir]}](*histogram of η̄min values*);
hl[ir] = Table[{(1 / 2) (hl0[ir][[1, i1]] + hl0[ir][[1, i1 + 1]]), hl0[ir][[2, i1]]},
  {i1, Length[ hl0[ir][[2]] ]}](*assign histogram heights to midpoints of bins*);
nlmB[ir] = NonlinearModelFit[hl[ir], {a (1 + ℯ^4 ((x-x0-b)/b))^-1 Exp[-1/2. ((x - x0)/b)^2], b > 0},
  {{a, nRunDataRgn[ir]/12}, {b, σB[ir]}, {x0, η0B[ir]}}, x]
(*Eq. (B4)*)(*step-Gaussian fit to histogram of η̄min values, x is η̄min*);
{amin[ir], bmin[ir], x0min[ir]} = {a, b, x0} /. nlmB[ir]["BestFitParameters"]
(*parameters of the fit*);
{damin[ir], dbmin[ir], dx0min[ir]} = nlmB[ir]["ParameterErrors"]
(*standard errors of parameters*);
sortηBarMax[ir] = Sort[ηBarMaxData[ir]](*table of η̄max, smallest first*);
η0Bmax[ir] = mean[ηBarMaxData[ir] ]; (*Guess the mean for the step-Gaussian. *)
σBmax[ir] = stanDev[ηBarMaxData[ir] ]; (*Guess the half-width.*)
hlMax0[ir] = HistogramList[sortηBarMax[ir], {η0Bmax[ir] - 5 σBmax[ir],
    η0Bmax[ir] + 5 σBmax[ir], 0.4 σBmax[ir]}](*histogram of η̄max values*);
hlMax[ir] = Table[{(1 / 2) (hlMax0[ir][[1, i1]] + hlMax0[ir][[1, i1 + 1]]), hlMax0[ir][[2, i1]]},
  {i1, Length[ hlMax0[ir][[2]] ]}](*assign histogram heights to midpoints of bins*);
nlmBMax[ir] = NonlinearModelFit[hlMax[ir], {a (1 + ℯ^-4 ((x-x0+b)/b))^-1 Exp[-1/2. ((x - x0)/b)^2], b > 0},
  {{a, nRunDataRgn[ir]/12}, {b, σBmax[ir]}, {x0, η0Bmax[ir]}}, x]
(*Eq. (B5)*)(*step-Gaussian fit to histogram of η̄max values, x is η̄max*);
{amax[ir], bmax[ir], x0max[ir]} = {a, b, x0} /. nlmBMax[ir]["BestFitParameters"]
(*parameters of the fit*);
{damax[ir], dbmax[ir], dx0max[ir]} = nlmBMax[ir]["ParameterErrors"]
(*standard errors of parameters*);
anglerHminToCenter[ir] =
 Table[ArcCos[Abs[rHminT[ir][[i]].rCenter[ir]] - 0.00001], {i, Length[rHminT[ir]]}]
 (*arc length from Hmin to sample center*);
ϴrHminToCenter[ir] = mean[anglerHminToCenter[ir]](*average arc length*);
σϴrHminToCenter[ir] = stanDev[anglerHminToCenter[ir]]
(*standard deviation of the arc lengths*);
anglerHmaxToCenter[ir] =
 Table[ArcCos[Abs[rHmaxT[ir][[i]].rCenter[ir]] - 0.00001], {i, Length[rHmaxT[ir]]}]
 (*arc length from Hmax to sample center*);
ϴrHmaxToCenter[ir] = mean[anglerHmaxToCenter[ir]](*average arc length*);
σϴrHmaxToCenter[ir] = stanDev[anglerHmaxToCenter[ir]]
(*standard deviation of the arc lengths*);
AppendTo[fitData, {{nSrcRgn[ir],
```

```
    ρRgn[ir], runDataRgn[ir][[1, 3, 1, 2]](*ρRMS*), nRunDataRgn[ir]},
   {x0min[ir], dx0min[ir]}, {bmin[ir], dbmin[ir]},
   {amin[ir], damin[ir]}, {x0max[ir], dx0max[ir]}, {bmax[ir], dbmax[ir]},
   {amax[ir], damax[ir]}, {σθrHminToCenter[ir],
    θrHminToCenter[ir]}, {σθrHmaxToCenter[ir],
    θrHmaxToCenter[ir]}}](*collect output data in fitData table*)
]
t2 = TimeUsed[];
```

FittedModel: The property values {ParameterErrors} assume an unconstrained model. The results for these properties may not be valid, particularly if the fitted parameters are near a constraint boundary.

FittedModel: The property values {ParameterErrors} assume an unconstrained model. The results for these properties may not be valid, particularly if the fitted parameters are near a constraint boundary.

FittedModel: The property values {ParameterErrors} assume an unconstrained model. The results for these properties may not be valid, particularly if the fitted parameters are near a constraint boundary.

General: Further output of FittedModel::constr will be suppressed during this calculation.

