CHAPTER 1

LINEAR INTERPOLATION IN N DIMENSIONS

Interpolation is one of the most common operations in astronomy. Resampling spectra to a different wavelength grid, projecting images or interpolating physical quantities in n-dimensional fluid dynamics simulation are all examples of interpolation. Interpolation can be described as a special case of curve fitting which requires the function to go through all points.

In one dimension interpolation is relatively easy and there exist multiple methods. The simplest method is nearest-neighbour interpolation in which the algorithm picks the closest neighbour point to the point to be interpolated.

Linear interpolation is one of the most common methods of interpolation. The two neighbouring points of the point to be interpolated are found and their slope and offset is used to obtain the interpolated point.

A more sophisticated approach is the spline-method of interpolation. Splines are piecewise polynomials of n-th degree whose first and second derivation are the same at the data points.

In one dimensional space there exists an abundance of interpolation methods. This multitude of options decreases rapidly with increasing number of dimensions. We will focus on the implementation of linear interpolation in N-dimensions, although there are a few other options like nearest neighbour interpolation and radial basis function.

For our linear interpolation we have opted to use Delauney Triangulation as a interpolation method. The interpolation involves multiple steps to arrive at an interpolation solution: At first a Delauney Triangulation is performed on the existing points. As a next step we need to find the simplex(geometric structure; a triangle in two dimensions) that contains the point to be interpolated. Finally we will use the barycentric coordinate system of the simplex to perform the actual interpolation.

1.1. Delauney triangulation

A triangulation is the process of connecting all points in a set with straight lines without any two lines crossing (see Figure 1.1). It is obvious that there many ways for a set to be triangulated. All triangulations however have the same outer boundary called the convex

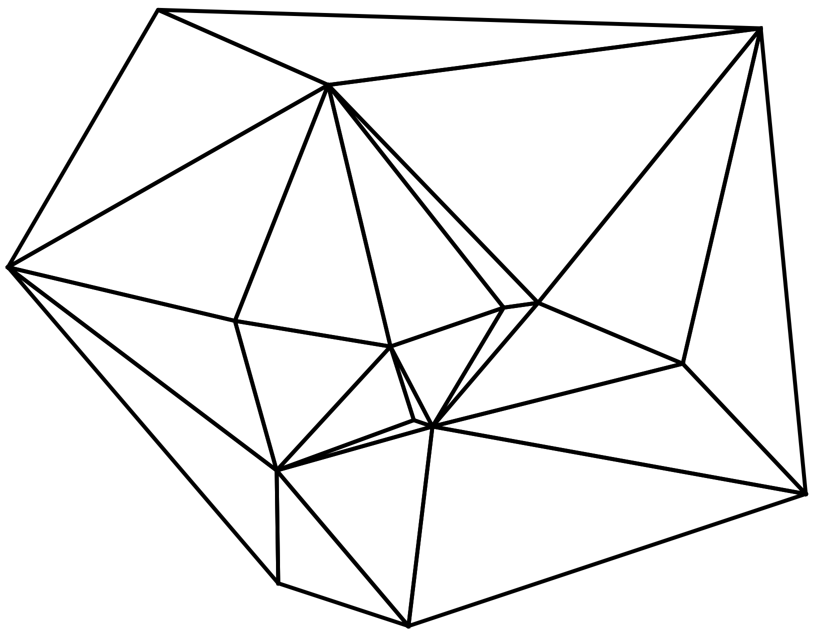


Figure 1.1 Delauney Triangulation of 20 points in two dimensions.

hull. One special kind of triangulation is the Delauney Triangulation. The Delauney Triangulation can be defined a various abstract ways and has intriguing properties.

