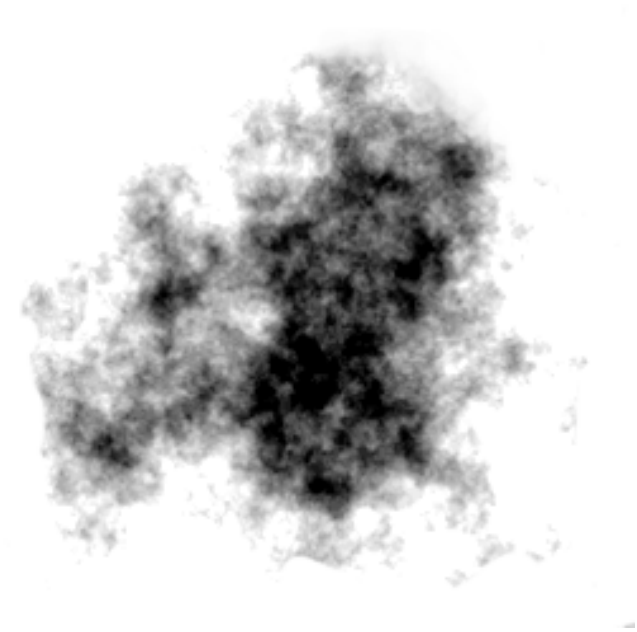
Abstract

In this master thesis I review the theory of fractals, interstellar clouds, and their interrelation. In particular, I examine some properties of fractal models, their projection, their relation to observed interstellar clouds and how the models can be related to the stellar initial mass function. An aid in my investigation has been a computer program package I devoted some time to develop for the purpose. The main conclusions can be summarised as follows:

- There is a fractal model of the density distribution of interstellar clouds, called fractional Brownian motion (fBm), that is consistent with the observed density distribution of real interstellar clouds.
- The relation between the fractal dimension of a three-dimensional fBm model cloud and the fractal dimension of its two-dimensional projection has been analytically determined.
- It is possible to derive the stellar initial mass function (IMF) from an fBm model for a star forming region, and the resulting IMF is consistent with the observed IMF. Also, the resulting IMF seems to be very stable with respect to parameter changes in the model (this is, however, to be confirmed by more thorough analysis).



Acknowledgements

I would like to thank my thesis adviser Dr. René Liseau for directing my attention towards the subject of this thesis, as well as for doing a critical structure analysis of the report. I also thank Frank Bensch for providing the latest posters and articles related to his work.

Table of contents

1. Introduction

1.1 Fractals in nature

As Mandelbrot put it, *clouds are not spheres*. Nor are they described particularly well by any other geometrical standard form used in the Euclidean geometry. Nature, in general, exhibits a complexity that cannot be accurately modeled within the context of simple geometrical figures. Despite this fact, scientists have used, and in most cases are still using, this Euclidean geometry to approximate nature with a varying degree of success. Curiously enough, Mandelbrot discovered in the early seventies that there are in fact more realistic models of several phenomena in nature; he coined these new models fractals.

When you take a closer look at nature, you begin to find fractals everywhere. Not only clouds, but coastlines, mountains, lightning, diffusion, stock market, galaxy distribution in the universe, blood-vessels, sunspots and even music[1] are also things well described by fractals. This is a truly remarkable fact, but from a mathematical point of view this may not be surprising, because as soon as you start to interpret equations as dynamical systems, there pops out a fractal almost surely.

There is a point I would like to emphasise here before we get too excited about these fractals; *they are only models*. And as such, they have a limited range of validity. There are no exact fractals in nature, in the same sense as there are no exact spheres in nature. One has to carefully investigate this range of validity in order to decide whether the model is useful or not. In several cases scientists have modeled phenomena with fractals when this range has been less than one order of magnitude; one can question if the use of fractals really is motivated in such cases[2]. However, in our case of interstellar giant molecular clouds, this range is surprisingly large. Indeed, in some cases there has been no significant deviation from the fractal model from the largest scale down to the smallest scale observed, a range of 8 orders of magnitude! A corresponding biological model that had an equally large range of validity would successfully model every animal from the whale down to the bacteria[3] (a range of about 10^{2} to 10^{-6}m).

1.2 Interstellar Clouds

The space between the stars, the interstellar space, is far from being empty. Although it is true that even the densest parts of the interstellar medium (ISM) resemble the very best vacuum one can produce on Earth, there are never less than several hydrogen atoms or protons per cubic meter space. Especially in a spiral galaxy like our own, there are plenty of particles, molecules and dust between the stars. The densities of these particles are not uniformly distributed in space, there are large variations on many scales. When the densities tend to rise significantly over the surrounding ISM, we call these regions interstellar clouds. This because of their close resemblance of terrestrial clouds, see the colour plates 2 and 4 for comparison.

---

[1] *Voss (1989) analysis music and finds that most music can be viewed as fractals. He even manages to generate some music sounding melodies using the model.*

[2] *See e.g. Avnir et al. (1998)*

[3] *This range excludes the recent findings of something called "nano-bacteria", bacteria thought to be as small as 10^{-9} meters.*

Now, if one such cloud happens to contain bright stars, then the radiation from the stars will cause the gas in the cloud to fluoresce and an *emission nebula* will appear. *Nebula* is just the Latin word for *cloud*. A famous example of an emission nebula is the nebula of Orion that can be seen by the naked eye a dark night as the middle ”star” in his sword. On photographs of the nebula it looks clearly red, but in binoculars or a smaller telescope it looks very greenish if any colour at all. This is because the cones that sense the colours in our eyes are most sensitive to green light, so they tend to favour the fainter green light, emitted by oxygen ions, over the more luminous red hydrogen emission seen by photographic film.

If the stars close to the cloud are less bright and hot, they will not cause the nebula to fluoresce but rather just reflect and spread the light. Because, just as in our own atmosphere, blue light scatters more easily than the redder part of the spectrum, reflection nebulae mostly are blue. On photographs of the Pleiades cluster, the seven sisters that can be seen in the constellation of Taurus, one can see the ghostly blue glow of the cloud the stars in the cluster happen to pass.

Because space is so vast, and stars inhabit a very small part of it, there is plenty of room for *dark clouds* to live. Historically one could see these indirectly, either because it appeared to be too few stars in their directions (for a long while astronomers actually debated whether this was due to a hole in the universe or just obscuring clouds. The evidence however, luckily turned into favour of the latter alternative) or because of their fingerprint on the spectrum from stars behind the clouds. More recently one has constructed radio antennas and infrared detectors to observe the cold radiation from these clouds directly. Because the atmosphere is a scourge for the infrared astronomers, infrared telescopes have been carried above the atmosphere as satellites. IRAS of the eighties and ISO of the nineties are two prominent such, ISO ending its service as late as in April this year 1998.

Some clouds are really big. Smaller or ordinary interstellar clouds have masses that range from 1 to 1000 solar masses M_0, where $M_0 = 2.0 \times 10^{30}$ kg. Larger or giant clouds on the other hand, can contain as much as one *million* solar masses, really a huge amount of mass. These clouds are generally very cold, typical temperatures are about 10 K (or -260°C). When first observed in radio telescopes, these clouds proved to emit a lot of radiation from the molecule carbon monoxide; hence the name *giant molecular clouds* (GMC). It is in these GMCs one finds the interesting and complex molecules of outer space, as ethanol for instance. But GMCs have one other and even more interesting quality; they are the cradles of the stars, the places where the stars are born.

1.3 The birth of a star

According to the ideal gas law, pressure in an ideal gas is proportional to the temperature and the density. This implies that colder gas can sustain the same pressure for a larger density. In the case of molecular clouds, these are so massive that gravitation is an essential factor in their dynamics, and the denser the cloud, the more important gravitation. Because GMCs in many respects can be viewed as an ideal gas, we find that if gravitation plays any role at all in interstellar clouds, it should play the greatest role in GMCs because they are massive, and they are cold.

Jeans investigated under which circumstances a cloud, or part of a cloud, can get so massive that the gravitation conquers the resistance of the heat pressure and starts a gravitational

collapse. He landed on what now is called the *Jeans criterion*, which states that the mass has to be greater than the *Jeans mass* $M_J \propto \rho^{-\frac{1}{2}} T^{\frac{3}{2}}$, where ρ is the mean density and T is the temperature of the region. According to this simple theory, clouds with masses above their Jeans mass should start to collapse in a free fall fashion. Observations tell us, however, that this is rarely the case. GMCs have generally masses well above their Jeans masses and have estimated ages that are many times greater than the free fall time. This hints that there have been simplifications in the theory that may not be valid. For instance, electromagnetic fields and turbulence are suggested to besides heat contribute substantially to the pressure. These pressures, together with centrifugal resistance, has to be overcome in order for stars to form.

1.4 The stellar Initial Mass Function

When we look above us, one clear night, the stars we see are the very brightest in our part of the Galaxy. Almost all stars we see with our unaided eyes are many times brighter, and many times more massive than our own sun. Should we then conclude that most stars are intrinsically bright and that our own sun is an exceptionally low mass star? Of course not. The reason for why we see many more bright stars than dim ones is that bright stars can be seen over much larger distances. What we see with our eyes are actually distant bright stars that outshine closer dimmer ones. If we want to know the mass distribution of stars in space, we cannot just look for every star we can see, but have to select an unbiased sample.

One choice of what could be an unbiased sample is to measure the mass of all stars within a certain distance from the sun. Another is to look in a clearly limited region somewhere else, as in star associations, clusters or even other galaxies. All samples have in common that one must be careful to determine the faintest stars to where one has total coverage.

The faintest stars are so dim that they are extremely hard to see over large distances. Indeed, the nearest neighbour of the Sun, the star Proxima Centauri in the multiple star system of α Centauri, is so faint that it is not even visible for the naked eye at an apparent visible magnitude of 10.7. The use of powerful telescopes in the search for dim stars is therefore essential. Since the faintest stars are often rather cold, they radiate most energy at longer wavelengths, that is at red and infrared wavelengths. One can therefore see fainter stars if one chooses to look in those wavelength bands, and thus improve the limit of total coverage.

These problems put aside, imagine that we have a sample for a region that is complete down to a certain limit. We measure the mass of each star and make a histogram to determine the number of stars in a mass interval $m + dm$. If we normalise this function with the total mass of all stars, it can be interpreted as the probability for a star of the sample chosen at random, to have a mass in the interval $m + dm$. This function is called the present day mass function (PDMF) and has been determined for several regions[4].

If we are interested in the formation of stars, however, this PDMF may not be of much direct help. The evolution and lifetime of stars depend very much on their masses, so the initial mass function (IMF), the probability function for a star being born with a certain mass, will be strongly affected as time goes on. More massive stars will tend to live shorter, so the PDMF will not directly reflect the IMF.

---

[4] *See e.g. Scalo (1986) and Rana (1987)*

What one has to do is to involve the theory of the structure and evolution of stars, which is quite laborious. One has to answer questions like ”OK, the observed mass function is not identical with the initial one because of the death of more massive stars, but what if all stars remained on the main sequence forever, what mass function would we see then?”.

This has been subject to a lot of work, prominently by Scalo (1986).

1.5 Observations of the IMF

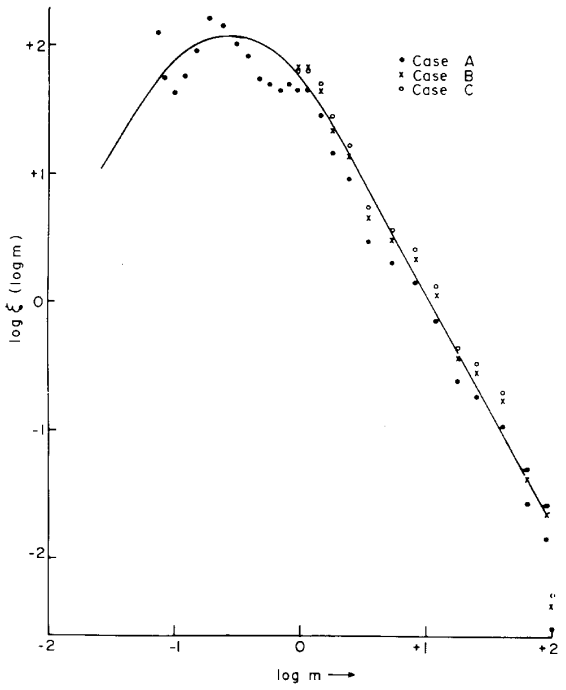


Figure 1-1 *The IMF of the solar neighbourhood, as determined by Rana (1987)*

What does the stellar initial mass function (IMF) look like? If we had no observational clues, we would not know what to expect. It could look like anything, a reasonable guess would be that it looked like random noise, without any special properties. *This is not the case*. If we plot the logarithm of the population mass density as a function of the logarithm of the mass, we will see that the IMF looks extremely smooth and is remarkable well approximated by a power law for higher masses. The IMF determined by Rana (1987) for the solar neighbourhood can be seen in Figure 1-1.

This surprising fact was not predicted by the theory of star formation. It is not only that the high mass end IMF of our solar neighbourhood is well described by a power law, but IMFs of other regions look quite the same - with the same power law fit!

The question that arises is ”Why?”. Nobody knows for certain, but we have some clues:

i) The high mass end IMF power law is stable over space and time.
ii) The clouds where the stars are born look self similar.
iii) The gravitational law has no intrinsic scale, or equivalently, the Jeans mass criterion is a power law of the density.

People have tried to explain the IMF by looking at the facts above. Henriksen (1986) examined the effects of gravitation coupled with turbulence; Larson (1992) tried a pure geometrical approach based on fractal geometry as well as Elmegreen (1997) who also did some numerical simulations. In chapter 6 I will try an alternative probabilistic approach based on a fractal model.

1.6 About this master thesis

In this master thesis, I will examine some fractal models for interstellar molecular clouds as well as one extension of the fractal concept, the multifractal. Three main problems are treated: firstly, to find a model that fits the data. Secondly to investigate the effects of projection of

three-dimensional structures onto two dimensions (”the plane of the sky”). Thirdly to look for a connection between the fractal model and the initial stellar mass function IMF.

This report is a part of my master thesis, performed at the Astronomy Department of the Stockholm University (SU). This thesis completes my Master of Science degree in engineering physics at the Royal Institute of Technology (KTH). My thesis advisor and examiner has been Dr. René Liseau at Stockholm Observatory.

1.7 Outline of the chapters

Chapter 2 introduces fractals and some properties of them. Chapter 3 treats the problem of describing the geometry of interstellar clouds with fractal (as well as multifractal) models. Chapter 4 investigates the effects of projecting fractals. Chapter 5 reviews methods of comparing fractal structure models with observations of interstellar clouds. Chapter 6 presents (analytical) results related to the IMF and fractional Brownian motion (fBm). Chapter 7 presents results from fBm computer simulations. Chapter 8 contains general conclusions of this master thesis. Appendix A contains a detailed description of the routines used in a developed computer program package for generating and analysing fBm. Appendix B contains the full source code to the program package. Appendix C is a vocabulary giving the definitions of some concepts and notations used in this report.

2. An introduction to fractals

2.1 The concept of dimension

The key concept in understanding what fractals are lies in the *fractal dimension*. When first heard of, the fractal dimension sounds like a completely crazy extension of our intuitive notion of dimension. The fractal dimension is often fractional; this is not at all compatible with what we learned from algebra, that the dimension of a space is defined as the smallest number of vectors needed to span that space. In the 3 space dimensions we live in, we traditionally denote the co-ordinates of three orthonormal vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. What should it mean then, that a space has a dimension of, say, 2.5? What kind of vectors span this space..?

The difficulty here is that *sets* are usually not vector spaces. With the definition of dimension given above, even an ordinary flat square would not have any dimension. As most of us would insist that a sensible definition would be one that assigned the flat square the dimension of 2, we could try to define the dimension as the least number of parameters needed to describe a set.

This definition seems to catch the essence of our intuition of the dimensional concept. Unfortunately, mathematics often reveal weaknesses in our intuition, and in this case Cantor showed that there exists a one-to-one correspondence between the line and the plane, that is, you need only one parameter to completely describe, for example, a square.

2.2 The topological dimension

To come to terms with this discrepancy between mathematics and intuition, Poincaré and Lebesgue among others contributed to what now is called the *topological dimension*. It is based on the following observation: to cut a connected volume into two pieces, you need at least a surface to separate the two parts. To cut a connected surface into two parts, you need at least a curve, and two cut a connected curve into two pieces you need at least one point. Define the dimension of a point to be zero, and the dimension of a connected set to be equal to 1 plus the dimension of the least set needed to cut it into two pieces. For a disconnected set, define the dimension of the set to be the maximum of the dimensions of the connected components.

2.3 The Hausdorff-Besicovitch dimension

With this definition of topological dimension, we are ready to face the dimension of Hausdorff and Besicovitch, more commonly known as the fractal dimension. Intuitively, it works as follows. We define the dimension of a D-cube to be D, that is, an ordinary cube has 3 dimensions, a square 2 and a line 1. Imagine a flat surface of finite area we would like to determine the dimension of. You can always make a square large enough to completely fit the surface into the square. Similarly, you can always make the square small enough to be entirely covered by the surface. Together, these things say simply that the area of the surface is finite and positive. If you take a cube instead of a square, you can surely make the cube large enough to completely contain the surface, but no matter how small you make the cube, it can never be fitted into the surface. This correspondingly means that the volume of the surface is finite and

zero. Choosing a straight line instead of a square or cube, no matter how long you make the line, it will never contain the hole surface, but making the line short enough it will easily fit into the surface. This means that the length of the surface is infinite and positive. Thus, the limiting case is precisely the square, and we define the dimension of the surface to equal the dimension of the square.

The idea of Hausdorff was to generalise this concept, not only to curved surfaces, but to any arbitrary set embedded in Euclidean space. To understand this we need first to consider a few mathematical relations.

To begin with, we define a *set* to be a collection of points in an Euclidean *n*-dimensional space $\boldsymbol{R}^n$. Volume, area and length are all special cases of what is called *measures*. Measures are mathematical entities, often denoted μ, that given a set, assign it a positive real number called the μ-measure of the set. A measure is thus a function from sets to positive real numbers. In the example above, we saw that the volume, area and length measures of a flat surface were zero, finite and infinite respectively.

There are several other measures than the volume, area and length measures. Hausdorff introduced a special class called Hausdorff measures. The volume, area and length measures are all special cases of it. It is defined as follows. Let U be a set. $|U|$ is then called the diameter of the set, and is defined as the largest of all distances between any two points in the set. Let $\{U_i\}$ be a countable collection of sets. Let F be another set. We say that $\{U_i\}$ is a δ-cover of F if no set U_i has a diameter that is larger than δ and if every point in F is also a point of at least one of the sets U_i. The Hausdorff s-measure of order δ is the minimum sum $\Sigma_i|U_i|^s$ of all possible δ-covers. When we decrease the order δ, there are less possible δ-covers to minimise the sum $\Sigma_i|U_i|^s$ over. Thus the minimum sum will tend to get larger as we make δ smaller. In the limit, when δ approaches zero, we have the Hausdorff s-measure (without order) denoted H^s.