```
The computer time needed to fit the data is 114.234 seconds.
```

```
The large file uploaded in Sec. B2 for N sources, with N =
 25,  and for the collection of regions with 21 radii ρNom yields
 21 records in the fitData table, one for each ρNom.
```

B6. Build the Library

Add the fitData records from the previous section to the Library.

Definitions:

sortfitData	fitData sorted with smallest number of sources first, then with smallest nominal radii
fitData0	pre-existing fitData table

In[54]:=
```
sortfitData = Sort[fitData];(*just in case it wasn't sorted*)
```

In[55]:=
```
Print["The number of sources being considered now is N = ", Round[nSrcRgn[1]], "."]
Print[
 "Some of the fitData records calculated above for N sources and various ρNom are: "]
Table[Print[sortfitData[[i]]], {i, 1, Length[sortfitData], 4}];
```

The number of sources being considered now is $N = 25$.

Some of the fitData records calculated above for N sources and various ρNom are:

```
{{25., 0.00249333, 0.00212949, 10000}, {0.685476, 0.000866607},
 {0.0689418, 0.0010357}, {1601.72, 21.5285}, {0.885171, 0.000895432},
 {0.0688858, 0.00107016}, {1602.84, 22.2783}, {0.216135, 0.150129}, {0.203678, 0.145209}}

{{25., 0.00581776, 0.00496513, 10000}, {0.680116, 0.00101775},
 {0.0682995, 0.00121633}, {1611.86, 25.6827}, {0.890708, 0.000945214},
 {0.0682551, 0.00112962}, {1616.04, 23.9294}, {0.188003, 0.122187}, {0.183158, 0.119692}}

{{25., 0.0349066, 0.029797, 10000}, {0.634422, 0.000634549},
 {0.0610906, 0.000758281}, {1633.24, 18.138}, {0.936116, 0.000499395},
 {0.0613099, 0.00059678}, {1627.95, 14.1777}, {0.190864, 0.122616}, {0.19543, 0.124122}}

{{25., 0.139626, 0.11932, 10000}, {0.60211, 0.000438061},
 {0.058819, 0.00052351}, {1599.86, 12.7396}, {0.968668, 0.000562113},
 {0.0591053, 0.000671757}, {1613.15, 16.4028}, {0.25102, 0.235553}, {0.248295, 0.235192}}

{{25., 0.418879, 0.361509, 10000}, {0.596685, 0.000424447},
 {0.0588581, 0.000507238}, {1612.52, 12.4328}, {0.973158, 0.000518576},
 {0.0584868, 0.000619716}, {1617.4, 15.3327}, {0.305708, 0.513204}, {0.310163, 0.520224}}

