Minimum Zone Algorithms of Sphericity and Flatness

郭偉森 B08502110、連樂 B08502132

National Taiwan University Mechanical Engineering Department

Sphericity

First of all, we have to define the sphericity. Sphericity is a measure of how closely the shape of an object resembles that of a perfect sphere [1]. In this case, we have referred to a paper by Fan, and Lee (1999), the algorithm for finding the minimum zone sphericity [2]. The algorithm is as follows:

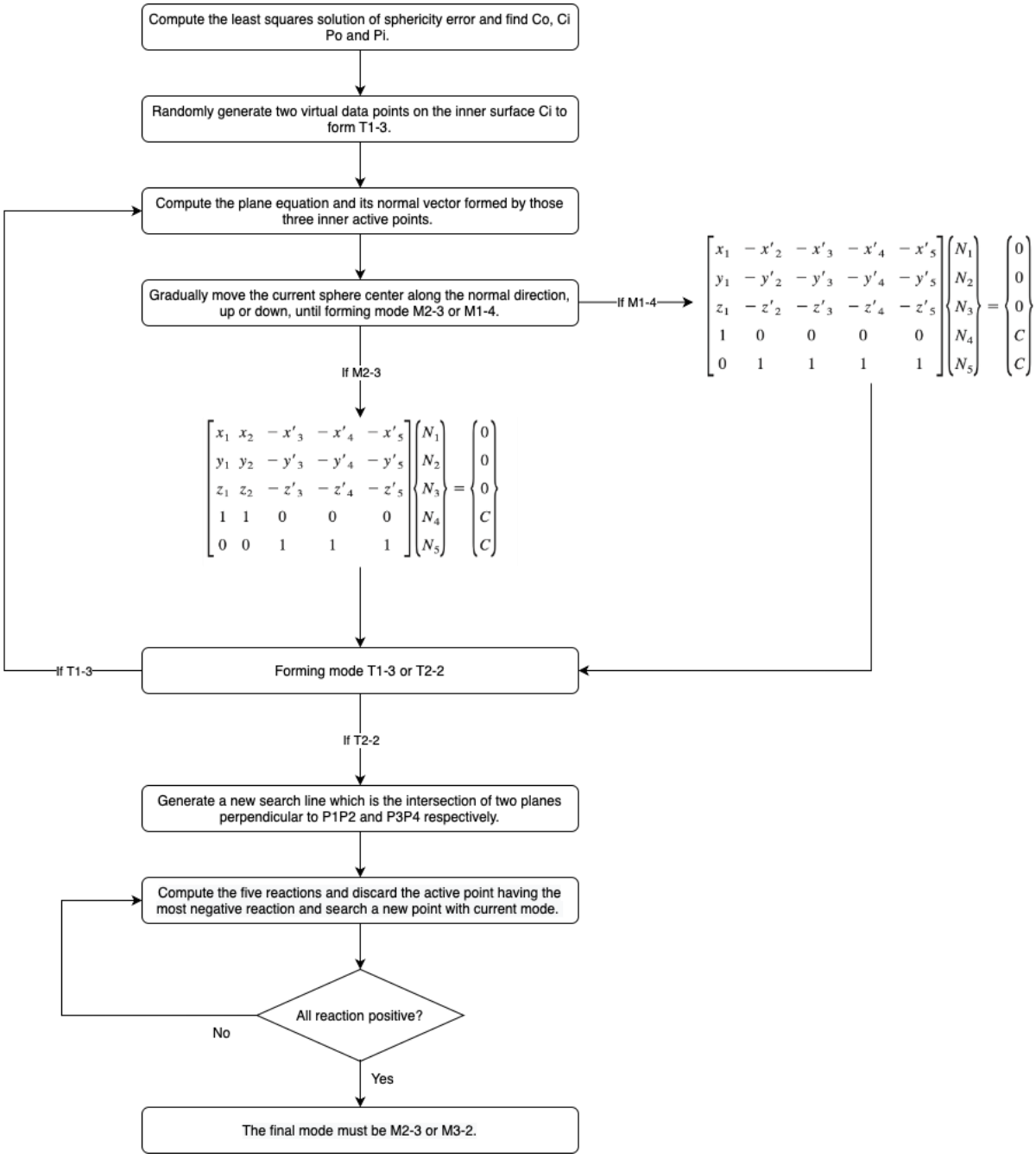


Fig 1. The flowchart of the algorithm for finding the minimum zone sphericity [2].

