Convection Cells in a 3D Hydrodynamical Model of Betelgeuse

Sigurður Finnsson

June 23, 2003

Contents

Chapter 1

Abstract

Observations of the M supergiant Betelgeuse have shown asymmetries attributed to evolving bright spots on the surface. In an article by Schwarzschild (1975) he puts forward arguments for only few giant convection cells covering the surface of supergiants. A supergiant has been simulated by B. Freytag using a 3D hydrodynamical simulation (called CO^{5}BOLD). The code indeed shows only few giant convection cells on the surface of the star. Furthermore, the models give hint of a hierarchical structure of the convection cells where smaller cells are superposed on bigger cells. This report makes an effort to analyse the structure of the convective cells and to measure the wide range of spatial scales spanned by the cells in the models. It is shown that the cells exist on very different scales which makes it hard to define a typical "mean" size of cells, giving support to the hierarchy hypothesis. Spectra of spatial scales are presented for two models having different numerical resolution. In addition an overview is given of a 3D visualisation tool programmed in IDL by the author in order to help with the analysis of the models.

This report is a master thesis done under the supervision of B. Freytag.

Chapter 2

Introduction, the red supergiant

In the constellation Orion lies Betelgeuse, a red supergiant of type M1-2 Ia-Iab (Gray, 2001). The radius is heavily dependent on wavelength, Gray (2000) gives it simply as "some 800 times larger than the Sun", the mass is not known but in the same article Gray mentions that most researches prefer 10-20 $M_\odot$. The Hipparcos satellite has measured its distance to be 131 ± 30 pc. The uncertainty is so large due to bright asymmetrical surface features which could disturb the position measurements of the satellite (Gray, 2000). The same article also places the surface temperature to roughly 3600 K.

Even if the star is so far away the immense size makes it an ideal star for studying since it has one of the largest apparent disks in the sky. Its radius in the visual is roughly 55 mas (Gray, 2000). This is big enough for studying of large scale features using interferometers (see e.g. Buscher et al., 1990), telescopes such as the Hubble Space Telescope can even be used at shorter wavelengths where the apparent radius is larger (Gilliland and Dupree, 1996).

2.1 Surface Granules

A star of this size has much lower surface gravity than our Sun. This gives hint of a very different "landscape" on the surface. Below I will go through scaling arguments put forward by Schwarzschild (1975) which support few giant convection cells being present simultaneously on the surface of a supergiant. Then I briefly discuss some observations of Betelgeuse. Observers have detected asymmetrical features on the surface which are usually interpreted as giant convection cells by most observers (but not all, see Gray (2000)).

2.2 Scaling arguments for big surface granules

Schwarzschild (1975) theorises about the photospheric structure of the Sun. He identifies potential quantities which control the scale of convection cells and compares with the few available numerical models of supergiants at the time (1974). He identifies the same quantities in the envelopes of the model supergiants and finds a new relative scale for their convection cells. The results turn out to be quite different from the Sun.

2.2.1 Solar granules

Schwarzschild began by analysing the surface granules on the Sun and used a typical diameter of 2000 km. He estimated the diameter/depth ratio of a granule granule to be 3. He did not justify this choice except mentioning that this number fits various physical phenomena. Using this he estimated the granules, or convection cells, to reach typical depths of 700 km. He used this figure to relate the convection cells to the change in various subphotospheric quantities using Solar models.

Figure 2.1 shows a diagram from his article. It is a Solar model constructed by R. Härm[1]. The zero of the depth scale is set at $\tau = 1$. The estimated typical depth of the granules is marked in the lowest part of the diagram. The important physical parameters here are pressure, density and QTS.

QTS, the super-adiabatic temperature gradient

What is QTS? It is based on the temperature gradients in a star. A temperature gradient ∇ in a gas is defined as

$$\nabla = \frac{\mathrm{d}\ln T}{\mathrm{d}\ln P}.$$

Since the pressure P is usually rising monotonically with depth ∇ is, in a way, the increase of temperature with depth.