{{25., 1.11701, 1.04921, 10000}, {0.597052, 0.000334124},
 {0.0578935, 0.000399293}, {1610.29, 9.93663}, {0.973337, 0.000506434},
 {0.0565756, 0.000605192}, {1631.67, 15.6159}, {0.3789, 0.991165}, {0.378232, 0.990255}}
```

In[58]:=
```
(*Start a new Library or update an existing Library*)
SetDirectory[homeDirectory];
(*Start a new Library:*)fitData0 = {};
(*Update an existing Library:*)
(*fitData0 = Get["NEW20220618fitData21Rgns14Nsv5NEW.dat"]; *)
```

The fitData0 table initially contains 0 records.

In[61]:=
```
For[i = 1, i ≤ Length[fitData], i++, AppendTo[fitData0, sortfitData[[i]]]
(*Add the records calculated above.*)
```

After appending the fitData table calculated above, the fitData0 table now contains 21 records.

In[63]:=
```
SetDirectory[homeDirectory];
(*Save the fitData0 table to the Library file*)
(*Put[fitData0,"NEW20220618fitData21Rgns14Nsv5NEW.dat"]*)
```

B7. Display some of the distributions

In[64]:=
```
lpEtaBarMinMaxHistograms =
  Table[showNLMB[ir] = Show[{Histogram[sortηBarMin[ir] (360./(2. π))], {η0B[ir] - 5 σB[ir],
      η0B[ir] + 5 σB[ir], 0.4 σB[ir]} (360./(2. π)), FrameTicks → { {Automatic, Automatic} ,
      {{{5, 5 °}, {10, 10 °}, {15, 15 °}, {20, 20 °}, {25, 25 °}, {30, 30 °}, {35, 35 °},
        {40, 40 °}, {45, 45 °}, {50, 50 °}, {55, 55 °}}, Table[{j, ""}, {j, 0, 55, 5}] }},
     PlotLabel → allRgnRadii[[ir]] "° (ρNom)", FrameLabel →
      {"η̄min", "ΔR", allRgnRadii[[ir]] "°"}, GridLines → Automatic,
     Frame → True], Plot[Normal[nlmB[ir]] /. {x → y (2. π/360.)},
     {y, (η0B[ir] - 5 σB[ir]) (360./(2. π)), (η0B[ir] + 5 σB[ir]) (360./(2. π))}],
    ListPlot[Table[{hl[ir][[i, 1]] (360./(2. π)), hl[ir][[i, 2]]}, {i, Length[hl[ir] ]}]],
    Graphics[{Text[StyleForm[ "N = ", FontSize → 8, FontWeight → "Plain"],
      {(η0B[ir] - 4. σB[ir]) (360./(2. π)), 1200.}],
     Text[StyleForm[ ToString[Round[nSrc[[1]]], InputForm, NumberMarks → False],
       FontSize → 8, FontWeight → "Plain"], {(η0B[ir] - 3.0 σB[ir]) (360./(2. π)), 1200.}],
     Text[StyleForm[ "R = ", FontSize → 8, FontWeight → "Plain"],