In Fig 1, the parameter shown is defined in the figure below:

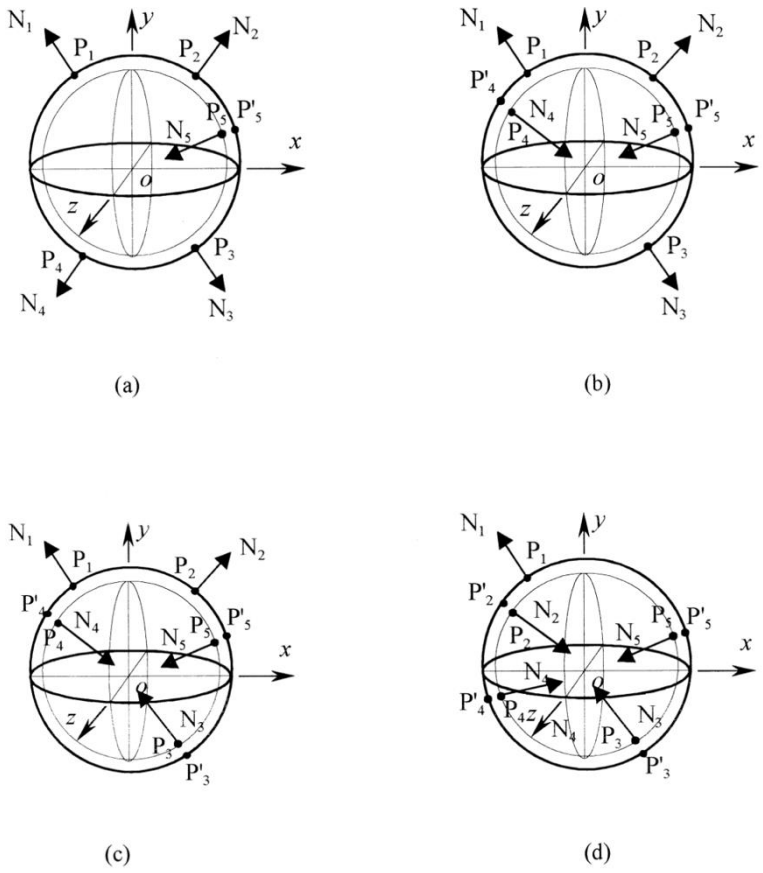


Fig 2. Free body diagrams of four modes of concentric spherical surfaces:
a) Mode M4-1, b) Mode M3-2, c) Mode M2-3, d) Mode M1-4 [2].

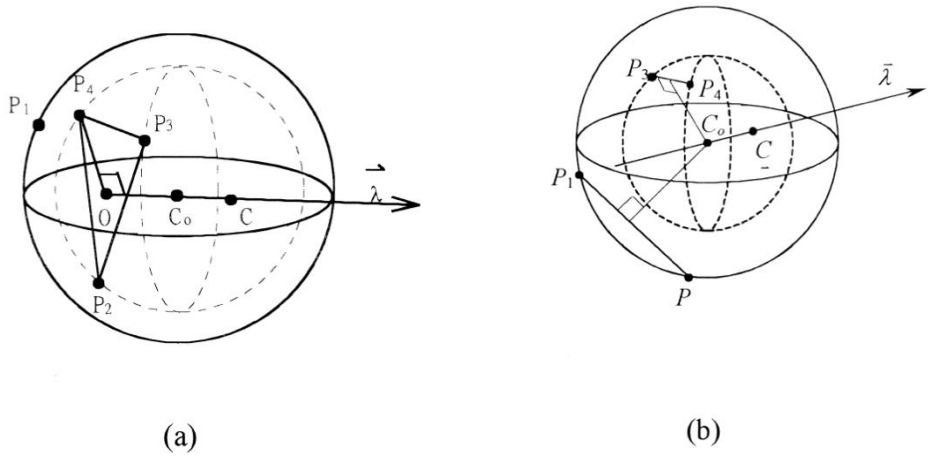


Fig 3. Generating new search direction: a) for Mode T1-3, b) for mode T2-2 [2].