For a description of the process we will limit ourselves to two dimensions. This process is expandable to n dimensions in which the triangles become geometric structures called n-simplices. One such defintion is that the circum-circle (circle going through all three points of the triangle) of each triangle must only contain three points. Figure 1.2, a simple example, shows one *legal* triangulation and one *illegal* triangulation. One can see in the *illegal* triangulation that the circum-circles of both triangles contain more than tree points. By doing a simple "*edge-flip*" one arrives at the Delauney Triangulation. In addition this ensures that the triangulation gives the largest minimum angle for both triangles.

Delauney Triangulation and convex hulls have a very interesting relation. It is possible construct the Delauney Triangulation in n dimensions from a convex hull of the points projected on a paraboloid in n+1 dimensions. Figure 1.3 shows an example of a Delauney Triangulation in two dimensions constructed from the convex hull in three dimensions. To project the points onto the paraboloid one just square sums the coordinates n dimensions and uses this as the coordinate for the point in n+1 dimensions.

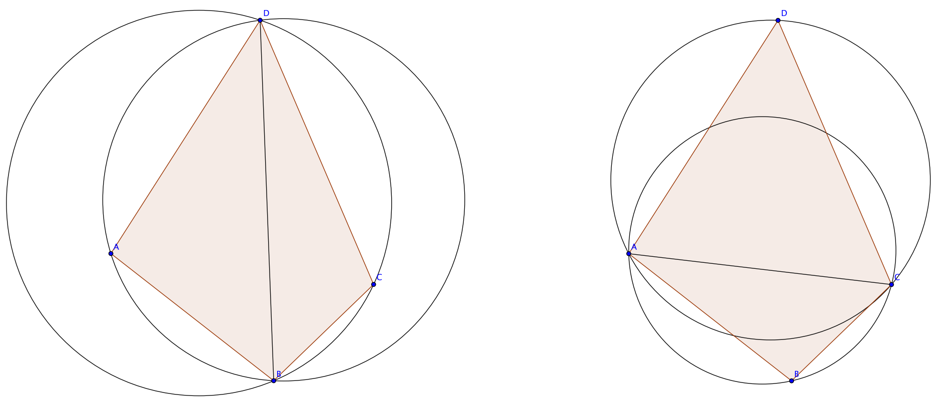


Figure 1.2 The left figure shows an '*illegal*' triangulation of the 4 points. Both circles include all the points. With a so called edge flip one can arrive at a '*legal*' triangulation.

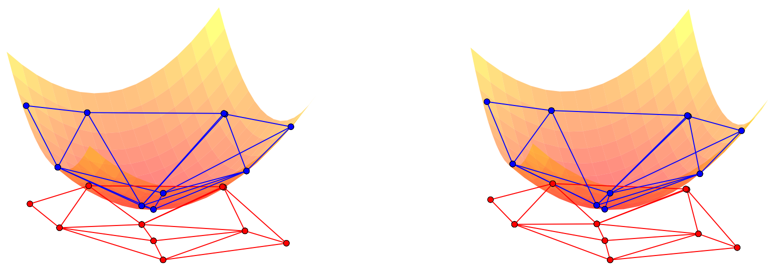
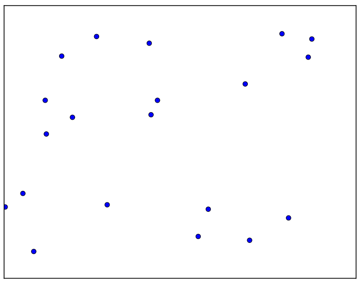


Figure 1.3 Stereogram (produced using a method described in Vogt & Wagner, 2011) of the projection of the convex hull in three dimensions to form the delauney triangulation in two dimensions.

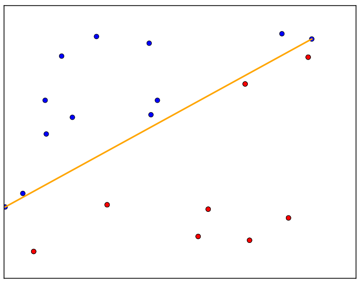
1.2. Convex Hull

In section 1.1 we have described the relation between the convex hull and the Delauney Triangulation. There are multiple ways to construct the convex hull for N points, we will limit ourselves to the description of the Quickhull algorithm (Barber et al., 1996). Similar to the Quicksort algorithm it follows the divide and conquer strategy.