In mathematical language, the definition is conveniently written as

$$H^s(F) = \lim_{\delta \to 0} \inf \left\{ \sum_{i=1}^{\infty} |U_i|^s : \{U_i\} \text{ is a } \delta\text{-cover of } F \right\}.$$

When s equals 3, 2 or 1, the Hausdorff measure coincides within a constant with the volume, area and length measures respectively. Curiously enough, the Hausdorff measure of a fixed set F is either infinite or zero for all but at most one value of s. In fact, there exists a limit point s_0 such that for all $s < s_0$, the Hausdorff measure is infinite, and for all $s > s_0$ it is zero. This unique point s_0 defines the Hausdorff dimension of the set F, and we write $\dim_H F = s_0$. The Hausdorff s_0-measure may or may not be a finite positive value.

Compare with our flat surface S: $H^3(S) = 0$, $H^2(S) < \infty$ and $H^1(S) = \infty$. Thus we concluded that the dimension of S is 2, $\dim_H S = 2$.

2.4 The box counting and self-similar dimensions

Even if it is mathematically useful and elegant, the definition of the Hausdorff-Besicovitch dimension is of little practical value. For example, very seldom can one check *every* δ-cover, which, according to the definition, is required. In practice one therefore uses what is called the

box-counting dimension, sometimes somewhat confusingly also called the fractal dimension. It is defined in a way similar to the Hausdorff-Besicovitch dimension, the difference is that it uses only boxes instead of arbitrary covering sets and therefore has the property that it is easy to calculate. Still, it almost always coincides with the Hausdorff-Besicovitch dimension. But it has some drawbacks, mainly two: firstly, it is difficult to handle the box-counting dimension mathematically, it is much less elegant than the Hausdorff-Besicovitch dimension. Secondly, it has some undesired properties. For example, the set $F = \{ 0, 1, 1/2, 1/3, ...\}$ is a compact set with box-counting dimension 1/2, despite being a countable union of points on the real axis.

For practical reasons however, like in calculations of the fractal dimension of clouds, these are not serious objections since they require quite pathological cases not present by chance.

There is also something called the *self-similarity dimension*; it is a heuristic method for calculating the Hausdorff-Besicovitch dimension making use of the scaling property, which I state without proof:

$$H^s(\lambda F) = \lambda^s H^s(F).$$

We recognise the scaling property from the ordinary volume, area and length measures; if we scale a flat surface with λ, its area gets λ^2 larger, and similarly for a body, its volume increases with λ^3.

The method only works when the set is self-similar, or statistically self similar which most often is the case in the patterns of nature. It goes like this: Let F be a set such that it contains n λ-scaled disjoint copies of itself. Because measures are additive, we have that the Hausdorff measure of F is equal to n times the Hausdorff measure of F scaled with λ:

$$H^s(F) = n\, H^s(\lambda F) = \{\text{scaling property}\} = n\, \lambda^s H^s(F).$$

Dividing by $H^s(F)$ on both sides and solving for s yields

$$s = -\frac{\log(n)}{\log(\lambda)}.$$

Thus - log(n) / log(λ) is called the self-similarity dimension. One realises that this derivation from the Hausdorff measure is heuristic and in fact a little bit suspicious if one remembers that the Hausdorff measure is almost always zero or infinite. One can therefore question if it is appropriate to divide by this quantity. In some cases this can be motivated, in other cases it can't, but this doesn't prevent the self similar dimension of being widely used.

***Example**: The Cantor set.* The so called middle-third Cantor set C is constructed iteratively by the following steps:

0) Start with the unit interval
1) Remove the middle third of the remaining connected intervals.
2) Repeat 1).

The resulting set will consist of uncountable many disconnected points and thus have the topological dimension $\dim_T C = 0$. The first five stages in the construction of C is shown in Figure 2-1. The self-similarity dimension is easy to calculate in this case, because if we scale up the left side of C three times we recover the hole set again, so $n = 2$ and $\lambda = 1/3$ makes the self-similarity dimension $\log(2) / \log(3) \approx 0.6309297$. It is not difficult to calculate the Hausdorff-Besicovitch dimension rigorously in this case to show that it coincides with this self-similarity dimension as well as the box-counting dimension.

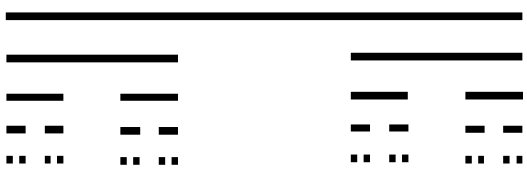


Figure 2-1. *The first five construction stages of the Cantor set.*

2.5 Definition of a fractal

Now to the definition of a fractal. *A fractal is a set whose Hausdorff-Besicovitch dimension differs from the topological dimension*. The Cantor set above is a good example of a fractal, since its topological dimension is zero (it consists of totally disconnected points). The Hausdorff-Besicovitch dimension is always larger than or equal to the corresponding topological dimension, thus, for a fractal set F, $\dim_T F < \dim_H F \leq D$, where D is the dimension of the embedding Euclidean space. There exist sets F with $\dim_T F < D$ but $\dim_H F = D$. If $\dim_T F = 1$ in such a case, F can be viewed as a very entangled curve that is filling the space and is often called a *space filling curve*. A famous example of such a curve is the Peano curve. For further discussion of the fractal dimension I refer to Falconer (1990).

3. Fractal cloud models

When you look at pictures of interstellar clouds, it may strike you that these clouds rarely show any intrinsic scale, blowing up the picture the overall structure looks quite the same. And as you go further down in scale, until you reach the resolution limit of the picture, the cloud doesn't look particularly smoother. This is a clear indication of the non-differential nature of the cloud structure. Instead, this self similar structure gives us a hint of another geometry which might prove useful in this case: the fractal geometry.

3.1 The image and the graph of a cloud

At first it is not at all evident, how one should apply the concept of a fractal to this seemingly self similar image of a cloud. A fractal, as defined in chapter 2, is a subset of points of an *n*-dimensional Euclidean space. An image, on the other hand, is a function $f(x,y)$ of two spatial variables, x and y. *All* points in a rectangle belong to the image, and an intensity value is assigned to each point. The trick in describing an image as a set, is to consider the subset defined by the surface $f(x,y)$ in a three-dimensional space, called the *graph* of $f(x,y)$. This way we may analyse this set using the ordinary tools used for fractals, i.e. the dimension of the set etc.

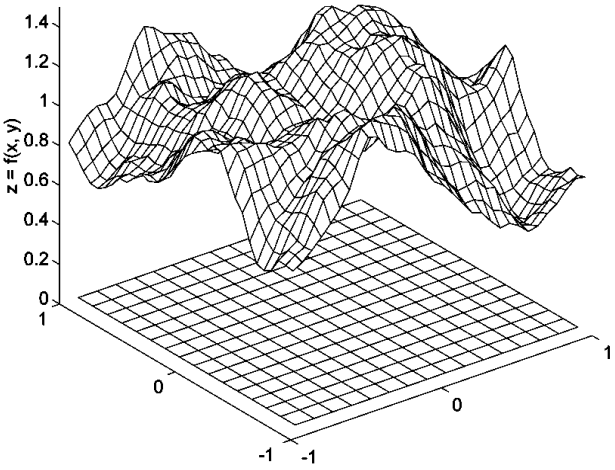


Figure 3-1. *An image viewed as a surface embedded in three Euclidean dimensions.*

Note that it is not the cloud itself one assigns a fractal structure this way; it is its projection on the plane perpendicular to the direction towards the cloud one models. Of course, the projection of a cloud is not as interesting as the cloud itself, but observations of the true three-dimensional structure of a cloud is currently not possible (and maybe will never be), so one has to start from the projection.

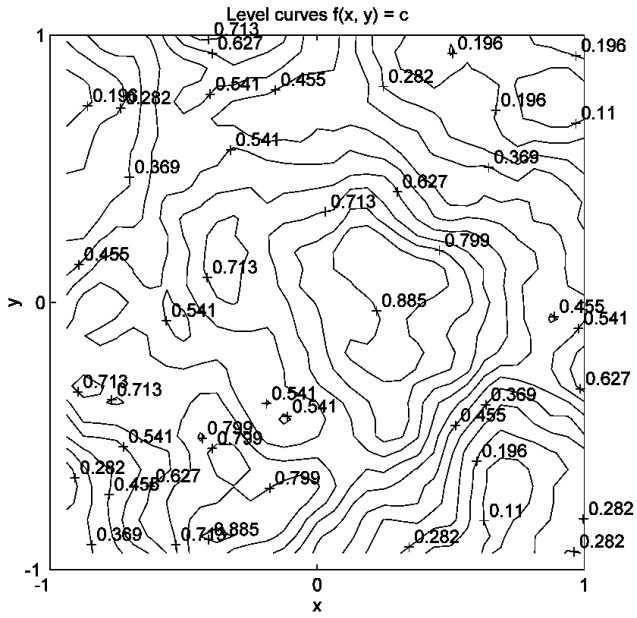


Figure 3-2. *An example of a level plot. The zero sets are labelled with their level values.*

Exploiting the fact that a surface is a rather special kind of a subset, one has several additional techniques to explore it. One often used technique is that one can plot the so called level curves, the sets $f(x,y) = c$, where c is a constant. These sets are also called *zero sets*, and in an image they are subsets of $\boldsymbol{R}^2$. Mandelbrot conjectured in *The fractal geometry of*

nature (1982), chapter 12, that if the set $f(x,y)$ is a fractal of dimension D, then its zero sets have a fractal dimension of $D - 1$. Furthermore, for each closed curve i of a zero set we can determine its fractal dimension by the *area-perimeter* relation $P_i(s) = \rho\, A_i(s)^{(D-1)/2}$, where $P_i(s)$ is the perimeter of the curve i as measured with a ruler of length s, $A_i(s)$ is the enclosed area measured with the same ruler, and ρ is a constant called the *prefactor*. For each ruler of size s one can plot the perimeters of all closed curves i as a function of their enclosed areas in a logarithmic diagram, and one should then find all points lie on a straight line with slope $(D - 1) / 2$. In practice one combines data from several ruler sizes and zero sets before doing the regression to find the slope, in order to lower the error. This method is widely used by astronomers trying to determine the fractal dimension of images of interstellar clouds, see for example Vogelaar & Wakker (1994) or Bazell & Désert (1988).

I have not been able to figure out why the area-perimeter relation is used so extensively, nor what its advantages are over the straight box counting algorithm. Maybe, it is less sensitive to noise or gives rise to a more effective algorithm or maybe it is just easier to implement.

A more recent method of analysing graphs of functions is the Δ-*variance* presented by Stutzki et al. (1998) as an n-dimensional generalisation of the successful *Allan-variance* previously used in analysis of 2-dimensional graphs. I will not describe the method in detail, see Stutzki et al. (1998) for a thorough exposition, but only mention that it exploits the fact that, if you convolve the n-dimensional data "image" with a special n-spherical symmetric function depending on a characteristic scale L, then the variance of the result will depend in such a way on the parameter L, that one can sometimes use this to determine the power spectrum of the data "image". That is, the power spectrum as a function of the frequency vector length, where the frequency is the Fourier frequency. In the case of an image of a *fractional Brownian motion* (fBm) described below, this power spectrum follows a power law and the Δ-variance works very well in determining the exponent of the power law. Furthermore, it is stable and readily discriminates between Gaussian noise and fBm.

3.2 Fractional Brownian motion

Traditionally, Brownian motion[1] describes the irregular path of a small particle in a soluble medium, bumped around by the molecules. This irregular motion had been observed in microscopes by the biologist Brown in 1827 and was first explained by Einstein 1905. The Brownian motion has a very simple mathematical formulation in one dimensional space. We start with a particle at the origin and plot the displacement of the particle, the distance from the origin, as a function of time. The essential property of this graph is that the mean square displacement depends *linearly* on the time. That means that, on the average, the distance of a particle after time Δt in is $\propto \sqrt{\Delta t}$ from where it started. This process has several names: *Brownian motion*, *Wiener process* and *1/f² noise* or *Brown noise*, due to the fact that the power spectrum depends on the frequency f as $1/f^2$.

[1] *For further discussions on Brownian motion and fractional Brownian motion, I refer to Mandelbrot (1982), Peitgen & Saupe (1988) and Peitgen, Jurgens & Saupe (1992).*

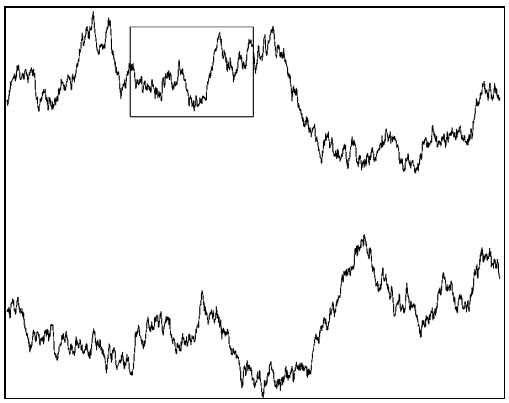


Figure 3-3. *The part inside the box in the upper Brownian curve is scaled and shown in the lower curve. Statistically, the lower curve is identical to the upper one.*

An interesting feature of this graph of Brownian motion is that it is self similar. Not exactly self similar, but rather *self affine*, since scaling time four times and the displacement twice yields a graph that has exactly the same statistical properties (in a *self similar* case we scale both axes equally). In spite of not being *self similar*, this graph is a fractal curve, and the fractal dimension can easily be derived using the box counting dimension. I will demonstrate that in section 4.1. The dimension of the graph should be greater than the topological dimension of a line, 1, and lesser than the Euclidean dimension of the plane, 2. Indeed, the fractal dimension of Brownian motion turns out to be 1.5.

The concept of Brownian motion seems so far pretty far from modeling interstellar clouds; we will need to generalise the concepts a bit. For instance, we would like to be able to change the fractal dimension of the graph arbitrarily between 1 and 2, using some parameter. This can be achieved by changing the statistical properties such that the mean square displacement depends on time as Δt^{2H}, where H is a parameter, often called the *Hurst exponent*, varying between 0 and 1. Doing this, we will have a power spectrum $\propto 1/f^{\beta}$, where $\beta = 1 + 2H$, and a fractal dimension $D = 2 - H$. We see that $H = 0.5$ corresponds to the Brownian motion. This generalisation is called *fractional Brownian motion* (fBm). Figure 3-1 provides an example of an fBm with two spatial (”time”) axes (x and y) and one intensity (”spatial”) axis (z).

We still need to generalise this concept to more dimensions. The obvious generalisation, making the particle move in more spatial dimensions, is of little interest to us, since we would only get a fractal path. Instead we extend the ”time” dimension into more dimensions, which I from now on call *spatial* dimensions, changing the name of the original spatial dimension at the same time to the *intensity* dimension. The generalisation can be done either by requiring that the usual Euclidean distance in the spatial dimensions fulfils the same relation to the displacement in the intensity dimension as the ordinary time in the one-dimensional process, or by simply requiring the power spectrum to fulfil the power law property $1/|f|^{\beta}$ (the phases being completely random) where certain constraints are put on β. These approaches turn out to be mathematical identical, and the relations are:

$$d(|\Delta t|) \propto |\Delta t|^{H}$$
$$P(|f|) \propto |f|^{-\beta}$$
$$\beta = n + 2H$$
$$D = n + 1 - H$$

Here n is the Euclidean dimension of the spatial space, and the graph is thus embedded in an Euclidean space of dimension $n + 1$. Both f and Δt are now n-dimensional vectors, $d(t)$ is the displacement in intensity between points at spatial distance t between each other, $P(f)$ is the power spectrum and, as before, D is the fractal dimension and we restrict H to the domain $0 \leq H \leq 1$. For the example of an image, we have $n = 2$, $2 \leq D \leq 3$ and $2 \leq \beta \leq 4$.

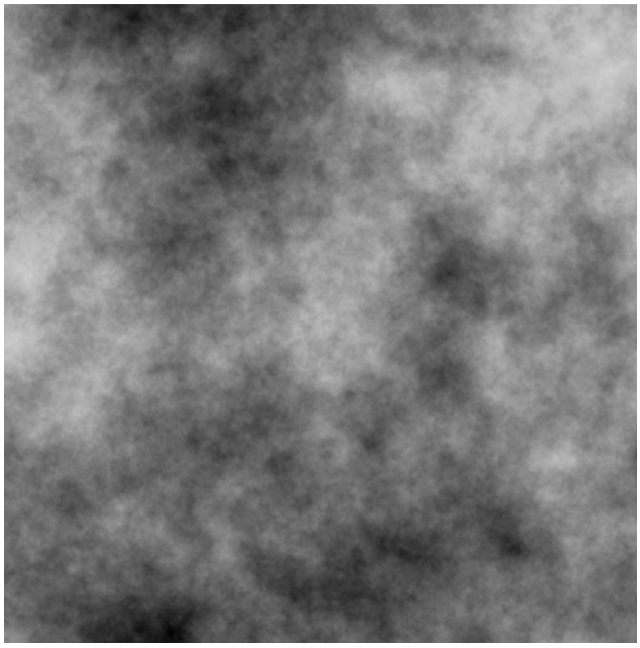


Figure 3-5. *An example of an image of a cloud modeled as fBm. A similar image is reproduced in colour in the back as colour plate 3.*

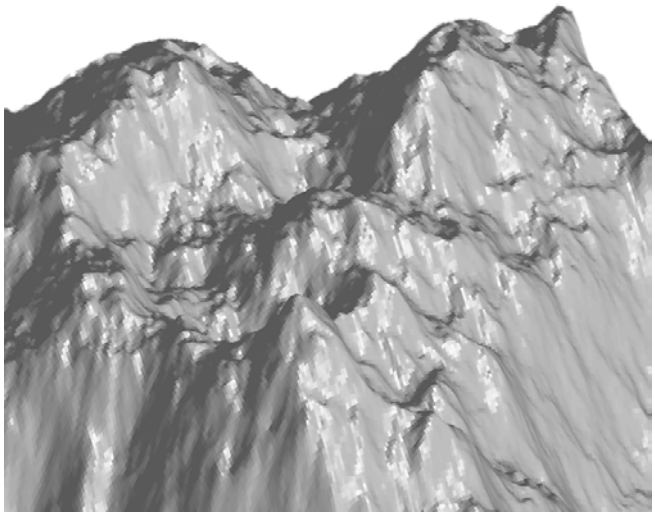


Figure 3-4 *fBm can also be used to produce realistically looking mountains. Here I used the developed package* `fBm` *together with Matlab to render the image.*

3.3 A multifractal model

As we shall see, fBm models appear very successful in describing what we actually see in interstellar clouds. But this doesn't prevent us from further explore models incorporating the observed self similar structure of clouds. In an fBm for instance, the scaling properties are independent of the displacements; that is, it does not matter how far you are from where you started, the scaling properties are still the same. One could wonder what would happen if we relaxed this condition and let the scaling properties be a function of the displacement. The structure would still be self similar in some sense. However, the fractal dimensions of the zero sets would not be constant but be functions of their levels. We would have a *multifractal*, with a spectrum of fractal dimensions called the*Hölder spectrum.*[2]

One essential property of the multifractals is that they are *not* defined as a subset of an Euclidean space; rather, they are defined as a *measure* on it. A measure is (as mentioned in chapter 2) a function from subsets of a set to the (positive) real numbers. This means that for each subset[3] of a set, the measure assigns a number to the set. A well known measure is the *Lebesgue* measure used to determine the length, area and volume of sets.