      {(η0B[ir] - 4. σB[ir]) (360./(2. π)), 1000.}],
     Text[StyleForm[ ToString[Round[Length[sortηBarMin[ir]]], InputForm,
        NumberMarks → False], FontSize → 8, FontWeight → "Plain"],
      {(η0B[ir] - 2.6 σB[ir]) (360./(2. π)), 1000.}]}]}], {ir, 1, Length[allRgnRadii], 4}];
```

Out[65]=

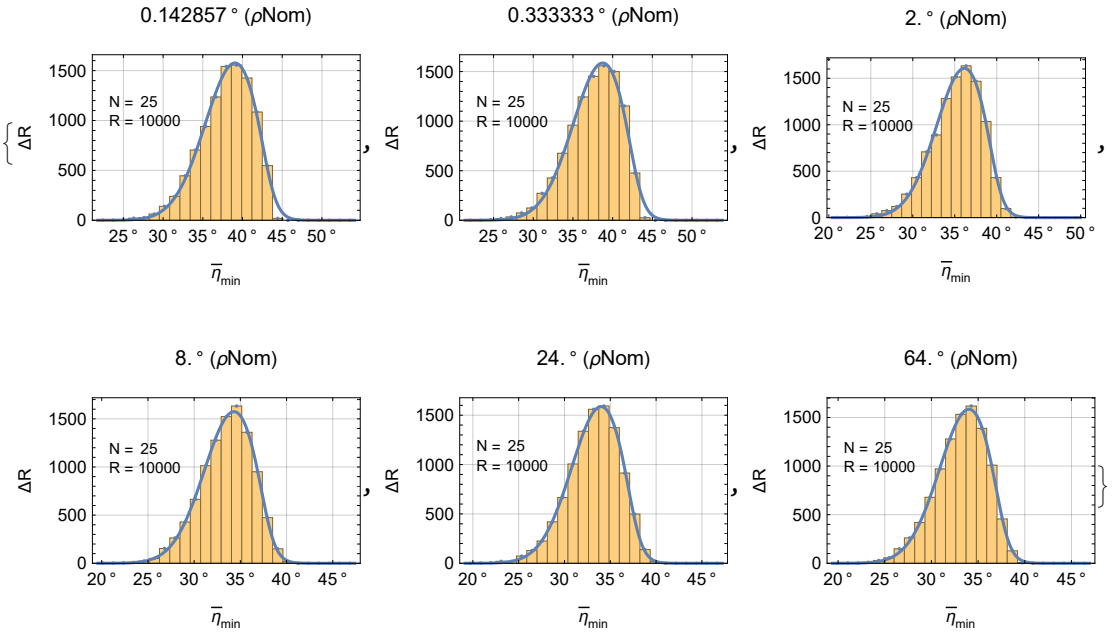

Figure B2. Some of the $\overline{\eta}_{min}$ histograms for the number N of samples currently being treated, N = 25 sources. The solid curves are the Step–Gaussian fits.
The plots are labelled by the nominal radii of each sample's region.

Out[67]=

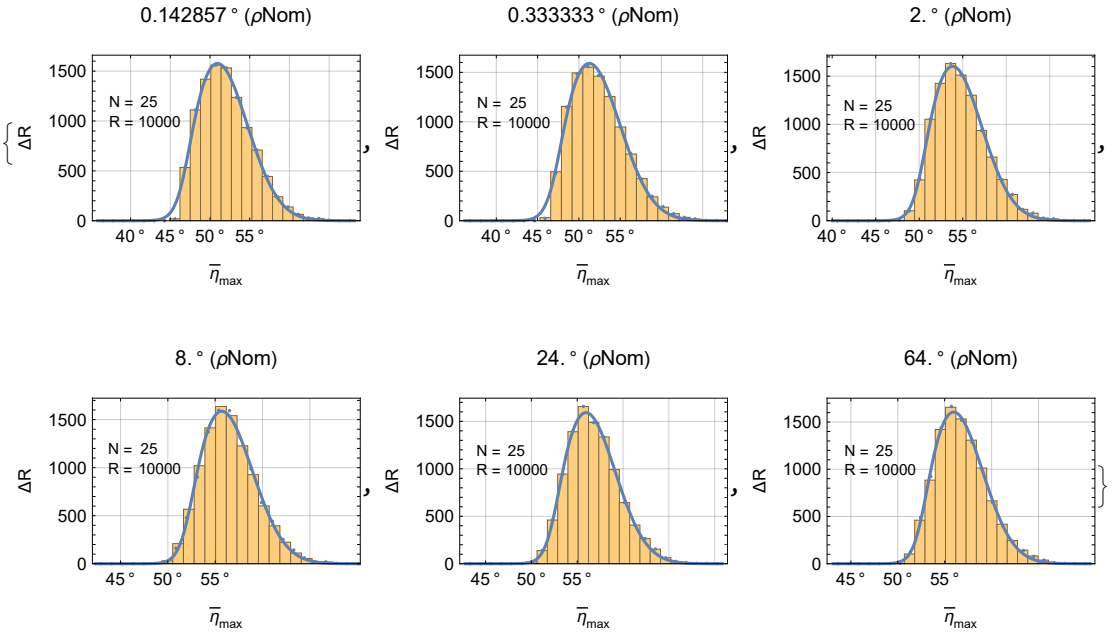

Figure B3. Some of the $\overline{\eta}_{max}$ histograms for the number N of samples currently being treated, N = 25 sources. The solid curves are the Step–Gaussian fits.
Note the shift of a few degrees from the lowest ρNom to the largest.

In[69]:=
```
ir = 7;
Show[{Histogram[sortηBarMin[ir] (360./(2. π)),
    {η0B[ir] - 5 σB[ir], η0B[ir] + 5 σB[ir], 0.4 σB[ir]} (360./(2. π)),
    FrameTicks → { {Automatic, Automatic} ,
      {{{5, 5 °}, {10, 10 °}, {15, 15 °}, {20, 20 °}, {25, 25 °}, {30, 30 °}, {35, 35 °},
        {40, 40 °}, {45, 45 °}, {50, 50 °}, {55, 55 °}}, Table[{j, ""}, {j, 0, 55, 5}] }},
    PlotLabel → allRgnRadii[[ir]] "° = ρNom", FrameLabel → {"η̄min", "ΔR", allRgnRadii[[ir]] "°"},
    GridLines → Automatic, Frame → True], Plot[Normal[nlmB[ir]] /. {x → y (2. π)/360.},
    {y, (η0B[ir] - 5 σB[ir]) (360./(2. π)), (η0B[ir] + 5 σB[ir]) (360./(2. π))}],
  ListPlot[Table[{hl[ir][[i, 1]] (360./(2. π)), hl[ir][[i, 2]]}, {i, Length[hl[ir] ]}]],
  Graphics[{Text[StyleForm[ "N = ", FontSize → 12, FontWeight → "Plain"],
     {(η0B[ir] - 4. σB[ir]) (360./(2. π)), 1200.}],
    Text[StyleForm[ ToString[Round[nSrc[[1]]], InputForm, NumberMarks → False],
      FontSize → 12, FontWeight → "Plain"], {(η0B[ir] - 3.0 σB[ir]) (360./(2. π)), 1200.}],
    Text[StyleForm[ "R = ", FontSize → 12, FontWeight → "Plain"],
     {(η0B[ir] - 4. σB[ir]) (360./(2. π)), 1000.}], Text[
     StyleForm[ ToString[Round[Length[sortηBarMin[ir]]], InputForm, NumberMarks → False],