This set of algorithms uses the concept of elastic potential energy to "pull each other" the outermost and innermost layers of the measured moral data points and compute the sphericity.

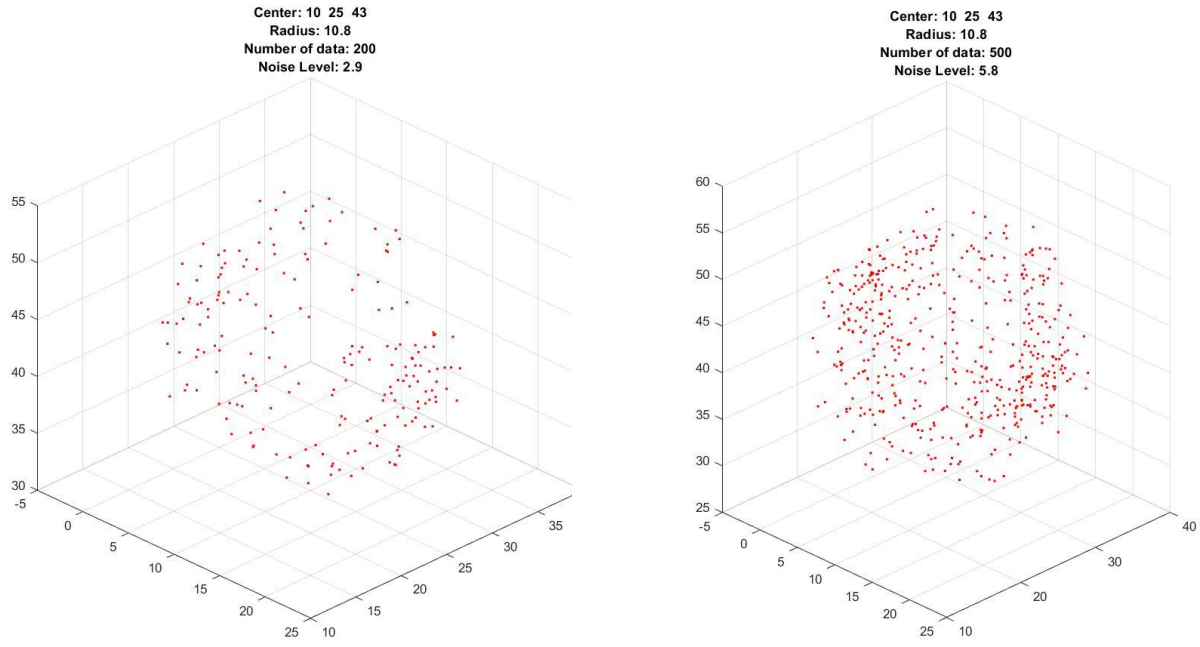
The minimum zone sphericity error is a measure of how closely the shape of a crystal grain in a polycrystalline material approximates a perfect sphere. The minimum potential energy theory is a mathematical model that predicts the equilibrium shapes of crystals by minimizing their potential energy.

To analyze the minimum zone sphericity error using the minimum potential energy theory, one would need to calculate the potential energy of the crystal grain with its actual shape and compare it to the potential energy of a perfect sphere with the same volume. The difference between these two values would give an indication of the deviation of the actual shape from the ideal spherical shape. This deviation can be quantified using the minimum zone sphericity error.

One approach to calculating the potential energy of a crystal grain using the minimum potential energy theory would be to use the elastic properties of the material and the deformation of the grain caused by external forces. The potential energy of the grain would then be given by the sum of the elastic strain energy and the surface energy associated with the grain boundaries.

In general, the minimum potential energy theory can provide valuable insights into the shapes of crystals and their deviation from the ideal spherical shape. By analyzing the minimum zone sphericity error using this theory, one can gain a better understanding of the factors that influence the shapes of crystals and the role of potential energy in determining their equilibrium shapes.