Another way of seeing multifractals as compared to fractals is that they have some sort of greyness instead of being black-and-white, as the fractals. Each point in the multifractal set has some sort of a weight. This slightly different view of the multifractals as compared to the fractals make them ideal for modeling clouds. There is namely no need for taking the

---

[2] *To read more about multifractals, see an appendix by Mandelbrot in Peitgen, Jurgens & Saupe (1992) or McCauley (1990)*

[3] *Often not* ***all*** *subsets, but rather all measurable subsets (depending on the measure), which form a* σ*-algebra.*

supporting set of the cloud into higher dimensions; it is sufficient to look at the cloud in the three space dimensions and letting the measure be mass or number of atoms. Each subset of the cloud is thus assigned a mass. This is a quite natural approach.

One often used multifractal model is called *multiplicative cascades*. It is iteratively defined as follows:

1) Start with some set.
2) Assign the set some measure of (preferably unit) mass.
3) Partition the current set into finite many parts.
4) Assign each part a positive number such that the sum of the numbers equals unity.
5) Let the measure of each part of the current set be the mass of the current set multiplied with the number assigned to the part.
6) For each part of the partition of the current set, do 3)

Step 4) assures that the total mass of the set is conserved. To make the measure defined by the above procedure a multifractal measure, we have to add some restrictions on it. A multifractal measure is self similar, or at least statistically self similar. We thus have to restrain 3) and 4) to have some regularity, i.e. not to depend on the level of iteration.

Because in a practical application we want to generate random numbers to assign the parts in a partition such that the sum of the numbers equals unity, we have to consider a probability distribution on a hyper surface in the parameter space of dimension equal to the number of parts in the partition, the hyper surface restrained by the equation $x_1 + x_2 + \ldots + x_n = 1$, where x_i is the number assigned to part i. Any arbitrary probability distribution on this surface is of course possible, but if we are to model isotropic clouds the distribution should be symmetrical with respect to all parameters x_i. Of course, the number of parts in a partition could also be stochastic in a really complex model.

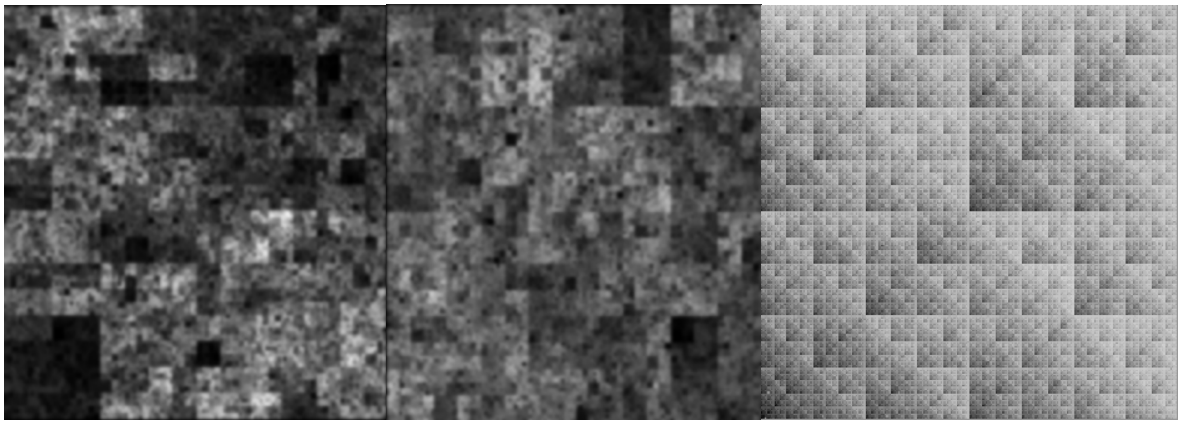


Figure 3-6. *Three examples of multiplicative cascades: the first with a uniform probability distribution, the second with a multivariate and the third with a rather skew (but not special in any other way) distribution.*

Since multiplicative cascades are so simple to implement, I tried out two variants in the mathematics and numerical analysis computer program Matlab. In the first I partitioned a square in exactly four parts each iteration and had a uniform distribution over the hyper surface (now embedded in four dimensional Euclidean space). The second model is the same as the first, except that I chose a symmetrical multivariate normal distribution over the hyper surface with cut-off. I also tried some skew probability distributions, resulting in variants on the Sierpinski gasket[4]. The results of these simulations are shown inFigure 3-6.

As can be seen there are clear artefacts from the construction procedure left in the images. In my opinion, they look too artificial to resemblance clouds. There must be other means of generating multifractals than the naive outlined in this section, but I haven't found any books on the subject and I haven't had time to explore the research articles in the area more thoroughly. So I leave the multifractals for now, concentrating on the fractal models instead.

[4] *For a description of the Sierpinski gasket, see any book on fractals in the reference list.*

4. The projection of fractals

What we see, when we look at the image of a molecular cloud, is not the three-dimensional structure, but the projected image of it. Analysing this two-dimensional projection does not guarantee that we will be able to reconstruct the full 3D structure. Specifically, since we are interested in describing the cloud structure with a fractal model, we ought to examine how a projection affects the fractal dimension. It is not evident that there exists a relation between the fractal dimension of a fractal and the fractal dimension of its projection and indeed, in general, there is no such relation. To see why, it is sufficient to consider the Cantor set embedded in a two-dimensional Euclidean space: along one axis its projection is again the Cantor set, along another it is merely a point. The corresponding fractal dimensions are $log(2) / log(3)$ and 0.

The problem arises from the fact that the Cantor set is not isotropic when viewed in two dimensions, i.e. its projection does not look the same from all angles. What about interstellar clouds? By eye inspection, they look isotropic. Even though we have no reasons to believe that interstellar clouds all have a special orientation relative to us, they all look quite the same. Furthermore, there have been studies that reveal that most of the clouds have about the same projected fractal dimension. Hence, we conclude that a three-dimensional fractal model of a molecular cloud should be, at least to a first approximation, isotropic. When it comes to isotropic fractals, it is still an open question whether their projections uniquely determine their structures. Beech (1992), who made some physical experiments, as well as Hetem & Lépine (1993), who made computer simulations, found that their results were consistent with $D_2 = D_3 - 1$, where D_3 is the fractal dimension of the three-dimensional structure and D_2 is the fractal dimension of the projected, two-dimensional structure.

4.1 The projection of optically thin fBm

For a special class of isotropic fractals, namely the fractional Brownian motion (fBm), I have been able to show that there is a special relation between the fractal and its projection. First a lemma.

Lemma. *Let d_1 and d_2 be equivalent metrics on $\boldsymbol{R}^n$ and θ a function providing that equivalence. Let $F \subset \boldsymbol{R}^n$ be a fractal set with the dimension D in the metric d_1. Then, it also has the fractal dimension D in the metric d_2.*

Proof: This is a special case of theorem 1.3 in chapter V in Barnsley's *Fractals everywhere* (1993). ■

In order to help to better visualise the propositions in arbitrary dimensions, I start with stating and proving the first in 2 + 1 dimensions.

Proposition 1'. *Let F be an fBm image with Hurst exponent H. Then it has the dimension dim F = 3 - H.*

Proof: I use that the largest deviation of a sample is proportional to the standard deviation. First, normalise the image so that its boundaries as well as the intensity range are of unit size. According to the lemma, this does not change the fractal dimension of the image. Partition the image and the intensity in b equal intervals so that we have b^3 boxes. To determine the box

counting dimension, we want to calculate how many boxes that contain a part of the fractal set. It is clear that since this image is a surface in $\boldsymbol{R}^{2+1}$, for each part (square) of the image, we have at least one box that intersects the intensity surface. To cover the hole intensity range in that part, we need the difference between the maximum and the minimum intensity and then multiply this difference with the number of intervals per intensity unit, *b*. The difference between the maximum and the minimum in a square is in the mean equal to two times the largest deviation Δ from the mean in a part. Because of the fBm structure, this is $\Delta = c / b^H$, where *c* is some constant. So, we have that the number of boxes needed to cover the set *F* is $N(\delta) = 2b\Delta b^2 = 2cb^{3-H}$, $\delta = 1/b$. This gives the box counting dimension

$$\dim F = \lim_{\delta \to 0} - \frac{\log(N(\delta))}{\log(\delta)} = \lim_{b \to \infty} \frac{\log(2cb^{3-H})}{\log(b)} = 3 - H. \blacksquare$$

Now to the *n*-dimensional case:

Proposition 1. *Let F be a finite fractal surface embedded in the Euclidean space* $\boldsymbol{R}^{n+1}$ *described by an fBm with Hurst exponent H. Then the fractal dimension of the surface dim F = n + 1 - H.*

Proof: I again use that the largest deviation of a sample is proportional to the standard deviation. I assume that the first *n* dimensions are spatial and that the last is the intensity dimension. Because the set is finite, we can enclose it with a finite rectangular box. Because of the lemma we can normalise the scale so that the box sides are of unit size and partition all sides with *b* intervals. This gives rise to b^{n+1} smaller boxes. Now, count the boxes needed to cover *F*. Because this set is a surface, there is at least one box that contains a part of the set for each part of the *n* first (spatial) dimensions. The number of boxes needed for each such part thus only depends on the last dimension, and it is the difference between the largest positive and negative deviations from the mean in that part times the number of space intervals per space unit, *b*. Because the largest deviation Δ is proportional to the deviation given by c' / b^H (due to the fBm structure) we have $\Delta = c / b^H$, where *c* and *c'* are some constants. The number of boxes for each part is thus (in the mean) $2\Delta b = \frac{2cb}{b^H} = 2cb^{1-H}$. This makes a total number of filled boxes $N(\delta) = 2cb^{1-H}b^n = 2cb^{n+1-H}$, $\delta = 1 / b$. Calculating the box counting dimension we get:

$$\dim F = \lim_{\delta \to 0} - \frac{\log(N(\delta))}{\log(\delta)} = \lim_{b \to \infty} \frac{\log(2cb^{n+1-H})}{\log(b)} = n + 1 - H. \blacksquare$$

A proof of proposition 1 was already given by the founder of fBm, Mandelbrot.

Proposition 2. *Let F be a finite fractal embedded in* $\boldsymbol{R}^{n+1}$ *Euclidean space described by an fBm with Hurst exponent* $0 \le H \le 0.5$ *and thus fractal dimension* $D_{n+1} = n + 1 - H$. *If one projects along one time axis, then the resulting projected fractal will have a fractal dimension of* $D_n = n - 0.5 - H$, *and thus* $D_n = D_{n+1} - 1.5$.

Proof: Normalise and partition a surrounding box as in proposition 1. Project this partition as well. Since each part in the projected partition corresponds to the sum of *b* boxes in the original partition with independent deviations $\Delta_{n+1} = c / b^H$, the deviation in a part of the projected partition will be the original deviation divided by $\sqrt{b}$, according to probability theory

(summation of independent equidistributed stochastic variables) and $\Delta_n = c / b^{H+0.5}$. Thus the number of boxes needed to cover the projected set is

$$N(\delta) = cb^{1-H-0.5}b^{n-1} = 2cb^{n-0.5-H},$$

and the box counting dimension is

$$\lim_{\delta \to 0} - \frac{\log(N(\delta))}{\log(\delta)} = \lim_{b \to \infty} \frac{\log\left(2cb^{n-0.5-H}\right)}{\log(b)} = n - 0.5 - H . \blacksquare$$

For $0.5 < H \leq 1$ we have that the projection smoothens the fractal structure so much that the projection is no longer a fractal, but of a fractal dimension equal to the topological. This result stands in strong contrast to the often stated but never proven conjecture $D_n = D_{n+1} - 1$.[5] One consequence of proposition 2 is that if interstellar clouds really are fractals well described by fBm, then their projections should have fractal dimensions that vary between 2.0 and 2.5, and the zero sets should correspondingly have fractal dimensions between 1.0 and 1.5.

An alternative and elegant proof of proposition 2 has been given independently by Stutzki et al. (1998). Their proof, however, is more indirect as they make use of the power law power spectrum of the fBm, looking at the Fourier space.

4.2 The projection of fBm with appreciable optical depth

The theory above applies only when the three-dimensional cloud we're looking at is optically thin in the relevant spectral range or line. What if the approximation of a transparent medium can't be made?

Fortunately, the lemma comes to a rescue again. If the emission from the medium is not heavily saturated[6], we can assume, as is usual, that the observed intensity is a function only of the integrated emission along the line of sight (column density), and that this function is monotone[7]. This function will provide an equivalence between the two metrics on 1) the emission projection space and 2) the observed intensity space. Thus the fractal dimension will not change under this transformation. However, if the three-dimensional cloud is well described by an fBm with $0 \leq H \leq 0.5$, then the resulting projection will no longer be an fBm. In particular, the variance will no longer depend only on the spatial co-ordinates but also on the intensity. The zero sets will still have the same dimension, but in the perimeter-area method of determining the fractal dimension one will no longer be able to combine different zero sets before doing the regression, unless one knows the observed intensity as a function of the intrinsic column density. If one goes the other way around and assumes that the three-dimensional structure of the cloud is well described by an fBm, this actually provides a method to determine the observed intensity as a function of the intrinsic column density.

In the case of optical thick clouds the above model is far too simple and one has to solve the full radiation transport equations.

---

[5] *See Falgarone, Philips & Walker (1991), Beech (1992) or Hetem & Lépine (1993).*

[6] *In astronomical nomenclature, we may assume, for example, that* $\tau \leq 2$.

[7] *Note that* ***monotone*** *doesn't necessarily mean* ***linear***.

The method of Δ-variance is unfortunately not very suitable for determining the fractal dimension of non fBm structures. Ossenkopf et al. (1998) report that, in case of an optically thick fBm cloud, simulations showed that the slope of the Δ-variance as a function of scale length will be lower than expected for an optically thin fBm. The relations between Δ-variance and optical depth are not yet very well known, but work on this is currently being done by that group.

5. Comparing models with observations

In order to discriminate between models, we need to compare them with observations. In our case, we need a method to quantitatively analyse two-dimensional images of interstellar clouds and compare them with projections of model clouds. Fortunately, Adams and Wiseman[8] have developed a method to do such analysis. I will now shortly review their approach, which makes use of some mathematical formalisms.

5.1 The space of cloud characteristics

Let the characteristics of a cloud (whatever they may be) be a point of some abstract topological space of all possible characteristics. To discriminate between two different clouds would be the same thing as discriminating between two points in that abstract space. To do this, we need to introduce a *metric*, that is, a distance function that tells us how close two points (clouds) are in that space. We will not, however, be able to determine *all* possible characteristics of a cloud; we have to be content with a projection of the abstract space onto some subspace of the characteristics we *can* determine. This projection may cause two different points in the original space to unite in the projected space. A metric on our projected space will thus be a *pseudo*-metric in the original space; the distance between two different points in the original space may very well be zero in our pseudo-metric, if their projections coincide. The other way around, it is clear that two different points in the projected space are different also in the original one.

The use of a pseudo-metric instead of a metric is not totally unfortunate; in this way we may concentrate on the characteristics of clouds we find essential. An example of a characteristic of a cloud we may not find essential is the distance of the cloud from the Sun. Ignoring this characteristic will cause two otherwise identical clouds be identical also in our pseudo-metric. The pseudo-metric thus groups together clouds we consider alike.

5.2 Some pseudo-metrics on the space of characteristics

Adams and Wiseman consider mainly four different characteristics: distribution of density, area, topological components and filaments. These are characteristics that apply to any two-dimensional map or image.

Let f(x,y) be the intensity of the image at co-ordinates x and y, and Θ be the Heaviside (step-) function

$$\Theta(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}.$$

We begin the definition of the pseudo-metric by assigning a function to each characteristic.

[8] *See Adams (1992), Adams & Wiseman (1994) and Wiseman & Adams (1994)*

The fraction of the mass m that is above a certain density level Σ is

$$m_f(\Sigma) \equiv \frac{\int f(x,y)\Theta\big(f(x,y)-\Sigma\big)dxdy}{\int f(x,y)dxdy},$$

where we integrate over the whole image. Similarly, the area of the fragments over a certain density level Σ is

$$A_f(\Sigma) \equiv \frac{\int \Theta\big(f(x,y)-\Sigma\big)dxdy}{\int dxdy}.$$

In the case of component characteristics, let

$$n_f(\Sigma) \equiv \text{the number of topological (connected) components above } \Sigma.$$

The filament associated function is slightly more complicated. We want to quantify the tendency of structures to be filamentary. Begin by, as in chapter 2, defining the diameter $|U|$ of a set U to be the largest of all distances between any two points in it. If the set was a disc, its area would be $\pi|U|^2 / 4$. If the set isn't a disc, its area will be strictly smaller. In fact, the more elongated the less the area. This motivates the definition of the filament index:

$$F = \frac{\pi|U|^2}{4A},$$

where A is the area of the set. Thus the more elongated the set, the higher the filament index.

Adams and Wiseman consider two possible functions to associate with the filament characteristic. One is the "unweighted":

$$f_f^{(a)}(\Sigma) \equiv \frac{1}{n_f(\Sigma)}\sum_j F_j,$$

where we indexed the filament index for the different topological components above threshold Σ with j, and the other is the "weighted":

$$f_f^{(b)}(\Sigma) \equiv \frac{1}{n_f(\Sigma)}\sum_j \frac{A_j}{\langle A\rangle_\Sigma} F_j,$$

where A_j is the area of component j and $\langle A\rangle_\Sigma$ is the average area of the components above threshold Σ.

So far, we have assigned a function to each characteristic that depend on the threshold level Σ. Adams and Wiseman now assign a metric to each characteristic by making use of the L^2 norm, the metric

$$d_\chi(f,g) = \|f-g\|_2 \equiv \left[\frac{1}{\langle\Sigma\rangle}\int_0^\infty \big|\chi_f(\Sigma)-\chi_g(\Sigma)\big|^2 d\Sigma\right]^{1/2},$$

where $<\Sigma>$ is some (arbitrary) reference density, χ is any of the corresponding characteristic functions and f, g are two intensity functions corresponding to images of the clouds.