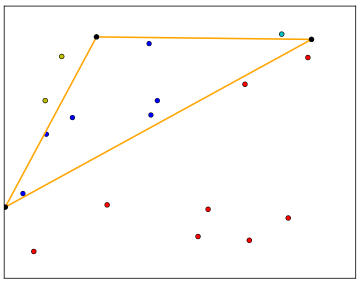
As an initial input we have N data points. Although this method works in n-dimensions, we will show an example in two dimensions. The first operation is finding the two extreme points in the horizontal axis, which are guaranteed to be part of the convex hull. We connect these two extreme points thus creating a division between a "left" and a "right" set of point. Now the divide and conquer method begins. We will only describe what



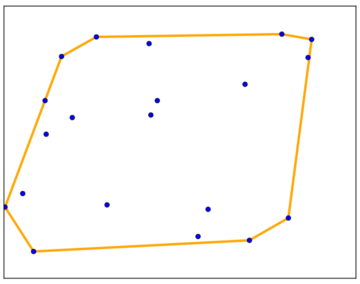
(a) Twenty points for which we are trying to find the convex hull.



(b) We find the points with the lowest and highest x-value and connect them with a line.



(c) Continuing with the points on the left (same process happens recursively on the right) we find the point furthest away from the line. We then draw two more lines and build a triangle. The points inside of the triangle are not part of the convex hull and are discarded. We will repeat the current step with the two new lines of the triangle.



(d) We have found points of the convex hull once we can't build a new triangle anymore.

Figure 1.4 Determination of a convex hull in two dimensions.

happens to the left side, but imply that the same steps are taken on the right side. We find the point furthest away from the dividing line and add it. A triangle is formed out of the two points of the initial dividing line and the additional point. All points inside the triangle do not belong to the convex hull and thus we exclude them. The triangle again divides the remaining points into two sets, one left of the triangle and one right which are again iterated over recursively.

The method is repeated until each subset only contains the start and end point of the dividing line. We have created the convex hull, which if projected to a $d-1$-dimensional space provides the Delauney Triangulation of the projected points. For this projection to work we need to construct a convex hull in more than two dimensions which uses the same technique as described.

1.3. Barycentric coordinates system

The actual interpolation transforms the interpolant's coordinate into the barycentric coordinates of the containing triangle.

One can construct the barycenter of a triangle by drawing lines from each point to the midpoint of the opposing side (see Figure 1.5).

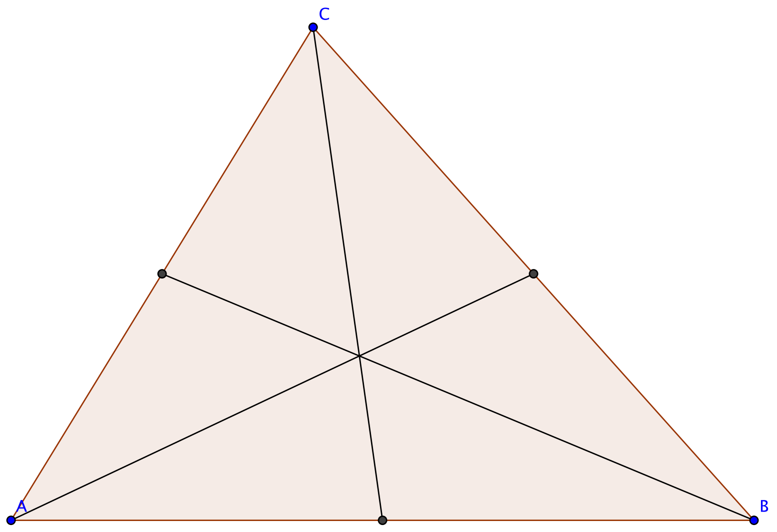


Figure 1.5 The triangle and its barycenter marked by the intersection of lines.