In order to check the feasibility of this algorithm, we generated random points, and these points all take an adjustable parameter as the center point, and with the noise level and the number of data points we set, produces a point cloud-like distribution, as shown below:



(a) (b)

Fig 4. The datasets generated by code.

In this case, we use the parameters shown in Fig 4(a) to compute the sphericity by Least Squares Method and the algorithm by Fan, and Lee.

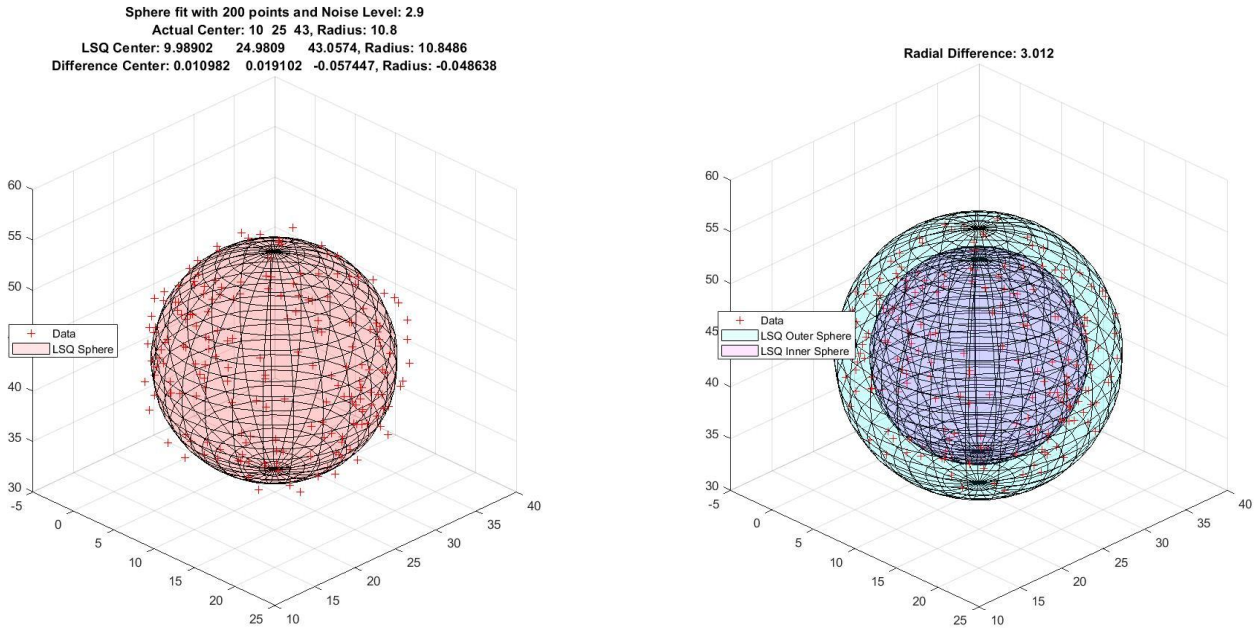


Fig 5. The result by using Least Square Method.

On the other hand, the minimum zone method:

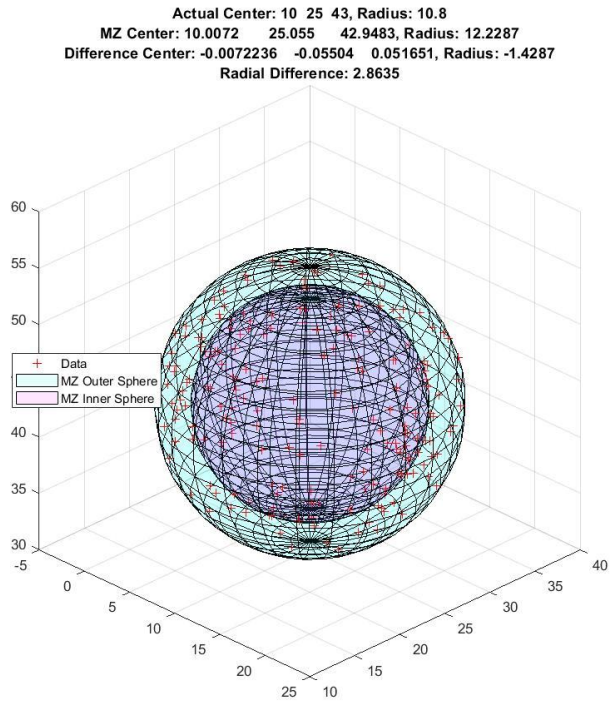


Fig 6. The result by using Minimum Zone Method.