Note that we do not have a metric on the combined space of the above characteristics, only on each subspace containing one characteristic. There is no unique way to combine the given

metrics into one single metric, but that's a fact we have to live with: we can only say that two clouds are close with respect to a specific characteristic.

5.3 Some properties of the pseudo-metrics

The nice thing about these characteristic functions and metrics is that they scale in a simple way. Adams and Wiseman summarised their properties in three theorems:

Theorem 1. *If we scale the intensity of an image f with a constant c, that is, cf, the function* χ_f *assigned to one of the above characteristics scales as* $\chi_{cf}(\Sigma) = \chi_f(\Sigma / c)$.

Corollary. $d_\chi(cf, cg) = \sqrt{c} d_\chi(f, g)$.

Theorem 2. *If we scale the intensity of an image f with a monotone function F, that is F(f)f, the function* χ_f *assigned to one of the above characteristics scales as* $\chi_{F(f)f}(\Sigma) = \chi_f(\Sigma')$, *where* Σ' *is the root of the equation F(f)f =* Σ'.

Let an affine transformation of an image be a scaling followed by a rotation followed by a translation of the spatial co-ordinates *x, y*. If the scaling of the spatial co-ordinates are equal for both axes, we call the transformation a similitude.

Theorem 3. m_f, A_f, *and* n_f *are invariant under affine transformations while* $f^{(a)}{}_f$ *and* $f^{(b)}{}_f$ *are invariant under similitudes.*

These theorems and simple proofs of them are stated in Adams & Wiseman (1994).

If the structure of clouds can be well described by fractals, as many observers claim, then we have one additional characteristic, namely the fractal dimension. It is invariant under all transformation of theorem 1, 2 and 3, which is shown in theorem 1.3 in chapter V in Barnsley's *Fractals everywhere* (1993).

5.4 Determining characteristics of fBm

These are the tools we can use to compare the models with observations. It would be interesting to know how well we can model the characteristics of interstellar clouds using, for instance, fBm. To evaluate this question, I implemented the mass, area and the number of components characteristics functions m_f, A_f and n_f, and applied them to fBm with Hurst exponents in the range 0 to 1. Since results for real clouds from the IRAS survey are published in Wiseman & Adams (1994), I was able to make comparisons.

One property of the mass functions is that they always decrease with Σ. Another is that the mass function always is larger than the area function, which, with a little thought, isn't difficult to realise. One property that isn't general but is very apparent in the derived functions from observations, is that the mass and area functions are *smooth*. As Wiseman and Adams point out in their paper, there is no *a priori* reason why. If the clouds were clumpy, for instance, there should be discontinuities which are not present in the observations. Of course, if the clumps were very fuzzy (or smaller than the resolution), then the mass and area functions would still be continuous.

What about fBm? A typical fBm mass and volume profile looks like Figure 5-1. As an example cloud I consider the typical cloud of Lupus at approximate equatorial co-ordinates 15^h 40^m, -35° 00'.[9] Note that while Σ in the fBm profile is normalised to the interval [0,1], in the Lupus profile it is not. This is not significant since we can stretch the fBm scale as we like without changing its properties.

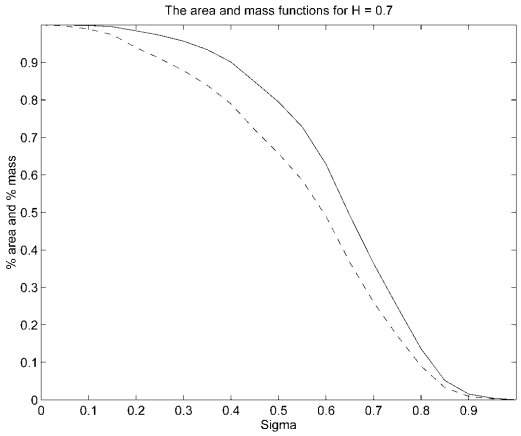


Figure 5-1. *The fraction of area over a certain level in an unbroken line and the fraction of mass as dotted line, as determined for a theoretical fBm.*

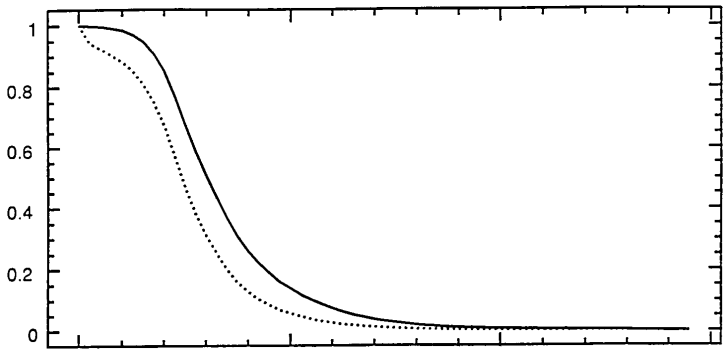


Figure 5-2. *The fraction of area over a certain level in an unbroken line and the fraction of mass as dotted line, as determined for by Wiseman, Adams (1994) for a cloud in Lupus. The horizontal Σ-axis has no marks, but it doesn't matter because we can scale the Σ-axis as we wish anyway.*

For fBm of different Hurst exponents but with the same normalisation, I noticed that the mass and area functions are slightly shallower for higher Hurst exponents (and thus lower dimensions). When I fixed the phases and only varied the Hurst exponent for a fixed fBm image, the derivative in the steepest part was a power law of the Hurst exponent.

In general, the number of components function is not monotonic and not continuous. Because of noise, Wiseman and Adams decided to only count components with more pixels than 3 and 9, respectively. This is somewhat arbitrary and yields different numbers of components depending on the (pixel) resolution. One should therefore be careful to only compare maps of the same resolution, or develop means to compensate for resolution differences. Choosing an fBm image with the same 400 by 400 pixels resolution as the Wiseman and Adams cases, I calculated the number of components function with the same 3 and 9 pixels restriction on the least number of pixels in a component. A comparison between the number of components function profiles can be seen in Figure 5-3 and Figure 5-4. Again, the intensity scale (Σ) is not fixed but can be arbitrarily scaled. However, the number of components is invariant under scaling. In particular, the largest number of components in a profile is invariant, so this provides a tool to discriminate between intrinsic properties of models/clouds not depending on scale. Again fixing the phase space and only varying the Hurst exponent for a fixed fBm image, I studied the maximum number of components along a profile as a function of the Hurst exponent *H*. It turned out to be roughly an exponential law. In any

[9] *The map of Lupus used by Wiseman and Adams is a column density map as derived by Wood, Myers & Daugherty (1994) from IRAS 100* μm *and 60* μm *data.*

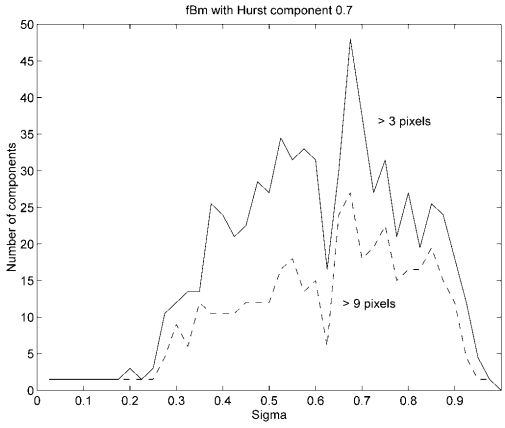


Figure 5-3. *The number of components as a function of level. The two curves correspond to 3 and 9 minimum number of pixels in a component, the unbroken and dotted line respectively.*

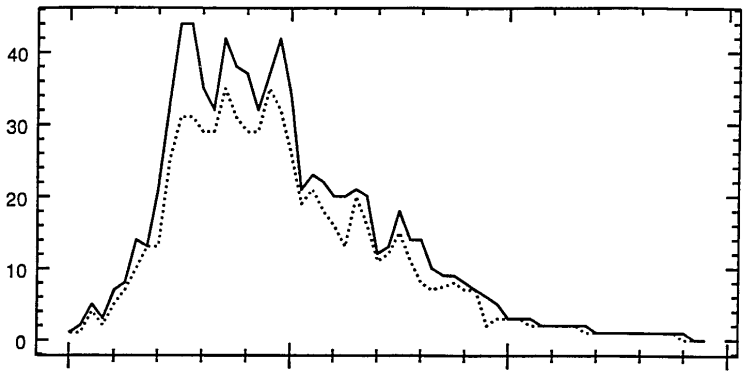


Figure 5-4. *This diagram shows the number of components of the cloud in Lupus as determined by Wiseman and Adams. The same note as in Figure 5-2 applies here: the scaling of the Σ-axis is of no importance.*

case it was far from being constant with respect to *H*, ranging from 275 to 10 components (see Figure 5-5).

I also tried to vary only the random phases and not the Hurst exponent of an fBm, to see how much the maximum number of components varied randomly. It turned out that the standard deviation is about 8 for $H = 0.5$. I did not check for other Hurst exponents. This shows that the maximum error in Figure 5-5 is *at least* 8 components.

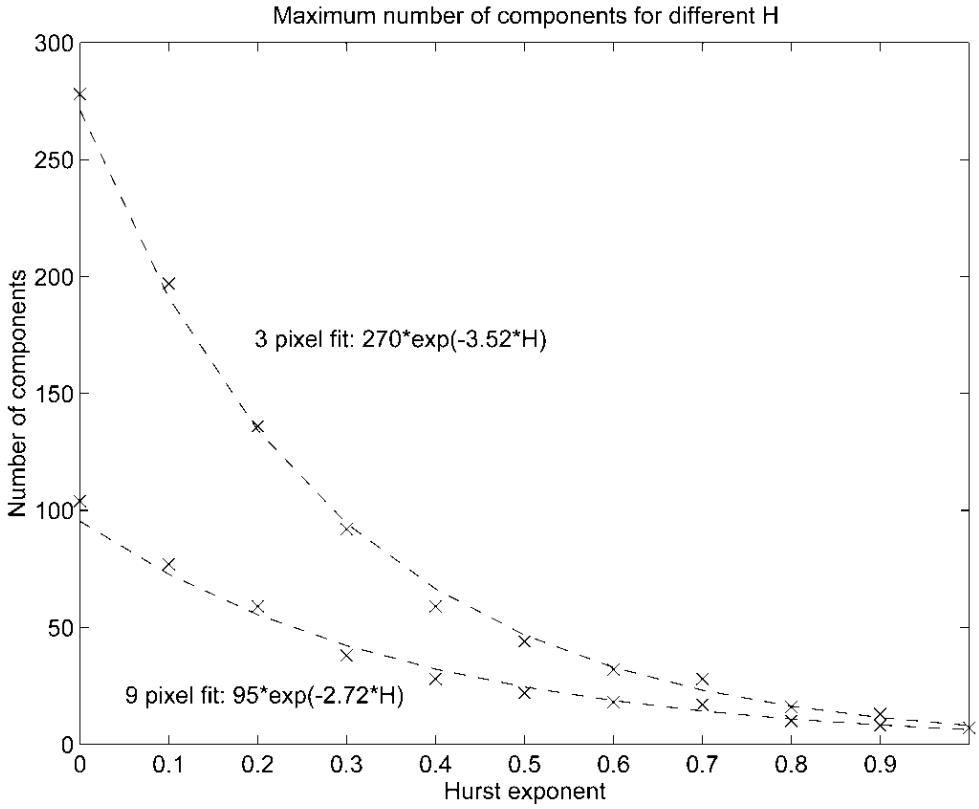


Figure 5-5. *I determined the maximum number of components for fBm with different Hurst exponents and found that the function can be approximated by an exponential law.*

6. Theoretical modeling of the IMF

The cloud mass spectrum of an fBm model of a cloud has already been investigated by Stutzki et al. (1998), who found that it was consistent with observed cloud mass spectra. In this chapter, I will examine how an IMF would look like if the mass distribution of a star forming cloud was fractal in the sense of an fBm. A simplification I will make is that the cloud is isothermal.[1]

I assume that the mass of a particular star comes from a particular region of the cloud with volume V, and that the mass of the star is proportional to the total mass of this region. The mass of this region will, of course, depend on the volume, as well as on where the region is located in the cloud, because the cloud is not homogenous. Because the gravitation law is spherically symmetric and an fBm mass distribution is isotropic, this region will be reasonable localised, i.e. not extremely elongated or disconnected (maybe even convex).

Now, consider a region randomly selected from all possible localised regions of volume V in an fBm cloud. The mass of such a region will be a stochastic variable, with an expectation mass that equals the mean mass of all possible localised regions of volume V, that is, the mean density of the cloud times the volume of the region. Because an fBm is a Gaussian process, this stochastic variable will be normally distributed. In order to simplify calculations, I will from now on assume that the localised regions we are considering here are well approximated by spherical regions, in the sense that the variance of the mass for a random region of volume V in the cloud is approximately the same as the variance of the mass for a random spherical part of volume V of the cloud.

Having made this approximation, I denote the stochastic variable of the mass of a region of volume V as $M^{(L)}$, where L is the radius of a sphere with volume V. The probability that this region will collapse is the same as that $M^{(L)} > M_J^{(L)}$, where $M_J^{(L)}$ is the critical mass for the region. For simplicity, I will assume that the critical mass is given by the Jeans mass. This will not be a crucial simplification, because, as will be seen, it does not change the slope of the IMF, only the cut-off for low masses. In an isothermal cloud, the Jeans mass is proportional to the Jeans radius, that is, $M_J^{(L)}$ scales as L.[2]

So, for a given radius L the mass distribution for stars will be proportional to the mass distribution for the mass of the region *given* that the mass is above the Jeans mass for that region. The number of these potentially star forming regions of size L scales as L^{-3}. Let us look on all possible regions in the permitted range L_0 to L_1. The probability distribution for a star to be of mass m is

$$N(m) = \frac{\int_{L_0}^{L_1} f_{M^{(L)}}(m)\Theta\left(m - M_J^{(L)}\right)L^{-3}dL}{\int_{L_0}^{L_1} L^{-3}dL},$$

[1] *This assumption has been motivated at some length by Benson & Myers (1989).*

[2] *In an isothermal cloud, both the Jeans mass and the Jeans radius are proportional to* $\rho^{-1/2}$*, thus they are also proportional to each other.*

where Θ also in this section is the Heaviside (step-) function

$$\Theta(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

and $f_{M^{(L)}}(m)$ is the probability density function for $M^{(L)}$.

Note that this derivation of $N(m)$ does not discriminate between large and small masses, large and small radii, large and small densities. $N(m)$ is the mass distribution among the stars if they are formed in clouds with small star forming efficiency, so small that the formation of a single star doesn't significantly influence the formation of later stars. Equivalently, maybe the mixing of the cloud is quick enough so that each time a star is to form, it meets a fresh cloud.

If the star forming efficiency reaches a high enough level, this assumption is no longer valid and one has to account for the loss of cloud mass; this problem is sometimes referred to the problem of *domain packing*. In this case we must also take into account the different rates at which stars with different initial region radii form; according to Jeans analysis the star formation rate will be proportional to $\rho_{region}^{-1/2}$, where ρ_{region} is the mean density of the region. This will favour high densities, and thus, small radii because of the larger density deviations on smaller radii.

However, in the case of extremely high domain packing, introduced by Richtler (1994) and further developed by Kaas et al. (1998), it is found that whatever the initial discriminating function, the resulting radius distribution will always, within the cut-off, tend to $\propto L^{-6}$ instead of the $\propto L^{-3}$ in the low domain packing (low star forming efficiency) case. Because now the regions are such that *all* regions result in stars, we have to condition the probability distribution with this fact. The resulting mass distribution after domain packing would thus be

$$N(m) = \frac{1}{\int_{L_0}^{L_1} L^{-6} dL} \int_{L_0}^{L_1} \frac{f_{M^{(L)}}(m)}{\int_{M_J^{(L)}}^{\infty} f_{M^{(L)}}(m) dm} \Theta\left(m - M_J^{(L)}\right) L^{-6} dL .$$

To determine the $N(m)$ now we only need the density distribution of the stochastic variable $M^{(L)}$. We noted before that its mean value should be the volume of the region times the mean density of the cloud, $V\rho_{cloud}$. Because of the fBm properties, the difference between two points at distance L from each other has a difference that is normally distributed such that the variance is $\propto L^{2H}$, i.e. the distribution $\in N(0, cL^{2H})$ (where c is a constant). To get the distribution for a region of radius L we integrate over all points in the region. The mass distribution for a region of radius L is thus

$$V\rho_{cloud} + \int_V X(\mathbf{r}) d\mathbf{r},$$

where $\mathbf{r}$ is the vector from the centre of the region, V is the set of the region, and each $X(\mathbf{r}) \in N(0, c|\mathbf{r}|^{2H})$ is the difference between the mass density at $\mathbf{r}$ and at the centre of the region. Since an fBm is a Gaussian process, the integral over normally distributed variables is also normal. Thus it suffices to calculate the variance to uniquely determine the distribution. Note that the integral is not trivial, because the stochastic variables $X(\mathbf{r})$ are not independent.

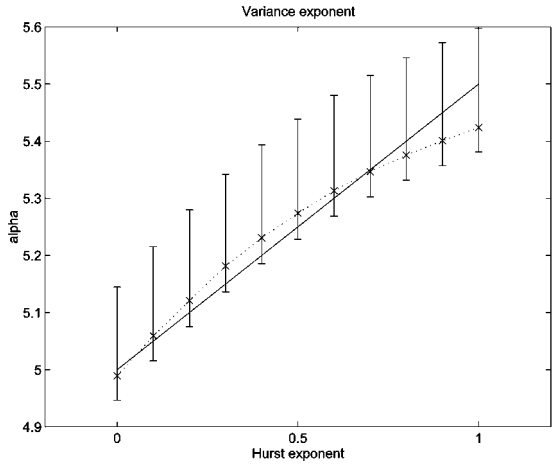


Figure 6-1. *The variance exponent as a function of the Hurst exponent. The dotted line represents linear interpolation between the determined values. The unbroken line is 5 + H/2, consistent with data due to the large error bars. However, I never made use of this fact.*

The above integral is in principal possible to evaluate, it may even be an easy one if you're confident enough with the theory of stochastic processes (with multiple time dimensions). I am not, however, so I went to the computer and used the developed fBm package[3] to calculate the variance. I did it by convolving the fBm with a sphere and determinening the variance as a function of the radius of the sphere. It turned out that the variance is a power law with an exponent α that depends on the Hurst exponent as seen in Figure 6-1. The average deviation from the power law was less than 2% in logarithmic scale. By displacing all numbers 2% in such a way that the gradient (that is, the power law exponent) was maximised and minimised respectively, I obtained the extreme error bars in Figure 6-1.

Summarising, we have

$$M^{(L)} \in N\left(\frac{4\pi}{3} L^3 \rho_{cloud}, c' L^{\alpha}\right),$$

for spherical regions, where *c'* is a constant. Now we only need to evaluate the integral. Since the integrals look a little bit scary I have to confine myself to numerical analysis.