      FontSize → 12, FontWeight → "Plain"], {(η0B[ir] - 3.0 σB[ir]) (360./(2. π)), 1000.}]}] }]
runDataRgn[ir][[1, 3, 1, 2]] (360./(2. π));
NumberForm[runDataRgn[ir][[1, 3, 1, 2]] (360./(2. π)), 3];
Clear[ir]
```

Out[70]=

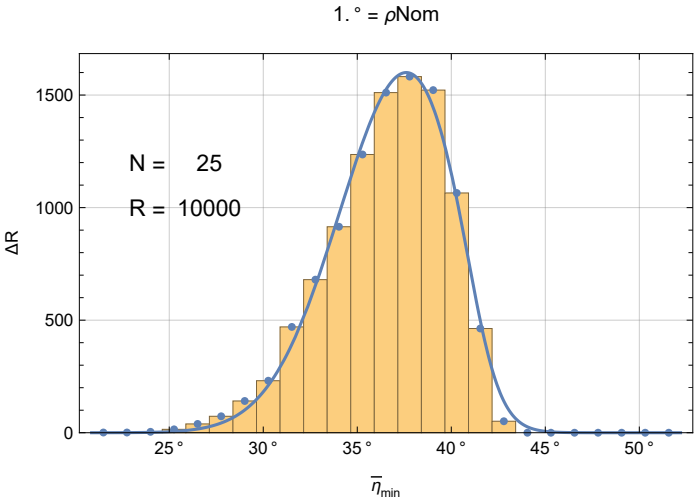

Figure B4. Each of $R = 10{,}000$ random runs yields a value of the smallest alignment angle $\overline{\eta}_{min}$. Those 10,000 values make the histogram drawn here. The fit to the histogram makes a distribution that depends on only two parameters, the location of the peak η_0 and the half-width σ.

Out[75]=

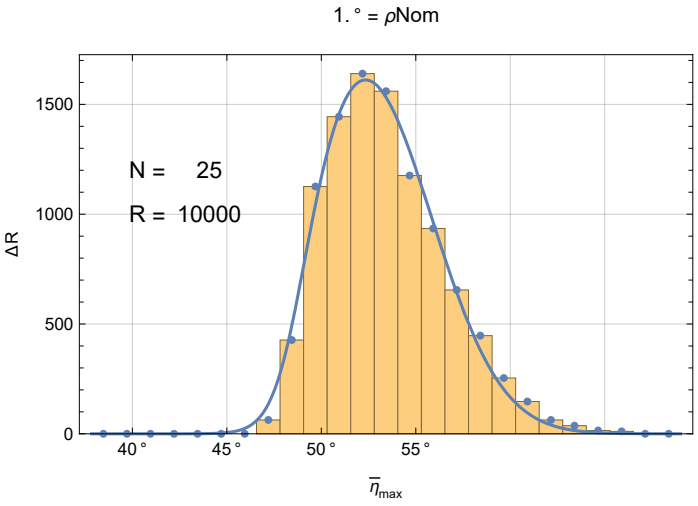

Figure B5. Each of $R = 10{,}000$ random runs yields a value of the largest avoidance angle $\overline{\eta}_{max}$. Those 10,000 values make the histogram drawn here. Note the symmetry about $\eta = 45°$ of the $\overline{\eta}_{min}$ and $\overline{\eta}_{max}$ histograms in Figs. B4 and B5.

Conclusion.

Figs. B4 and B5 summarize the article. The $R = 10{,}000$ random runs are generated by the computer program in Appendix A for each case (N, ρNom). The results collected in the table runData contain, as well as other information, the 10,000 values of the smallest alignment angle $\overline{\eta}_{min}$ and the 10,000 values of the largest avoidance angle $\overline{\eta}_{min}$. The computer program in Appendix B fits each histogram, reducing the 10,000 values to two parameters, the location of the peak η_0 and the half-width σ. Those two parameters plus other potentially useful or interesting data are collected in the fitData table. The fitData table is then available to other programs to reconstitute the distribution functions. Those distribution functions can be used to estimate the probability distributions appropriate for observed samples and determine the significance of Hub Test results. Thus, the fitData table is a 'reference Library' of distributions.

The date and time that this statement was evaluated: Wed 22 Jun 2022 07:15:17 GMT−4

The computer time expended so far is 160.265 seconds.

The computer memory in use is 591 467 120 bytes.