The coordinates of the barycenter M can simply be expressed by,

$$\vec{M} = \frac{1}{3}(\vec{A} + \vec{B} + \vec{C}).$$

Not only the barycenter can be expressed by the vectors of $\vec{A}$, $\vec{B}$ and $\vec{C}$ but every point p inside the triangle can be expressed by,

$$\vec{p} = \alpha\vec{A} + \beta\vec{B} + \gamma\vec{C},$$

where

$$\alpha + \beta + \gamma = 1.$$

α, β and γ are called the barycentric coordinates. If the point p lies within the triangle all barycentric coordinates are positive.

1.4. Triangle Finding and Interpolation

To calculate the interpolation using barycentric coordinates we need to find the n-simplex that contains the interpolant. We use a method called directed walk (priv. comm. Pauli Virtanen). We choose a random starting n-simplex. In order to interpolate to a given point (interpolant) we calculate the barycentric coordinates for the interpolant and test if all of them are larger than 0. In that case we have found the n-simplex that contains the point. If the n-th barycentric coordinate is negative we jump to the neighbouring n-simplex which is opposite the n-th point. This is iterated until the containing n-simplex is found or the next jump would lead outside the convex hull of the Delauney Triangulation. For the latter case the point is outside of the grid and can not be interpolated.

If this algorithm converges and the right n-simplex is found the interpolation can be easily performed using the barycentric coordinates:

$$f(\vec{p}) = \alpha f(\vec{A}) + \beta f(\vec{B}) + \gamma f(\vec{C})$$

where $\vec{A}$, $\vec{B}$ and $\vec{C}$ are the points of the triangle.

1.5. Conclusion

We have described the method of linear interpolation using Delauney Triangulation mainly in two dimensions. As mentioned this method is easily extensible to n dimensions. The triangles (3-simplices) become n-simplices in n dimensions (e.g. Tetrahedrons in three dimensions). The method itself however stays very similar for higher dimensions.

In this work we have made extensive use of n-dimensional linear interpolation using the implementation present in the SciPy-package (**?**), called LinearNDInterpolator. In this case creation of the convex-hull is performed by the QHULL implementation described in Barber et al. (1996).

We have tested the performance of the algorithm by creating a three dimensional grid with $20 \times 10 \times 10$ gridpoints and an array of $10,000$ double values at each gridpoint. On a standard 2011 MacBook Pro (Intel Core i7, 2.6 GHz, running only on a single processor) we have measured the initial building of the Delauney Triangulationand storing it in an appropriate data structure to 256 ms. The interpolation for random points took on average $\approx 600\,\mu$ s. This technique lends itself very well to explore large datasets even on moderately equipped machines.

We have used this technique extensively to interpolate a spectral grid in three dimension (effective temperature, surface gravity and iron abundance). When trying to extract stellar parameters from an input spectrum we calculated the χ^2 for the observed spectrum with interpolated synthetic spectra from the grid. The interpolated spectra resulting from the interpolation are continuous, but not differentiable at the ridges of the grid structure (ridges are the borders of the simplices that make the grid). This non-differentiability at the ridges can be seen even in the χ^2 space. Optimizers that employ gradient methods (such as MIGRAD James & Roos, 1975) show some difficulties in some regions of the search space. We have tried to alleviate this problem by beginning the optimization from different starting points. In almost all cases this lead to the same minimum.

In summary, the presented linear n-dimensional interpolator is a very robust and quick way to explore large parameter spaces without having to compute each single point.

Future work will be directed to exploring other n-dimensional interpolators in the astrophysical context.

BIBLIOGRAPHY

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. 1996, ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE, 22, 469

James, F. & Roos, M. 1975, Comput. Phys. Commun., 10, 343

Vogt, F. & Wagner, A. 2011, Astrophysics and Space Science, submitted.