I have not fully investigated the dependence of the integral on parameters. However, for those cases I did make calculations, the derived IMF looked intriguingly similar to the observed IMF. The slope the power law fit for larger masses[4] seemed to be surprisingly stable with respect to the parameter variations I did, and seems to be perfectly consistent with observations. At the present time, however, I cannot say whether this is a coincidence or not. I will make further investigations, both numerical and analytical, but am not able to include any results in this report due to its time schedule. As a foretaste an example of an IMF derivation can be seen in Figure 6-2 (page 30). I have so far only examined the low star forming efficiency case.

[3] *The fBm package is described in detail in Appendix A.*
[4] *That the IMF showed a power law behaviour for large masses was first found by Salpeter (1955).*

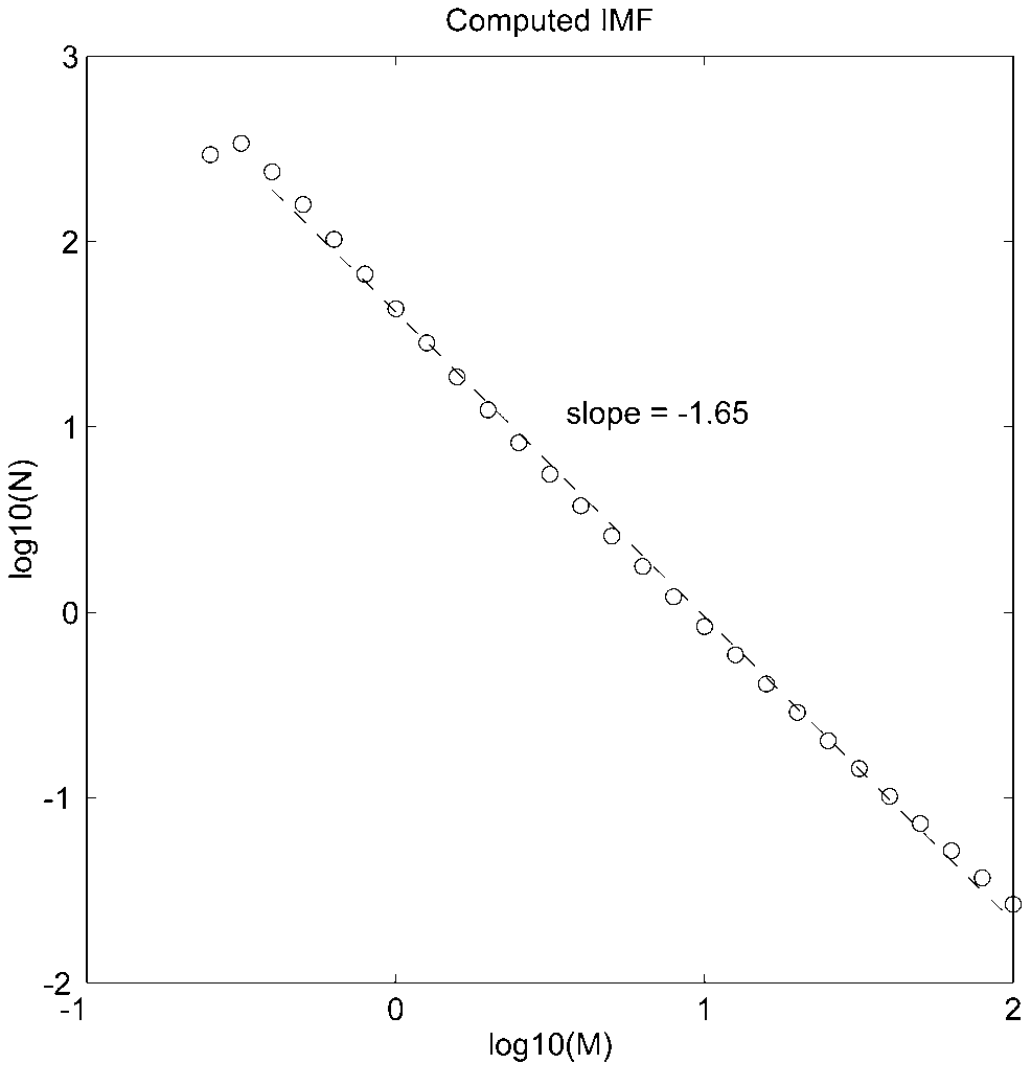


Figure 6-2. *An example of a stellar initial mass function as calculated with the model outlined in this chapter. A straight line has been fitted with the slope -1.65. The result in this graph seems to be rather stable with respect to parameter changes. The scale has been set to resemble the observed IMF; only the slope is invariant under scaling. The cusp is caused by the critical mass cut-off and is also a consequence of the scale. Compare with the observed IMF in Figure 1-1.*

7. Results from fBm simulations

Before making any serious computer simulations, I tested the developed software thoroughly. Because of the complexity raised by *n*-dimensionality, I only tested the routines for up to three dimensions (where I checked lots of things by hand), but I presume that they will work as expected for more dimensions as well.

7.1 Using `fBm` as an fBm generator

I made several different images with `fBm` with different Hurst exponents, just too see how the images appeared to the eye. Here I show two examples, one smooth of high Hurst exponent (H = 1) and one Brownian with intermediate Hurst exponent (H = 0.5). Look at the colour plates for fBm images in colour. I also made three-dimensional fBm and looked at slices of them. Combining slices along one axis made a nice movie, showing a cloud reshaping. Because the Fourier generated fBm are cyclic (periodic) by nature, so was the movie.

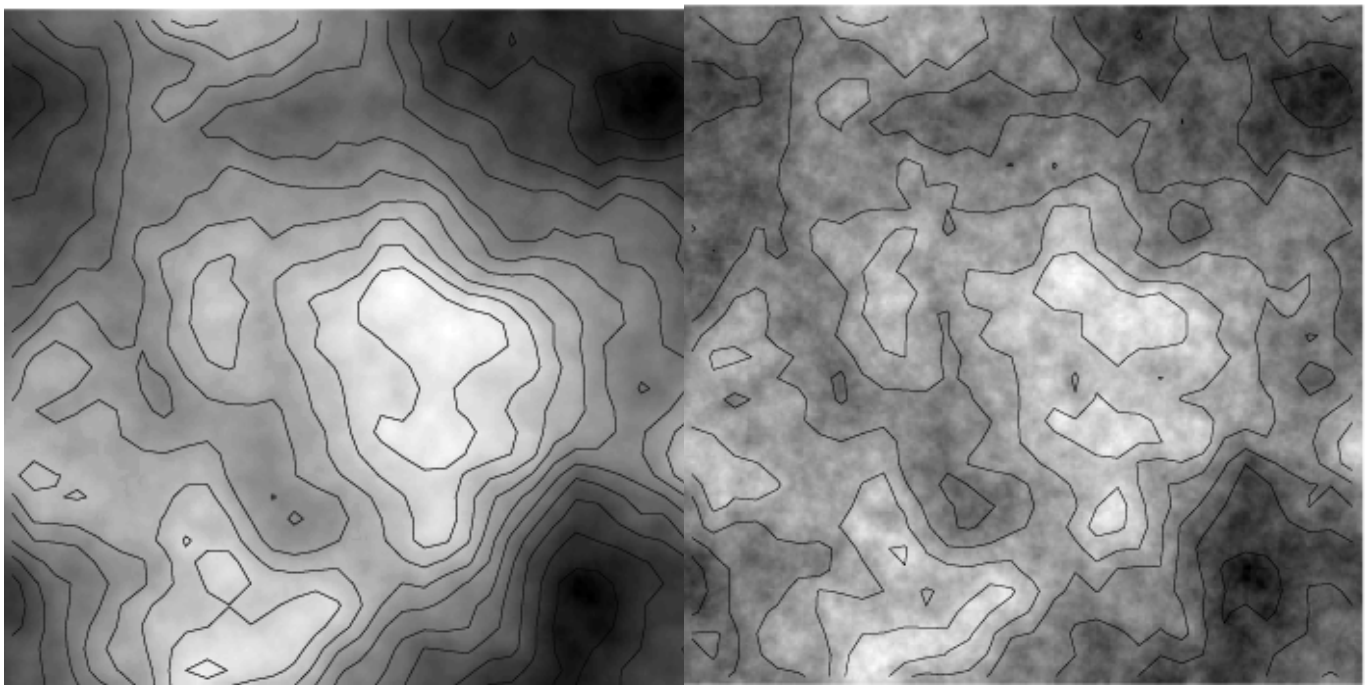


Figure 7-1. *Two examples of fBm images. On the left H = 1, on the right H = 0.5. Level plots are laid on to elucidate the structure. Note that the phases are the same in both images, it is only the slope of the amplitudes that is changed. Also note that the images are periodic; this is an artefact due to the discrete Fourier transform.*

7.2 Box counting the fBm

To determine how well the box counting algorithm worked for fBm, I tried it on several two-dimensional fBm images with different resolutions and different Hurst exponents.

I first tried to determine how sensitive `boxcount` is to displacements of the origin. I made up to 30 displacements for different box sizes, resolutions and Hurst exponents. It turned out that the variance of the determined number of boxes for one specific image is very low and that

5 displacements is far sufficient to make the uncertainty negligible even for the lowest resolutions.

The program I made to control the `boxcount` routine let it estimate the number of boxes needed to cover the fBm for up to 16 different box sizes (depending on the resolution; the higher resolution the larger the range to vary the box sizes over) equally distributed with sides ranging from half the cloud down to two times the resolution limit. Putting the data together this yielded a table with the number of covering boxes as a function of their box sizes, where I used the median as the number of covering boxes. I used Matlab to analyse the data, finding that as expected a straight line could with very good approximation be drawn through the points in a logarithmic diagram. Estimating the equation of line with the least square method I interpreted the slope of the line as the box counting dimension. The error in this estimate was small, about 0.1 dimensions.

To check the stability with respect to different fBm of the same theoretical dimension, I determined the dimension for up to 10 different fBm. I found that the standard deviation is about 0.1 dimensions, but that it has a slight dependence on the Hurst exponent such that the smaller the Hurst exponent (and the larger the fractal dimension), the larger the standard deviation.

Taking the mean over the different determined dimensions determined for fBm created with the same Hurst exponent, I repeated this for ten different Hurst exponents uniformly distributed over the unit interval.

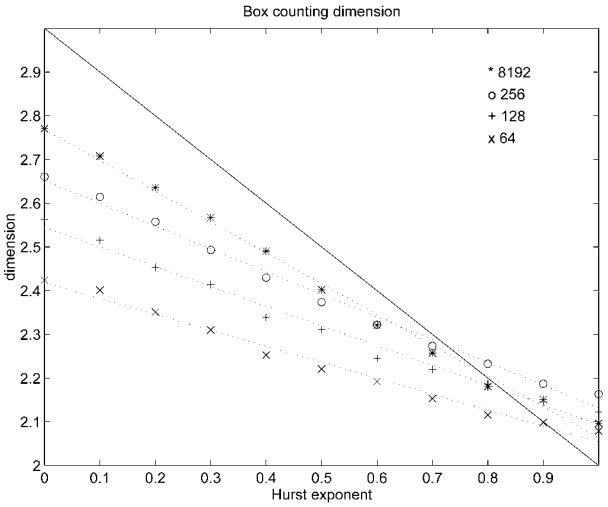


Figure 7-2. *The figure shows the determined box dimension for four different resolutions along with the theoretical dimension as an unbroken line. The dotted lines represent the least square fits.*

To my disappointment the fractal dimension provided by `boxcount` proved not to be a reliable estimate of the true fractal dimension, as determined from theory. I estimated the box counting dimension of pictures which ranged from 64×64 pixels up to 8192×8192 pixels and it turned out that the deviations from theory shrunk a bit with the resolution. This is in support of the conclusion of Bensch et al. (1998), who investigated the box counting algorithm on 64×64 grids, that the discrepancy between theory and the estimated values are due to the discrete samplings. If so, the convergence towards the theoretical values is a very slow function of resolution however (see Figure 7-2). The small errors in the estimating process of the box counting dimensions show that the discrepancies with the theoretical values are systematic.

7.3 Using the `deltavar` algorithm

Even though, in principle, the box counting algorithm could be calibrated to compensate for the deviation, I decided not to use it. Instead, I turned to the method of Δ-variance developed by Stutszki et al. (1998).

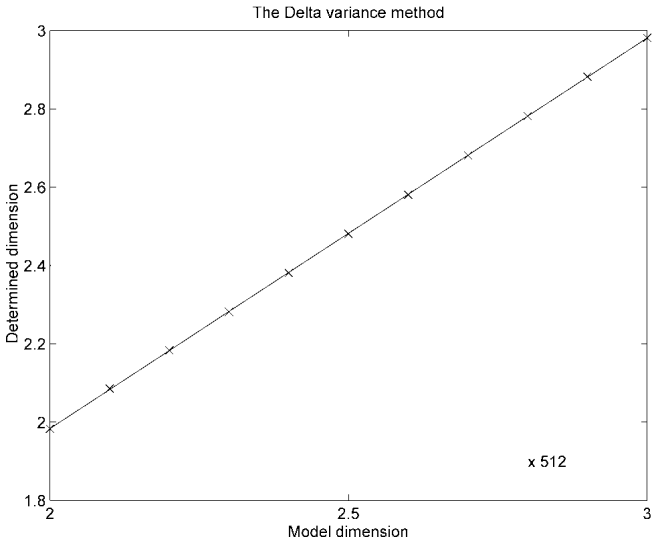


Figure 7-4. *This diagram shows the fractal dimension, as determined by* `deltavar`*, as a function of the theoretical fractal dimension of an fBm image of size 512 × 512.*

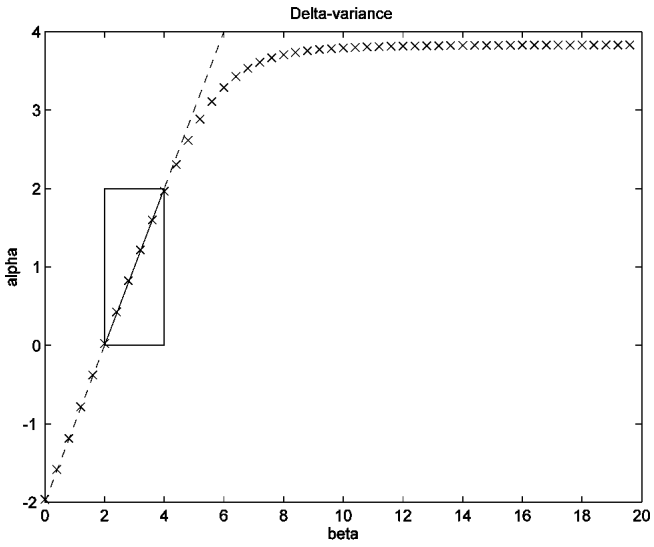


Figure 7-3. *This diagram shows the exponent α, determined by* `deltavar`*, as a function of the theoretical β of the fBm. The box encloses the region where the fBm is fractal along with the theoretical unbroken line. The dotted line represents the theoretical part where α = β - 2 (see section 3.2).*

Again, I wanted to check the reliability of the resulting algorithm, `deltavar`. The procedure runs much in the same way as `boxcount`, except that no origin displacements were needed and that the test space had to be drastically diminished because of the high time complexity of the algorithm ($O(n^2)$). Also, because I found that (as predicted from theory) the Δ-variance does not depend on the phases but only on the amplitude spectrum of the fBm, only one fBm per Hurst exponent was needed.

For each fBm image, the controlling program let `deltavar` determine the Δ-variance for up to 16 different scale sizes of the ”*n*-sphere”. As in the case of `boxcount`, the data was analysed in Matlab. Again the data lined up very nicely along a straight line in a logarithmic diagram. The slope, determined from least squares fitting, was identified with the parameter α from which the fractal dimension can be calculated as $D = 3 - \alpha / 2$. Doing this for ten different Hurst exponents distributed uniformly on the unit interval, I draw the diagram of Figure 7-4. As one can see, this time the estimated dimensions corresponded very closely to the theoretically expected ones.

I did not try images larger than 512 × 512 pixels, because already they took on the order of 10 hours computing time each. And because the reliability of `deltavar` seems satisfactory even for 64 × 64 pixels images, I see no reason why it shouldn’t get even better at higher resolution.

7.4 The `projection` of fBm

What I really wanted to do was to analyse the projection of three-dimensional fBm onto a discrete grid. Because three-dimensional arrays require a lot of space, I decided to generate the three-dimensional clouds and save only the projections (accomplished by `project`) in order to free computer resources. I worked with $256 \times 256 \times 256$ grids, which required 64 MB of free RAM, and also with smaller $128 \times 128 \times 128$ grids just to compare the results. The resulting projections where analysed using the `deltavar` routine as described in section 7.3. The diagram in Figure 7-5 was the result. The unbroken line is the dimension of the projection as a function of the dimension of the three-dimensional fBm as expected from proposition 2 in chapter 4. As one can see, the determined values closely follow this line, however slightly lower. These results suggest that the theory works just as fine on a finite grid as in the continuous case.

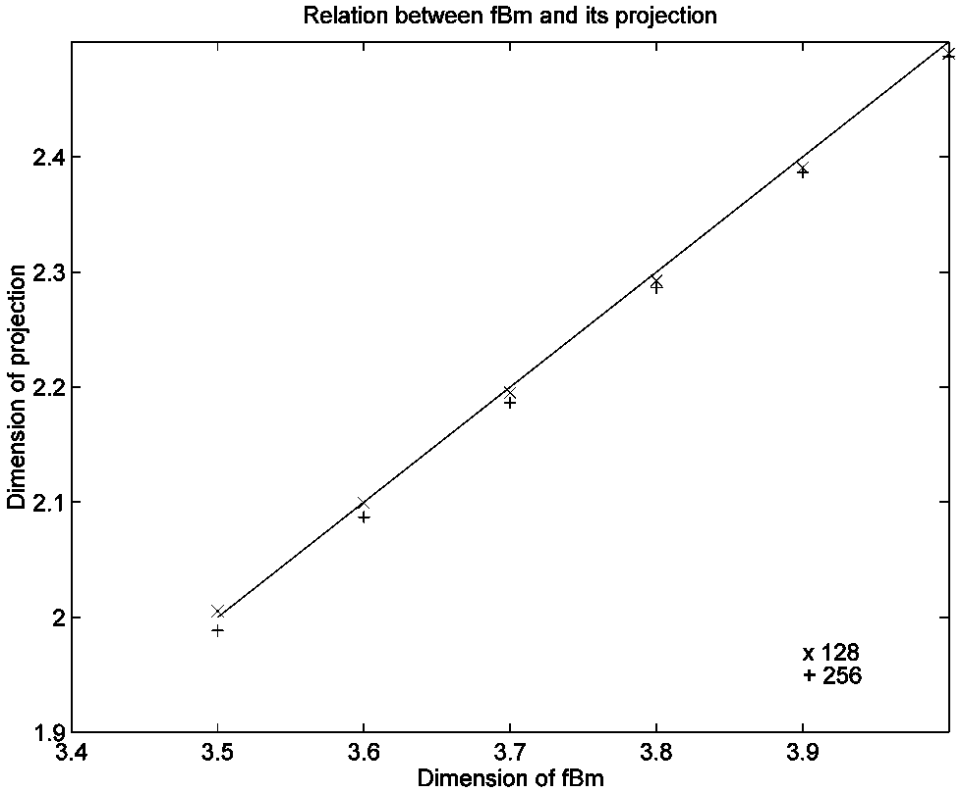


Figure 7-5. *This is the fractal dimension of the projected fBm, as determined by* `deltavar`*, as a function of the theoretical dimension of the three-dimensional cloud. The unbroken line represents the theoretical relation as derived in section 4.1.*

8. General conclusions

8.1 The projection of fractal models

In the case of *fractional Brownian motion*, the problem of determining the intrinsic fractal dimension as a function of the projected fractal dimension, has been solved. For more general fractal structures, the problem is still open. The derived relation $d_2 = d_3 - 1.5$ instead of the generally assumed $d_2 = d_3 - 1$ affects the properties of three-dimensional fractal cloud models strongly when compared to observed, projected, real clouds. The three-dimensional fractal dimension as derived from observations will be 0.5 larger with this new relation.