From Fig 5 and Fig 6, we can observe that both algorithms can compute the right center point of datasets accurately, the error is about 0.12%. However, the algorithm by least squares method is unable to reach the correct value of radius compare with the algorithm by Fan, and Lee. It showed that the algorithm introduced above is more accurate.

Flatness

As for flatness calculation, our TA 傅子英 (Fu, Zih-Ying) provided us a flatness minimum zone method algorithm (MATLAB code) and a set of data consists of 100 points. Therefore, we did not plan on rewriting the code from scratch, instead we tried to think of any idea to improve the code.

In order to improve code running efficiency, there are two approaches we can take. One is reducing the space used throughout the code (space complexity), the other is reducing the time used for the code to run (time complexity). While we were brainstorming, we came across a function provided by MATLAB called *convhull()* [3], which the input parameters are the coordinates of data set, the output parameters are the index of the coordinated provided that surrounds other data, or we can say is all the outermost points among all the other points.

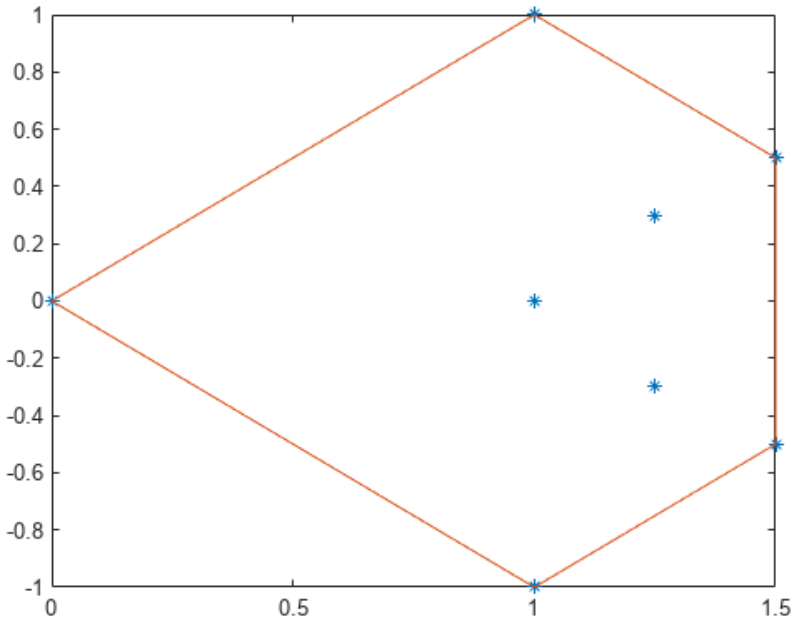


Fig 7. *convhull()* on 2D data sets.[3]

The *convhull()* function can act as a filter to choose only the important data points to be calculated using the minimum zone method. Before this, the original algorithm will go through all the data points given to calculate which 4 points are needed to form the 3-1 model or 2-2 model. To simplify the problem, the algorithms are actually needed to go through the outermost points among the whole data in order to perform minimum zone method calculation, and *convhull()* function will choose the outermost points for us.

Fig 8. shows the original data and the filtered data that is highlighted by the code. By looking at the 3D plot, we can visualize it very easily that the number of filtered data is much fewer than the original data.