In the case of real clouds with appreciable optical thickness, care must be taken when determining the fractal dimension. The projected fractal dimension will satisfy the same relation as for optical thin clouds as long as the intensity is a monotone function of the column density, that is, as long as the emission, along the line of sight, is not heavily saturated.

8.2 Fractal models

The fractal model *fractional Brownian motion* is so far consistent with observations. It has shortcomings; one of which I consider to be serious is that it is completely isotropic. There is no room for the filaments often observed in real clouds. A fractal model may still be valid, if one can show that these filaments can be produced by means of stretching and bending an fBm, for example. The model would then still have a well defined fractal dimension. A more systematic and thorough study is needed to see where to otherwise improve the models.

The multifractal models look promising, but better means of generating multifractals remain to be seen.

The discussed models have been static; dynamic models would be an interesting extension. This should require a more thorough understanding of the physics in the cloud, but, maybe, a simple geometrical model can account for the observed velocity distribution.

8.3 Theoretical modeling of the IMF

Using fractional Brownian motion as a model of the mass distribution of a cloud and using some assumptions, I have been able to derive an analytical model of the IMF. Although the model is analytic, I have not (yet) been able to solve the equations analytically, but rather I've tried numerical methods. The results so far look promising, but further investigations are needed in order to determine how the equations behave under various parameter changes.

8.4 Computer simulations

The developed `fBm` package has been used to check some theory in practice. The box counting algorithm has been found to be an unreliable method to determine the fractal dimension. The recently developed Δ-variance method on the other hand, works flawlessly on fBm. The theoretical relation $d_2 = d_3 - 1.5$ holds also in practice on finite data grids.

References

Adams, *A topological approach to the study of astrophysical maps*, ApJ 1992, **387**:572

Adams & Wiseman, *Formal results regarding metric-space techniques for the study of astrophysical maps*, ApJ 1994, **435**:693

Avnir, Biham, Lidar & Malcai, *Is the geometry of nature fractal?*, Science 1998, **279**:39

Barnsley, *Fractals everywhere*, 1993, 1988 Academic Press Inc.

Bazell & Désert, *Fractal structure of interstellar cirrus*, ApJ 1988, **333**:353

Beech, *The projection of fractal objects*, 1992, Ap&SS, **192**:103

Bensch, Ossenkopf & Stutzki, *Structure analysis of molecular clouds: Methods*, poster presented at the 2nd Guillermo Haro Conference *Interstellar Turbulence*, January 1998

Benson & Myers, *A survey for dense cores in dark clouds*, ApJ Suppl. 1989, **71**:89

Elmegreen, *The initial stellar mass function from random sampling in a turbulent fractal cloud*, ApJ 1997, **486**:944

Falconer, *Fractal geometry*, 1990 Wiley

Falgarone, Philips & Walker, *The edges of molecular clouds: Fractal boundaries and density structure*, ApJ 1991, **378**:186

Frigo & Johnson, *FFTW 1.3 User's manual*, MIT 1998, http://theory.lcs.mit.edu/~fftw/

Gonzales, *Digital Image Processing*, 1987 Addison-Wesley

Henriksen, *On molecular cloud scaling laws and star formation*, ApJ 1991, **377**:500

Henriksen, *Star formation in giant molecular clouds*, ApJ 1986, **310**:189

Hetem & Lépine, *Fractal 3-D simulations of molecular clouds*, 1993, A&A, **270**:451

Kaas, Östlin & Kristen, *A geometrical approach to the initial mass function - I. The space of molecular cloud fragmentation*, 1998

Larson, *Towards understanding of the initial mass function*, 1992, MNRAS, **256**:641

Mandelbrot, *The fractal geometry of nature*, 1982 Freeman

McCauley, *Introduction to multifractals in dynamical systems theory and fully developed fluid turbulence*, Physics Reports 1990, **189**:*5*:225

Ossenkopf, Bensch & Zielinsky, *Structure analysis of molecular clouds: Observations and simulations*, poster presented at the 2nd Guillermo Haro Conference *Interstellar Turbulence*, January 1998

Peitgen, Jurgens, Saupe, *Chaos and fractals - new frontiers of science*, 1992 Springer-Verlag

Peitgen, Saupe, *The Science of fractal images*, 1988 Springer-Verlag

Rana, *Mass function of stars in the solar neighbourhood*, A&A 1987, **184**:104

Richtler, *On a possible origin of the initial mass function of stars*, A&A 1994, **287**:517

Råde, Westergren, β - *mathematics handbook*, 1990 Studentlitteratur, 1990 Chartwell-Bratt Ltd

Salpeter, *The luminosity function and stellar evolution*, ApJ 1955, **121**:161

Scalo, *The stellar initial mass function*, Fund. Cosmic. Phys. 1986, **11**:1

Stutzki, Bensch, Heithausen, Ossenkopf & Zielinsky, *On the fractal structure of molecular clouds*, A&A 1998, in press

Vogelaar & Wakker, *Measuring the fractal structure of interstellar clouds*, A&A 1994, **291**:557

Voss, Random fractals: *Self-affinity in noise, music, mountains, and clouds*, Physica D 1989, **38**:362

Wiseman & Adams, *A quantitative analysis of IRAS maps of molecular clouds*, ApJ 1994, **435**:708

Wood, Myers & Daugherty, *IRAS images of nearby dark clouds*, ApJ Suppl 1994, **95**:457

Further reading

Bate & Burkert: *Resolution requirements for smoothed particle hydrodynamics calculations with self gravity*, MNRAS 1997, **288**:1060

Chistodoulou & Tohline, *Phase transitions time scales for cooling and isothermal media* , ApJ 1990, **363**:197

Constantin & Procaccia, *Scaling in fluid turbulence: A geometric theory* , Physical Review E 1993, **47**:5:3307

Cox, *Overall models of the interstellar medium* , proceedings of the 11th IAP Astrophysical meeting 1995.

Devaney, *Chaotic Dynamical Systems*, 1989 Addison-Wesley

Elmegreen & Efremov, *A universal mechanism for open and globular clusters in turbulent gas*, ApJ 1997, **480**: 235

Elmegreen & Falgarone, *A fractal origin for the mass spectrum of interstellar clouds* , ApJ 1996, **471**: 816

Falgarone, *Small scale structure of pre-star-forming molecular clouds*, proceedings of the 11th IAP Astrophysical meeting 1995.

Gut, *An intermediate course in probability* , 1995 Springer-Verlag

Heithausen, Bensch, Stutzki, Falgarone & Panis, *The IRAM key project: Small-scale structure of pre-star forming regions.*, (Letter A&A 1998)

Hentschel & Procaccia, *Relative diffusion in turbulent media: The fractal dimension of clouds*, Physical Review A 1984, **29**:3:1461

Houlahan & Scalo, *Recognition and characterisation of hierchial interstellar structure. II. Structure tree statistics*, ApJ 1992, **393**:172

Juvela, *Clumpy cloud models for CS and C34S spectra observed towards southern massive star forming cores*, A&A 1998, **329**:659

Kallenberg, *Random measures*, 1983 Academic Press Inc.

Kanjijal & Basu, *A study of the fragmentation of molecular clouds and the form of initial functions for low-mass protostellar fragments*, A&SpS 1992, **193**:17

Kramer, Stutzki, Röhrig & Corneliussen, *Clump mass spectra of molecular clouds*, A&A 1998, **329**:249

Labini, Montuori & Pietronero, *Angular projections of fractal sets* , issa preprint 1996

Larson, *Star formation and galactic evolution* , proceedings of the 11th IAP Astrophysical meeting 1995.

Larson, *Star formation in groups*, MNRAS 1995, **272**:213

Maris & Kadanoff, *Teaching the renormalisation group*, Am.J.Phys. 1978, **46**(6):652

McKee, *Global structure of the multiphase interstellar medium*, proceedings of the 11th IAP Astrophysical meeting 1995.

Östlin, Kristen & Kaas, *A geometrical approach to the initial mass function - II. Subfragmentation - from cloud clumps to stars*, 1998

Richardson, *A self-consistent numerical treatment of fractal aggregate dynamics* , Academic Press 1995, **5**:320

Schroeder, *Fractals, Chaos, Power laws*, 1991 Freeman

Stauffer, Aharony, *Introduction to percolation theory* , 1994 Taylor & Francis

Vega, Sánchez & Combes, *Fractal dimensions and scaling laws in the interstellar medium: A new field theory approach*, (Phys Rev D Nov 15, 1996)

Vega, Sánches & Combes, *Self-gravity as an explanation of the fractal structure of the interstellar medium*, (Letter to nature, September 5, 1996)

Appendix A - An fBm implementation

To test how the theory of fractal clouds works in practice on finite discrete grids, I decided to implement a simulation package. Because of their close resemblance to real interstellar clouds, I chose to implement and analyse mainly the fractional Brownian motion (fBm). I also decided to use the C language since it is the computer language I am most familiar with.

The package, which I call `fBm`, consists of several parts. I will discuss the various parts in detail below. The header file as well as the source code can be found in Appendix B.

A note on the implementation: I figured that it would be desirable to be able to produce fBm for at least one, two, three and maybe four dimensions, so I decided not to fix the dimension in the implementation but allow *n* dimensions. This made the code a bit more complex however, with some to me unusual problems as how to implement *n* entangled for-loops when *n* is a variable, for example. I solved these problems but they made the code less penetrable, although I've tried to make it as clean as possible.

The `fBm` routine

`fBm` is the main routine of the package. Its function is straightforward: generate an fBm embedded in *n* spatial dimensions using the Hurst exponent *H*. One has to submit information about the grid size as well, for instance we may wish to use a 31×415×92 grid for our three-dimensional cloud.

`fBm` uses a method called Fourier synthesis to generate the fBm, a method that makes use of the discrete fourier transform. One generates the function in Fourier space, making the coefficients having a power law distribution depending on *H*, and then use the Fourier inverse transform to transform it back to "time" space.

Due to the nature of discrete Fourier transform (DFT), the resulting fBm will have two special properties[1]:

i) It will be periodic, that is, there will be no unique boundaries.
ii) The product of the grid sizes will be even.

In case of the `fBm` implementation, the second property is strengthened to require that the last grid size will be even.

Because, in general, the inverse Fourier transform is complex and we want our fBm cloud to be real, we have to put some constraints on the coefficients generated by `fBm`. These can be stated in a single equation,

$$a_{i_1,i_2,\ldots,i_n} = -\overline{a}_{d_1-i_1,d_2-i_2,\ldots,d_n-i_n},$$

where $a_{i_1,i_2,\ldots,i_n}$ is the (complex) element in Fourier space on the co-ordinates (i_1, i_2, …, i_n), $\overline{a}_{i_1,i_2,\ldots,i_n}$ its complex conjugate and d_1, d_2, …, d_n are the sizes of the grid on the corresponding

[1] *See e.g. Gonzales (1987)*

axis. To produce the correct power law distribution, the amplitudes of the coefficients should be like

$$\left|a_{i_1,i_2,\ldots,i_n}\right| \propto \frac{1}{\left(j_1^2 + j_2^2 + \ldots + j_n^2\right)^{\frac{2H+n}{4}}},$$

where $j_k = \min(i_k,\ d_k\ -\ i_k)$. I assume that $i_k \in \{0,2,..,d_k - 1\}$ periodically, that is, j and j (modulo d_k) are identified, and in the case all i_k are zero, the amplitude is set to zero as well, in order to avoid an infinite amplitude.

In `fBm` the amplitudes are set as above and the phases are completely random, distributed uniformly on the interval 0 to 2π radians.

Since I didn't want my master thesis to be solely devoted to the implementation of DFT, I made a search on the internet and found `FFTW`[2] ("the Fastest Fourier Transform in the West"), a fast Fourier transform implementation by Matteo Frigo and Steven G. Johnson at Massachusetts Institute of Technology (MIT). The `FFTW` package is GNU freeware and worked without any major problems. The only problem I encountered was that I wanted to use the real `FFTW` package (`RFFTW`), and its data format is somewhat intricate and not well documented since `RFFTW` is not yet an official part of `FFTW`.

The reason I wanted to use `RFFTW` is computer memory space. Because of the symmetries mentioned above that have to be fulfilled for the Fourier transform of a real function, we need not save all coefficients in the Fourier space to uniquely determine the "time" space. The `RFTTW` makes use of this fact and saves 50% computer memory space.

Another thing I made to save space was to use `float` variables (using four bytes) instead of `double` (using eight bytes) for the real numbers and saved another 50% that way. I found that the extra precision provided by `double` is not necessary.

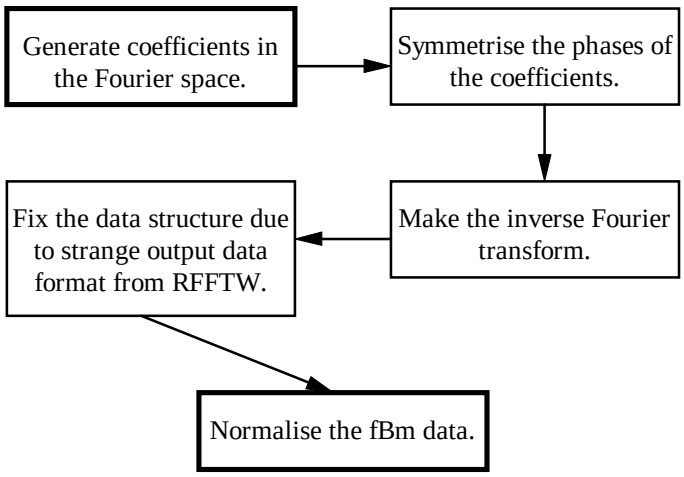


Figure 8-1. *A scheme over the* `fBm` *routine.*

That memory is a crucial factor is easy to appreciate if one considers that even grids with 512 elements on each axis will in three dimensions require $512 \times 512 \times 512 \times 4 = 512$ MB of memory.[3]

Having Fourier transformed the output the last thing `fBm` does is to normalise the fBm data to the interval [0, 1]. This is safe to do, because according to the lemma of chapter 4, this will not change any essential properties of the fBm.

The `fBm` routine is quite fast with the time complexity $O(n \log n)$, where n is the number of data. Although it is FFTW that stands for the $O(n \log n)$ complexity, the rest being linear, in practice, it is the generation of the coefficients that is the main bottleneck. Not a large such though and it shouldn't be (and isn't for me) a problem to generate as large fBm as one wishes.

The `deltavar` routine

The `deltavar` routine is used to analyse an n-dimensional data grid with the method of Δ-variance. I will only review the method here, it is thoroughly investigated by Stutzki et al. (1998). The method is based on the following observation. If we have an fBm data grid and convolve it with a special function depending on a variable L, then the variance of the data grid under some circumstances will be a power law of this variable. The exponent α of the power law determines the power spectrum power law exponent, β, and thereby the Hurst exponent, H, and the fractal dimension, D, of the fBm by the relations

$$\begin{aligned} \beta &= \alpha + 2 \\ H &= \frac{\beta + n}{2} \\ D &= n + 1 - H \end{aligned} \quad ,$$

where n is the dimension of the embedding Euclidean (spatial) space. The special function depending on L looks like

$$\Phi_L(r) = c \cdot L^{-n} \cdot \begin{cases} 1 & \text{if} \quad |r| \le \frac{L}{2} \\ \frac{-1}{3^n - 1} & \text{if} \quad \frac{L}{2} < |r| \le \frac{3L}{2} \\ 0 & \text{if} \quad \frac{3L}{2} < |r| \end{cases},$$

where r is the co-ordinate in spatial space and c is some constant that isn't important to us since we are only interested in how the Δ-variance changes with L, the power law dependence. Since this special function is spherical, I call it, somewhat improperly, "n-sphere" in the comments attached to the code.

Taking only the parameter L and the data structure as input, the `deltavar` procedure runs like this: Start by generating the special function. Since we are working with discrete data, care must be taken. The function is designed so that the integral over it will equal zero in the

---

[2] *See Frigo, Johnson (1998)*

[3] *Note that 1 MB corresponds to 1048576 Bytes = 2^{20} Bytes and not 10^6 Bytes as one perhaps would expect.*

continuous case, but since we are discrete, this need not be so. Therefore, `deltavar` adjusts the outer part of the *n*-sphere to be such that the integral over that part is equal to minus the integral of the inner part. Ultimately, a list with all non zero elements of the special function is generated.

The next step is to convolve the function with the given data. Since we do not want to generate another huge data set with the convolved data, nor do we want to destroy the original data, `deltavar` calculates the variance at the same time as it is convolving. This is done by using the formula

$$\text{variance} = \frac{1}{N-1}\left(\sum_i c_i^2 - \frac{\left(\sum_i c_i \right)^2}{N} \right),$$

where c_i is the value of the data set at i (it is assumed that the data are ordered) and N is the total number of elements. This formula means that we do not need to keep all data points c_i in memory at the same time to calculate the variance. It is sufficient to keep only the sum and the quadratic sum of the data points. In our case, since the sum over all data points is zero, we only need to save the quadratic sum. Note that the data are assumed to be periodic to avoid boundary effects.

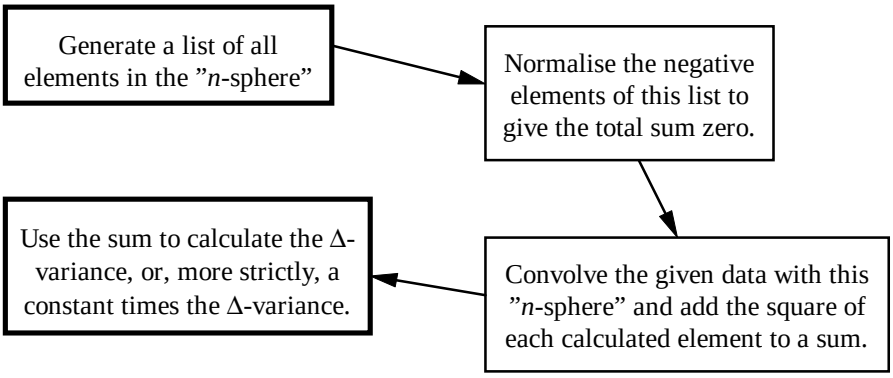


Figure 8-2. *A scheme over the* `deltavar` *routine.*

`The deltavar` routine has a very high time complexity, $O(n^2)$, and is therefore not well suited for larger data arrays. To obtain an accurate value of α for a 512×512 image for instance, takes on the order of 10 hours on a Sun Ultra Sparc. Of course one can still content oneself with small ”*n*-spheres” or to lower the resolution to get manageable computing times.

The `boxcount` routine

The `boxcount` routine is used to analyse an *n*-dimensional data grid with the box counting algorithm. The algorithm is quite straightforward. Its goal is to calculate how many (n + 1)-boxes are needed to cover the surface (embedded in n + 1 dimensions) defined by the data grid, when the boxes are of length (*grid size*) / *intervals* in each direction.

Since computer memory space is of high priority, we cannot just implement the trivial algorithm to count the boxes, that is, generate all possible boxes and see how many have a part

of the set contained in them. Instead `boxcount` is using a so called traversing box, one box that goes through every box position and counts how many times it has a part of the set contained in it. `boxcount` doesn't use this algorithm exactly either, but takes advantage of the fact that the set is a surface in the $n + 1$ embedding dimensions. If we exclude the intensity dimension, and partition only the spatial axes, each part will correspond to at least one box. More precisely, one will need precisely as many boxes as there is room for between the maximum and the minimum intensities of that spatial part. This fact is exploited in `boxcount` so that only the parts of the spatial partition are visited, and the boxes corresponding to that part are calculated as above.

In more detail, `boxcount` works like this: Start by choosing an origin at random. This, because we want to avoid any systematic errors that can arise if the generated fractal we are to measure has the same origin as the partition into boxes. Partition the spatial space and visit each part once determining the maximum and the minimum of the intensity co-ordinates for all points of the fractal set belonging to that part, and adding the difference to a variable, sum all these differences for all parts. From this sum the number of boxes needed to cover the set is determined. The procedure is repeated as many times as desired with new displacements of the origin each time.

Since this algorithm visits each point in the set exactly one time for each displacement of the origin, its complexity is $O(n)$. This means that it is only linearly dependent on the number of elements: double the number of elements in the set and the algorithm will only double its computation time. This fact makes `boxcount` more suitable than `deltavar` when it comes to analyse huge data sets.

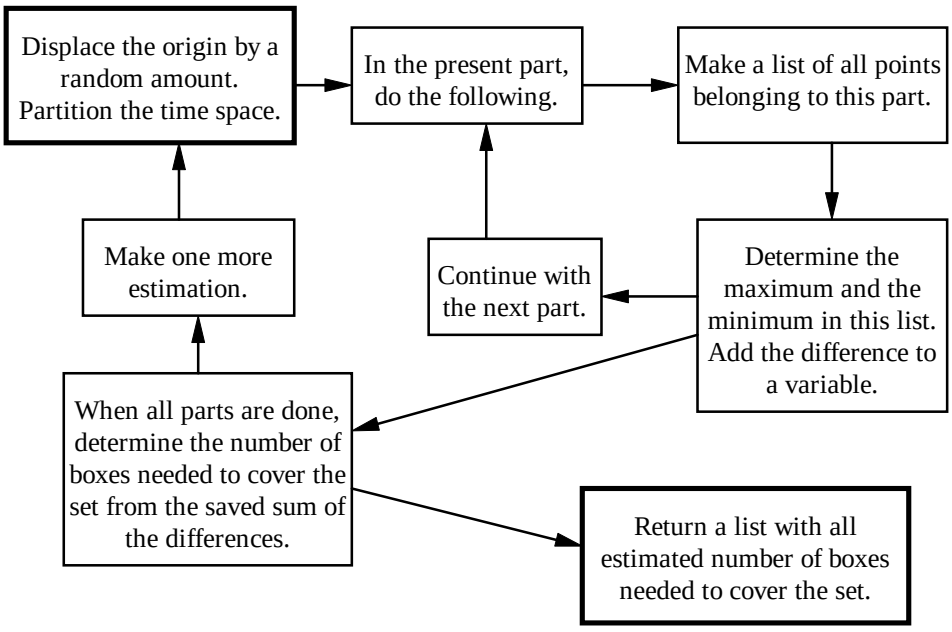


Figure 8-3. *A scheme over the* `boxcount` *routine.*

The `project` routine

The `project` routine is used to study the effects of projection of fractal sets. It is very simple: Just integrate along a given axis. If the input data have n dimensions, the output will have n - 1 dimensions. The output from `project` is normalised to the interval [0, 1].

The `ext_ind` and `ext_coo` routines

Because the extraction of data from the data array can sometimes be confusing doing it directly, I have included the two routines `ext_ind` and `ext_coo` in the package. Given the co-ordinates of an element, `ext_ind` extracts the index, i.e. the address to the element in the data array. Similarly, `ext_coo` extracts the co-ordinates of a given index. Both routines use the fact that the data elements are stored at the address

$$index = (c_1 + d_1 \times (c_2 + d_2 \times (\ldots (c_{n-1} + d_{n-1} \times c_n) \ldots))),$$

where c_i is the i:th co-ordinate and d_i is the number of elements in the$i$:th direction.

The `gauss` and `gauss_init` routines

In the C language, there is no standard pseudo random number generator with a normal distribution. `gauss` uses the built in pseudo random number generator with uniformly distribution and the method of Box-Müller[4] to transform the uniformly distributed numbers into normally distributed ones.

`gauss_init` merely sets the seed of the pseudo random number generator.

[4] *See Råde, Westergren (1990)*

Appendix B - Source code

The header file `fBm.h`

```
/* fBm - a package that creates D-dimensional fractal Brownian motion
 * clouds and calculates their fractal dimensions.
 *
 * Original code: Alexis Brandeker 19980320
 * Last changed:  Alexis Brandeker 19980423
 *
 * Email: alexis@astro.su.se
 *
 * Version: 1.1
 */

#ifndef FBM_H
#define FBM_H

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "rfftw.h"

#ifndef PI
#define PI 3.1415926536
#endif

/* Define the precision used by fBm */
#define FBM_PREC float

/* fBm_struct: A structure used to handle the fBm data.
 */
struct fBm_struct
{
   FBM_PREC          *data;      /* Where the actual data is stored    */
   unsigned int      D;          /* Number of axes in the data array   */
   unsigned int      *dim;       /* A pointer to array with dimensions *
                                  * of the different axes              */
};
typedef struct fBm_struct fBm_struct;


/* fBm: Generates an fBm structure using the FFTW package. Input is
 * the Hurst exponent and an fBm_struct with fBm_struct.data allocated
 * to n[1]*n[2]*n[3]*...*n[D-1]*(n[D]+2)*size(FBM_PREC) bytes, where
 * n[i] is the dimension of axis i and D is the number of the axis.
 * Caution: it is important to add the term 2 to the last axis
 * dimension (as above) because the extra bytes are used in the
 * process even if not apparent in the output.
 * The data in fBm_struct.data will be on row major form; see coo_ext
 * and ind_ext for the procedure.
 */

fBm_struct
*fBm(float           H,          /* The Hurst exponent                          */
     fBm_struct      fBm_data);  /* An uninitiated but allocated fBm structure */


/* deltavar: returns the delta variance of a given data structure and
 * size of the "n-sphere" used to convolve with.
 */

float
deltavar(float       L,          /* Radius of the "n-sphere"   */
         fBm_struct  fBm_data);  /* An initiated fBm structure */


/* boxcount: Given an initiated fBm structure, number of interval
 * partitions along an axis and a number disp_num defining the number of
 * diplacements, boxcount returns a list of number of boxes needed
 * to cover the fBm structure (when viewed as a fractal hypersurface).
 * The list contains disp_num entries, one for each random displacement
 * of origo.
 */

unsigned int
*boxcount(unsigned int intervals,      /* Number of intervals per axis        */
          unsigned int disp_num,       /* Number of displacement iterations   */
          fBm_struct   fBm_data);      /* An initiated fBm structure          */
```

```
/* project: Given an indata structure, an uninitiated outdata
 * structure and an axis, project projects the data along the axis
 * and writes the resulting projection to the outdata structure
 * which thus has one less dimension.
 */

int
project(fBm_struct   fBm_in,     /* The original fBm structure  */
        fBm_struct   fBm_out,    /* The projected fBm structure */
        unsigned int axis);      /* The axis to integrate along */


/* ext_ind: extract the index of the element on the co-ordinates
 * coord. ext_ind returns the index.
 */

inline unsigned int
ext_ind(unsigned int D,          /* Number of axes in the data array   */
        unsigned int *dim,       /* A pointer to array with dimensions *
                                  * of the different axes              */
        unsigned int *coord);    /* Co-ordinates of element            */


/* ext_coo: extract the co-ordinates of an element with known index.
 * A pointer to the co-ordinate array where the result is stored is
 * given as input as well. ext_coo returns a pointer to the co-ordinate
 * array.
 */

inline unsigned int
*ext_coo(unsigned int D,         /* Number of axes in the data array   */
         unsigned int *dim,      /* A pointer to array with dimensions *
                                  * of the different axes              */
         unsigned int *coord,    /* Co-ordinates of element            */
         unsigned int index);    /* Index of element                   */


/*
 * init_gauss: Initialize the pseudo-random number
 * generator gauss() with a positive integer as seed.
 */

void init_gauss(unsigned int seed);


/*
 * gauss: Returns a N(0,1) real number using a
 * pseudo-random number generator.
 */

inline float gauss(void);

#endif FBM_H
```