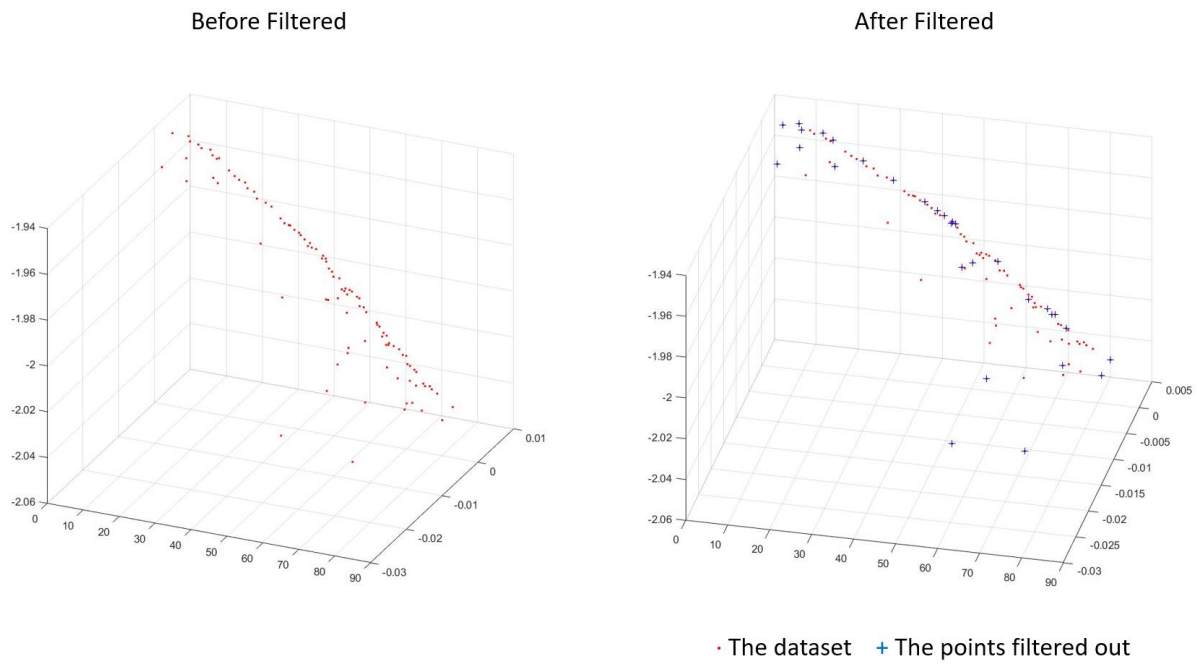


Fig 8. Filtering the data using *convhull()*.

Before filtering the data, we need to make sure where we should be filtering the data. If filtering the data before calculate the least squares, the least squares calculated will be different from the original least square values, this is not the case we want. The improvement we made to the code has to compute the same values before the improvement, else it is just a different code. Thus, we need to filter the data after calculating the least squares and before calculating the minimum zone.

Attempt 1	Attempt 2
>> flatfinal {'Number of data: 100' } {'LSQ = 0.0047125' } {'minimum zone = 0.0038265'} Elapsed time is 0.226419 seconds. >> >> flatfinal_improve {'Number of original data: 100' } {'Number of data reduced to: 39'} {'LSQ = 0.0047125' } {'minimum zone = 0.0038265' } Elapsed time is 0.104090 seconds.	>> flatfinal {'Number of data: 100' } {'LSQ = 0.0047125' } {'minimum zone = 0.0038265'} Elapsed time is 0.218539 seconds. >> >> flatfinal_improve {'Number of original data: 100' } {'Number of data reduced to: 39'} {'LSQ = 0.0047125' } {'minimum zone = 0.0038265' } Elapsed time is 0.110433 seconds.

Fig 9. Time taken to run the original and improved algorithm.

In Fig 9, “flatfinal” is the original algorithm by TA, “flatfinal_improve” is the improved version of the algorithm. We can see that after running for two attempts, the time taken to run the algorithms have significant difference. One more thing to remind, the time taken for the same code to run will not always be the same every time, there are several reasons for this. First is because the setup for everyone device is different or the background running applications for the being is different at any given time as well. Despite having different time, the time taken for improved version algorithm is always faster than the original version.

Reference

[1] https://en.wikipedia.org/wiki/Sphericity
[2] K.C. Fan, J.C. Lee, Analysis of minimum zone sphericity error using minimum potential energy theory, Precis Eng. 23 (1999) 65—7
[3] https://www.mathworks.com/help/matlab/ref/convhull.html