The source file `fBm.c`

```
/* fBm - a package that creates D-dimensional fractal Brownian motion
 * clouds and calculates their fractal dimensions.
 *
 * Original code: Alexis Brandeker 19980320
 * Last changed:  Alexis Brandeker 19980423
 *
 * Email: alexis@astro.su.se
 *
 * Version: 1.1
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "rfftw.h"
#include "fBm.h"

/* fBm: Generates an fBm structure using the FFTW package. Input is
 * the Hurst exponent and an fBm_struct with fBm_struct.data allocated
 * to n[1]*n[2]*n[3]*...*n[D-1]*(n[D]+2)*size(FBM_PREC) bytes, where
 * n[i] is the dimension of axis i and D is the number of the axes.
 * Caution: it is important to add the term 2 to the last axis
 * dimension (as above) because the extra bytes are used in the
 * process even if not apparent in the output.
 * The data in fBm_struct.data will be on row major form; see coo_ext
 * and ind_ext for the procedure.
 */

fBm_struct
*fBm(float     H,                      /* The Hurst exponent                          */
     fBm_struct   fBm_data)            /* An uninitiated but allocated fBm structure */
{
   unsigned int      qs;               /* Quadratic sum                               */
   unsigned int      *dim_comp;        /* Dimension index for the complex array */
   unsigned int      *dim_even;        /* Half dimension, if even                     */
   unsigned int      *coord;           /* Co-ordinates in the complex array     */
   char              *bin_index;       /* Binary expansion index                      */
   unsigned int      prod;             /* Product of dimensions                       */
   FBM_PREC          fmin, fmax;       /* Normalising factors                         */
   FBM_PREC          rad, phase;       /* Arguments to a complex number         */
   FFTW_COMPLEX      *in_data;         /* Complex pointer; In this case it      *
                                        * points to fBm_data.data                     */
   rfftwnd_plan      plan;             /* The plan used by RFFTW                      */
   unsigned int      i, j, k, temp;    /* Just dummies                                */

   in_data = (FFTW_COMPLEX *) fBm_data.data;

   if(fBm_data.D == 0 || (fBm_data.dim[fBm_data.D - 1] % 2 != 0))
   {
   /* We must have a positive Euclidean dimension, and
    * the last dimension must be even. If not, exit.*/
      return (fBm_struct *) NULL;
   }

   if(((dim_comp = (unsigned int *)
         malloc( fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((dim_even = (unsigned int *)
         malloc( fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((coord = (unsigned int *)
         malloc( fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((bin_index = (char *)
         malloc( fBm_data.D * sizeof(char))) == NULL))
      return (fBm_struct *) NULL;

   for(i = 0; i < fBm_data.D; i++)
   {
      dim_comp[i] = fBm_data.dim[i];
   }
   dim_comp[fBm_data.D - 1] /= 2;      /* The last dimension is halved
                                       .* in the complex case */
   dim_comp[fBm_data.D - 1] += 1;      /* ...and 1 is added. See RFFTW
                                       .* for details. */

   for(i=0; i < fBm_data.D - 1; i++)
   {
      if(dim_comp[i] % 2 == 0)
         dim_even[i] = dim_comp[i] / 2;
      else
         dim_even[i] = 0;
   }
   dim_even[fBm_data.D - 1] = dim_comp[fBm_data.D - 1] - 1;
```

```
for(i = 0, prod = 1; i < fBm_data.D; i++)
   prod *= dim_comp[i];

for(i = 0; i < prod; i++)
{
   for(j = 0, k = i, qs = 0; j < fBm_data.D; j++)
   {
      temp = k % dim_comp[fBm_data.D - 1 - j];
      k -= temp;
      k /= dim_comp[fBm_data.D - 1 - j];
      if(fBm_data.dim[fBm_data.D - 1 - j] - temp <= temp)
      {
            qs += (fBm_data.dim[fBm_data.D - 1 - j] - temp) *
            (fBm_data.dim[fBm_data.D - 1 - j] - temp);
      }
      else
      {
            qs += temp * temp;
      }
   }

   if(qs != 0)
      rad = (FBM_PREC) pow( (double) qs,  -0.25 * (2 * H + fBm_data.D));
   else
      rad = 0;
   phase = (FBM_PREC) 2.0 * PI * rand() / RAND_MAX;
   c_re(in_data[i]) = (FFTW_REAL) rad * cos((double) phase);
   c_im(in_data[i]) = (FFTW_REAL) rad * sin((double) phase);
}

/* Symmetrise the array according to certain rules */

for(i = 0; i < prod; i += dim_comp[fBm_data.D - 1])
{
   ext_coo(fBm_data.D, dim_comp, coord, i);

   for(k = 0; k < fBm_data.D; k++)
      coord[k] = (dim_comp[k] - coord[k]) % dim_comp[k];

   j = ext_ind(fBm_data.D, dim_comp, coord );

   c_re(in_data[j]) = c_re(in_data[i]);
   c_im(in_data[j]) = -1 * c_im(in_data[i]);

   ext_coo(fBm_data.D, dim_comp, coord,
   i + dim_comp[fBm_data.D - 1] - 1);
   j += dim_comp[fBm_data.D - 1] - 1;

   c_re(in_data[j]) = c_re(in_data[i + dim_comp[fBm_data.D - 1] - 1]);
   c_im(in_data[j]) = -1 * c_im(in_data[i + dim_comp[fBm_data.D - 1] - 1]);
}

for(i = 1; i < (unsigned int) pow(2.0, (double) fBm_data.D); i++)
{
   for(j = 0, k = i; j < fBm_data.D; j++)
   {
      bin_index[fBm_data.D - 1 - j] = (char) k % 2;
      k -= (unsigned int) bin_index[fBm_data.D - 1 - j];
      k /= 2;
   }
   for(j = 0; j < fBm_data.D; j++)
   {
      coord[j] = (unsigned int) bin_index[j] * dim_even[j];
   }
   temp = ext_ind(fBm_data.D, dim_comp, coord);
   c_im(in_data[temp]) = (FFTW_REAL) 0;
}
/* Use the complex-to-real Fast Fourier Transform package RFFTW */

plan = rfftwnd_create_plan(fBm_data.D, fBm_data.dim, FFTW_BACKWARD,
            FFTW_ESTIMATE | FFTW_IN_PLACE, COMPLEX_TO_REAL);
rfftwnd(plan, 1, in_data, 1, 0, in_data, 1, 0);
rfftwnd_destroy_plan(plan);

/* RFFTW returns data in a peculiar form, the last two bytes in
 * each row are not used. Therefore before further using the data,
 * "pack" it to normal form.
 */

for(i = 0, prod = 1; i < fBm_data.D; i++)
   prod *= fBm_data.dim[i];
```

```
   for(i = 0, j = 0; i < prod; i++)
   {
      if((i % fBm_data.dim[fBm_data.D - 1]) == 0)
         j += 2;
      *(fBm_data.data + i) = *(fBm_data.data + i + j - 2);
   }

   /* Normalise the data to the interval [0,1] */

   for(i = 0, fmin = *(fBm_data.data), fmax = *(fBm_ data.data);
       i < prod; i++)
   {
      if(*(fBm_data.data + i) > fmax)
         fmax = *(fBm_data.data + i);
      if(*(fBm_data.data + i) < fmin)
         fmin = *(fBm_data.data + i);
   }

   for(i = 0; i < prod; i++)
   {
      if(fmax > fmin)
         *(fBm_data.data + i) = (*(fBm_data.data + i) - fmin) / (fmax - fmin);
      else
         *(fBm_data.data + i) = 1.0;
   }

   free(bin_index);
   free(coord);
   free(dim_even);
   free(dim_comp);

   return &fBm_data;
}


/* deltavar: returns the delta variance of a given data structure and
 * size of the "n-sphere" used to convolve with.
 */

float
deltavar(float       L,                /* Radius of the "n-sphere"   */
         fBm_struct  fBm_data)         /* An initiated fBm structure */
{
   unsigned int      *coord;           /* Relevant co-ordinate in fBm-data     */
   unsigned int      *coord_tot;       /* Co-ordinate used in convolution      */
   FBM_PREC          *coordf;          /* Floating co-ordinate                 */
   unsigned int      selements;        /* # of elements in "n-sphere"          */
   unsigned int      *axis_factor;     /* Normalisation factor for axis        */
   unsigned int      positive;         /* # of positive elements in "n-sphere" */
   unsigned int      negative;         /* # of negative elements in "n-sphere" */
   int               *displace;        /* Displacing co-ordnates used in conv  */
   unsigned int      prod;             /* Product of dimensions in fBm data    */
   unsigned int      sprod;            /* Product of dimensions in "n-cube"    */
   unsigned int      *sphereC;         /* Co-ordinates in "n-sphere"           */
   FBM_PREC          *sphereV;         /* Values in "n-sphere"                 */
   FBM_PREC          qsum;             /* Quadratic sum                        */
   FBM_PREC          value;            /* Value of element                     */
   float             variance;         /* The calculated variance              */
   unsigned int      i, j, k;          /* Just dummies                         */


   if((L <= 0) || (L > 1))             /* Limit the scale L to (0,1] */
      return 0;

   if(((displace = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((coord = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((coord_tot =  (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int) )) == NULL) ||
      ((coordf = (FBM_PREC *)
         malloc(fBm_data.D * sizeof(FBM_PREC))) == NULL) ||
      ((axis_factor = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL))
      return (float) 0;

   for(i = 0, prod = 1, sprod = 1; i < fBm_data.D; i++)
   {
      prod *= fBm_data.dim[i];
      axis_factor[i] = (unsigned int) fBm_data.dim[i] * (L * 1.5);
                                /* (L * 1.5) is the radius of the "n-sphere" */
      sprod *= axis_factor[i];
   }
```

```
/* Construct the "n-sphere" to convolve with */

if(((sphereC = (unsigned int *)
      malloc(sprod * sizeof(unsigned int))) == NULL) ||
   ((sphereV = (FBM_PREC *)
      malloc(sprod * sizeof(FBM_PREC))) == NULL))
   return 0;

for(j = 0, selements = 0, positive = 0, negative = 0; j < sprod; j++)
{
   ext_coo(fBm_data.D, axis_factor, coord, j);

   for(k = 0; k < fBm_data.D; k++)
      coordf[k] = ((FBM_PREC) coord[k]) + 0.5;

   /* Calculate the quadratic sum */

   for(k = 0, qsum = 0.0; k < fBm_data.D; k++)
   {
      if(2 * coordf[k] > axis_factor[k])
      {
            qsum += (2.0 * coordf[k] / ((FBM_PREC) axis_factor[k]) - 1.0) *
                    (2.0 * coordf[k] / ((FBM_PREC) axis_factor[k]) - 1.0);
      }
      else
      {
            qsum += (1.0 - 2.0 * coordf[k] / ((FBM_PREC) axis_factor[k])) *
                    (1.0 - 2.0 * coordf[k] / ((FBM_PREC) axis_factor[k]));
      }
   }

   /* Check whether element belongs to "n-sphere" */

   if(qsum <= 1)
   {
      sphereC[selements] = j;
      if(qsum < (1.0 / 9.0))        /* The inner third is positive */
      {
            sphereV[selements] = 1.0;
            positive++;
      }
      else
      {
            sphereV[selements] = -1.0;
            negative++;
      }
      selements++;
   }
}
if(negative == 0)
   negative = 1;

/* Make sure that the integral over the "n-sphere" is zero */

value = -1.0 * ((FBM_PREC) positive) / ((FBM_PREC) negative);
for(i = 0; i < selements; i++)
{
   if(sphereV[i] < 0)
   sphereV[i] = value;
   sphereV[i] /= (FBM_PREC) pow( (double) L, (double) fBm_data.D);
}

/* Convolve the fBm data with the "n-sphere", calculating the
 * variance at the same time to save space */

for(i = 0, qsum = 0; i < prod; i++)   /* For each point...    */
{
   ext_coo(fBm_data.D, fBm_data.dim, coord, i);
   for(j = 0, value = 0; j < selements; j++)        /* For each selement... */
   {
      ext_coo(fBm_data.D, axis_factor, displace, sphereC[j]);
      for(k = 0; k < fBm_data.D; k++)
      {
            coord_tot[k] = (coord[k] + displace[k] +
                  fBm_data.dim[k] - axis_factor[k] / 2) %
                  fBm_data.dim[k];
      }
      k = ext_ind(fBm_data.D, fBm_data.dim, coord_tot);
      value += *(fBm_data.data + k) * sphereV[j];
   }
   qsum += value * value;
}
```

```
   variance = (float) (qsum / (prod - 1));            /* This is the variance *
                                                       * since the mean is 0  */
   free(sphereV);
   free(sphereC);
   free(axis_factor);
   free(coord_tot);
   free(coord);
   free(displace);

   return variance;
}


/* boxcount: Given an initiated fBm structure, number of interval
 * partitions along an axis and a number disp_num defining the number of
 * diplacements, boxcount returns a list of number of boxes needed
 * to cover the fBm structure (when viewed as a fractal hypersurface).
 * The list contains disp_num entries, one for each random displacement
 * of origo.
 */

unsigned int
*boxcount(unsigned int     intervals,  /* Number of intervals per axis        */
          unsigned int     disp_num,   /* Number of displacement iterations   */
          fBm_struct       fBm_data)   /* An initiated fBm structure          */
{
   unsigned int      *tot_box;         /* Total number of covering boxes         */
   unsigned int      *box_coord;       /* Co-ordinate of current box             */
   FBM_PREC          *box_factor;      /* Interval normalisation factor          */
   unsigned int      *displace;        /* Origin displacement                    */
   unsigned int      *coo_term;        /* Co-ordinate terms                      */
   unsigned int      *coo_sum;         /* Sum of co-ordinates                    */
   unsigned int      *coo_dum;         /* Co-ordinate dummy                      */
   FBM_PREC          rho_displace;     /* Displacement in intensity (rho)        */
   unsigned int      rho_max;          /* Maximum intensity in part              */
   unsigned int      rho_min;          /* Minimum intensity in part              */
   unsigned int      delta_rho;        /* Difference between rho_max and rho_min */
   unsigned int      prod;             /* Product of dimensions in fBm data      */
   unsigned int      bprod;            /* Total number of boxes                  */
   unsigned int      i, j, k;          /* Just dummies                           */
   unsigned int      l, iter;          /* More dummies                           */

   if(intervals == 0)      /* Allow only positive number of intervals */
      return (unsigned int *) NULL;

   if(((tot_box = (unsigned int *)
         malloc(disp_num * sizeof(unsigned int))) == NULL)  ||
      ((box_coord = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((box_factor = (FBM_PREC *)
         malloc(fBm_data.D * sizeof(FBM_PREC))) == NULL) ||
      ((coo_term = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((coo_sum = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((coo_dum = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL) ||
      ((displace = (unsigned int *)
         malloc(fBm_data.D * sizeof(unsigned int))) == NULL))
      return (unsigned int *) NULL;

   for(i = 0; i < fBm_data.D; i++)
      box_factor[i] = ((FBM_PREC) fBm_data.dim[i]) / ((FBM_PREC) intervals);

   for(i = 0, prod = 1; i < fBm_data.D; i++)
      prod *= intervals;

   for(iter = 0, rho_max = 0, rho_min = 0; iter < disp_num; iter++)
   {
      for(i = 0; i < fBm_data.D; i++)
      {
         displace[i] = (unsigned int) (((float) rand() /
               (float) RAND_MAX) *
               (float) (box_factor[i] / 2.0));
      }
      rho_displace = (FBM_PREC) ((0.5 / ((float) intervals)) * rand () /
         RAND_MAX);
      for(i = 0, tot_box[iter] = 0; i < prod; i++)     /* For all boxes... */
      {
         for(j = 0, k = i; j < fBm_data.D; j++)
         {
               box_coord[fBm_data.D - 1 - j] = (unsigned int) k % intervals;
               coo_dum[fBm_data.D - 1 - j] = (unsigned int)
                     box_coord[fBm_data.D - 1 - j] *
                     box_factor[fBm_data.D - 1 - j];
               k -= (unsigned int) box_coord[fBm_data.D - 1 - j];
               k /= intervals;
         }
```

```
        for(j = 0, bprod = 1; j < fBm_data.D; j++)
              bprod *= (unsigned int) box_factor[j];

        for(j = 0; j < bprod; j++)    /* Check all rho in a box */
        {
              for(k = 0, l = j; k < fBm_data.D; k++)
              {
                    coo_term[fBm_data.D - 1 - k] = l %
                          ((unsigned int) box_factor[fBm_data.D - 1 - k]);
                    l -= (unsigned int) coo_term[fBm_data.D - 1 - k];
                    l /= (unsigned int) box_factor[fBm_data.D - 1 - k];
              }
              for(k = 0; k < fBm_data.D; k++)
                    coo_sum[k] = (coo_dum[k] + displace[k] + coo_term[k]) %
                          fBm_data.dim[k];
              k = ext_ind(fBm_data.D, fBm_data.dim, coo_sum);

              if(j == 0)
              {
                    rho_max = ((unsigned int)
                          ((*(fBm_data.data + k) +
                          rho_displace) * intervals)) % intervals;
                    rho_min = ((unsigned int)
                          ((*(fBm_data.data + k) +
                          rho_displace) * intervals)) % intervals;
              }
              else
              {
                    if((((unsigned int)
                          ((*(fBm_data.data + k) +
                          rho_displace) * intervals)) % intervals) >
                          rho_max)
                    {
                          rho_max = ((unsigned int)
                                ((*(fBm_data.data + k) +
                                rho_displace) * intervals)) %
                                intervals;
                    }
                    else if((((unsigned int)
                          ((*(fBm_data.data + k) +
                          rho_displace) * intervals)) % intervals) <
                          rho_min)
                    {
                          rho_min = ((unsigned int)
                                ((*(fBm_data.data + k) +
                                rho_displace) * intervals)) %
                                intervals;
                    }
              }
        }
        delta_rho = (unsigned int) (rho_max - rho_min) + 1;
        tot_box[iter] += delta_rho;
     }
   }

   free(displace);
   free(coo_dum);
   free(coo_sum);
   free(coo_term);
   free(box_factor);
   free(box_coord);

   return tot_box;
}


/* project: Given an indata structure, an uninitiated outdata
 * structure and an axis, project projects the data along the axis
 * and writes the resulting projection to the outdata structure
 * which thus has one less dimension.
 */

int
project(fBm_struct   fBm_in,          /* The original fBm structure  */
        fBm_struct   fBm_out,         /* The projected fBm structure */
        unsigned int axis)            /* The axis to integrate along */
{
   unsigned int      prod_in;         /* Product of the in-dimensions  */
   unsigned int      prod_out;        /* Product of the out-dimensions */
   unsigned int      *coord_in;       /* Co-ordinates in indata        */
   unsigned int      *coord_out;      /* Co-ordinates in outdata       */
   FBM_PREC          fmax, fmin;      /* Normalising factors           */
   unsigned int      i, j, k;         /* Just dummies                  */
```

```
   /* Before integrating, make a rough check that
    * inputs are sensible */

   if(fBm_in.data == NULL ||
      fBm_out.data == NULL ||
      fBm_in.dim == NULL ||
      axis >= fBm_in.D)
      return 1;

   if(((coord_in =    (unsigned int *)
         malloc(fBm_in.D * sizeof(unsigned int))) == NULL) ||
      ((coord_out =  (unsigned int *)
         malloc(fBm_in.D * sizeof(unsigned int))) == NULL))
      return 1;

   for(i = 0, prod_in = 1; i < fBm_in.D; i++)
      prod_in *= fBm_in.dim[i];
   prod_out = prod_in / fBm_in.dim[axis];

   for(i = 0; i < prod_out; i++)
      *(fBm_out.data+j) = 0.0;

   for(i = 0; i < prod_in; i++)
   {
      ext_coo(fBm_in.D, fBm_in.dim, coord_in, i);
      for(k = 0, j = 0; k < fBm_in.D; k++)
      {
         if(k != axis)
               coord_out[j++] = coord_in[k];
      }
      j = ext_ind(fBm_in.D, fBm_in.dim, coord_in);
      k = ext_ind(fBm_out.D, fBm_out.dim, coord_out);
      *(fBm_out.data+k) += *(fBm_in.data+j);
   }

   /* Normalise the data to the interval [0,1] */

   for(i = 0, fmin = *(fBm_out.data), fmax = *(fBm_out.data);
      i < prod_out; i++)
   {
      if(*(fBm_out.data + i) > fmax)
         fmax = *(fBm_out.data + i);
      if(*(fBm_out.data + i) < fmin)
         fmin = *(fBm_out.data + i);
   }

   for(i = 0; i < prod_out; i++)
   {
      if(fmax > fmin)
         *(fBm_out.data + i) = (*(fBm_out.data + i) - fmin) / (fmax - fmin);
      else
      *(fBm_out.data + i) = 1.0;
   }

   free(coord_out);
   free(coord_in);

   return 0;
}


/* ext_ind: extract the index of the element on the co-ordinates
 * coord. ext_ind returns the index.
 */

inline unsigned int
ext_ind(unsigned int D,          /* Number of axes in the data array   */
        unsigned int *dim,       /* A pointer to array with dimensions *
                                  * of the different axes              */
        unsigned int *coord)     /* Co-ordinates of element            */
{
   unsigned int i;               /* Just a dummy        */
   unsigned int index;           /* The extracted index */

   for(i = 0, index = 0; i < D; i++)
      index = coord[i] + dim[i] * index;

   return index;
}
```

```
/* ext_coo: extract the co-ordinates of an element with known index.
 * A pointer to the co-ordinate array where the result is stored is
 * given as input as well. ext_coo returns a pointer to the co-ordinate
 * array.
 */

inline unsigned int
*ext_coo(unsigned int D,         /* Number of axes in the data array   */
         unsigned int *dim,      /* A pointer to array with dimensions *
                                  * of the different axes              */
         unsigned int *coord,    /* Co-ordinates of element            */
         unsigned int index)     /* Index of element                   */
{
   signed int        i;               /* Just a dummy  */
   unsigned int      temp_index;      /* Another dummy */

   for(i = D - 1, temp_index = index; i >= 0; i--)
   {
      coord[i] = temp_index % dim[i];
      temp_index -= coord[i];
      temp_index /= dim[i];
   }
   return coord;
}


/*
 * init_gauss: Initialize the pseudo-random number
 * generator gauss() with a positive integer as seed.
 */

void
init_gauss(unsigned int seed)
{
   srand(seed);
}


/*
 * gauss: Returns a N(0,1) real number using a
 * pseudo-random number generator.
 */

inline float
gauss(void)
{
   float U1, U2;     /* Uniform distributed pseudo-random numbers */
   float X;          /* The normal distributed return value */

   U1 = (rand() * 1.0) / RAND_MAX;     /* U1 is U(0,1) */
   U2 = (rand() * 1.0) / RAND_MAX;     /* U2 is U(0,1) */

   X = (float) sqrt(-2.0 * log((double) U1)) * cos(2.0 * PI * (double) U2);

   return X;
}
```

Appendix C - vocabulary

fBm	fractional Brownian motion. See section 3.2.
Commutative	when (A,B) = (B,A).
Complexity	the speed with which a computer increases its calculation time for a given input. Example: $O(n^2)$ means that if you double the number of data, n, the computation time will be four times longer.
DFT	Discrete Fourier Transform . See Gonzales (1987).
Equatorial co-ordinates	a celestial spherical co-ordinate system that relates directions to celestial objects relative to the rotation axis of the Earth and the equinox points.
Equinox points	The two directions in the sky where the Sun is when day and night are of equal length on Earth.
FFT	Fast Fourier Transform. An algorithm for calculating DFT. See Gonzales (1987).
FFTW	Fastest Fourier Transform in the West. An implementation of FFT. See Frigo, Johnson (1998).
FT	Fourier Transform. A mathematical tool used to look at problems from another angle.
Gaussian process	a stochastic process such that every vector consisting of stochastic variables at arbitrary time points is multivariate. See Gut (1995).
GMC	an interstellar Giant Molecular Cloud. See section 1.2.
GNU	a free software foundation.
Group	a set G with a binary operator $\bullet$ such that for all $a, b, c \in G$, (i) $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ (ii) $\exists e \in G: \forall a \in G, e \bullet a = a \bullet e = a$ (iii) $\forall a \in G: \exists a^{-1} \in G, a \bullet a^{-1} = a^{-1} \bullet a = e$.
IMF	the stellar Initial Mass Function. See section 1.4.
IRAS	InfraRed Astronomical Satellite.
ISM	Inter Stellar Medium - the medium between the stars. See section 1.2.
ISO	Infrared Space Observatory.
Isothermal	when the temperature is equal everywhere.
Main sequence	a state where an ordinary star spend most of its lifetime.
Magnitude	a brightness measure. The lower the magnitude, the brighter the star. The faintest stars one can see with unaided eyes are of magnitude 6.
Maple	a mathematics computer program with symbolic handling. By Waterloo Maple Inc.
Matlab	a mathematics and numerical analysis computer program. By The Mathworks Inc.
Measure	a non-negative real valued set function μ on a σ-algebra A such that $\mu(\varnothing) = 0$ and $\mu\left(\bigcup_{n=1}^{\infty} E_n\right) = \sum_{n=1}^{\infty} \mu(E_n)$, $\forall E_n \in A$.
Metric	a real-valued function ρ defined for every pair (x, y) of points in a set such that (i) $\rho(x, y) \geq 0$, $\rho(x, y) = 0 \Leftrightarrow x = y$. (ii) $\rho(x, y) = \rho(y, x)$ (iii) $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.
Modulo n	when one identifies numbers i with $i + n$.
Monotone	when a function is either only increasing or only decreasing.

$N(\mu, \sigma^2)$	normal distribution with mean μ and standard deviation σ.
Optical depth	a measure on how far one can see into a medium.
Optically thin	when the medium is transparent.
Optically thick	when the medium is opaque.
Partition	when you divide a set into parts.
PDMF	Present Day Mass Function. See section 1.4.
Pseudo-metric	same as metric, except that we do not require $\rho(x, y) = 0 \Leftrightarrow x = y$.
Power spectrum	the square of the Fourier spectrum
σ-algebra	a class A of subsets of a set X such that $x, y \in A \Rightarrow$ (a) $\varnothing \in A$ (b) $x - y \in A$ (c) $x \cup y \in A$ (d) $X \in A$ (e) $A_n \in A \Rightarrow \bigcup_{n=1}^{\infty} A_n \in A$.
Sample space	the space of all possible outcomes of a stochastic variable.
Spectral density	the derivative of the power spectrum
Stochastic process	an indexed collection of stochastic variables with the same sample space.
Translation	when moving a distance without rotating or scaling.
$U(a, b)$	uniform distribution over the interval (a, b).
Variance	a measure of the spread of values in a sample.
Vector space	a real vector space V is an abelian (commutative) group with a scalar product such that for all α, $\beta \in \boldsymbol{R}$ and $u, v \in V$: *(i)* $\alpha(\beta v) = (\alpha\beta)v$ *(ii)* $(\alpha + \beta)v = \alpha v + \beta v$ *(iii)* $\alpha(u + v) = \alpha u + \alpha v$ *(iv)* $1v = v$.

Captions for the colour plates

Colour plate 1 and 2

Colour plate 1 is an fBm coloured to resemble real terrestrial clouds. An example of some real terrestrial clouds is provided by colour plate 2, which is a photograph.

Colour plate 3 and 4

An fBm image with Hurst exponent 0.5 can be seen in colour plate 3. I have convolved the image with a Gaussian beam for a more realistic look. Colour plate 4 is a real cloud as observed by IRAS at wavelength 100 μm. It is a part the cloud of Lupus, centred at the approximate equatorial co-ordinates $15^h\ 40^m$, -35° 00’.