A layer of gas is unstable against convection when the surrounding temperature gradient ∇_{surr} is higher than the adiabatic temperature gradient ∇_{ad}. This condition implies that a rising gas bubble (which has a temperature gradient close to ∇_{ad}) will cool less than its surroundings. Assuming the bubble will always have the same pressure as the surrounding gas its higher temperature causes it to have a lower density and therefore rise still higher because of buoyancy forces. (Here it is assumed for simplicity that the mean molecular weight is the same in the layers,

[1]No bibliographic reference is given for the model except that it is based on code for hydrogen and helium supplied by A. Sweigart

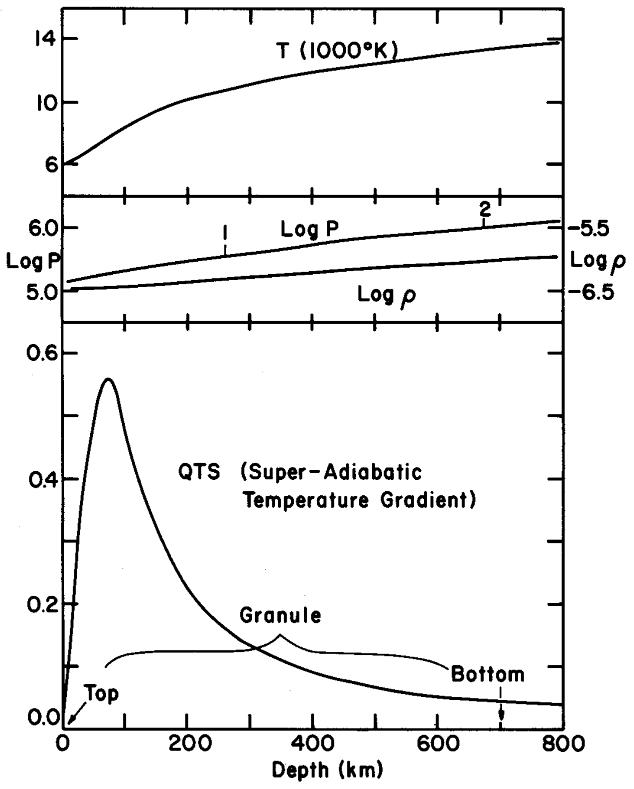


Figure 2.1: A solar model showing the relevant physical quantities that control the scale of granules. The diagram is taken directly from Schwarzschild (1975).

but generally that must be checked as well.). Therefore the condition for a layer to be convectionally unstable is

$$\nabla_{\text{surr}} > \nabla_{\text{ad}}.$$

Furthermore, this reasoning implies that the strength of the buoyancy force is closely related to the temperature difference between the bubble and the surrounding gas. A higher difference implies the bubble to be lighter and therefore rising more easily. The quantity

$$\nabla_{\text{surr}} - \nabla_{\text{ad}}$$

gives a measure of the "strength" of convective driving, large positive values should imply large temperature differences and strong convection.

This is very similar to the definition of QTS except it is defined as the difference between the *actual* temperature gradient ∇ and the adiabatic one. The actual temperature gradient is the effect of all acting energy fluxes in the gas, i.e. radiation, convection and heat conduction. The definition of QTS is therefore

$$\text{QTS} = \nabla - \nabla_{\text{ad}}.$$

Therefore QTS must be positive for convection to be able to take place. A very high positive QTS implies strong convection.

Quantities controlling solar granules

Figure 2.1 shows that the granules extend vertically over roughly 2 pressure or 1 density scale height(s). This hints that the extent of granules is on the order of one scale height of pressure and/or density. QTS has a prominent maximum just below the surface. The granule ends where QTS has become flattened and low (i.e. where it is roughly 10% of the maximum height). This made Schwarzschild assume the QTS plays an important role in determining the depth of the granules.

2.2.2 Solar supergranules

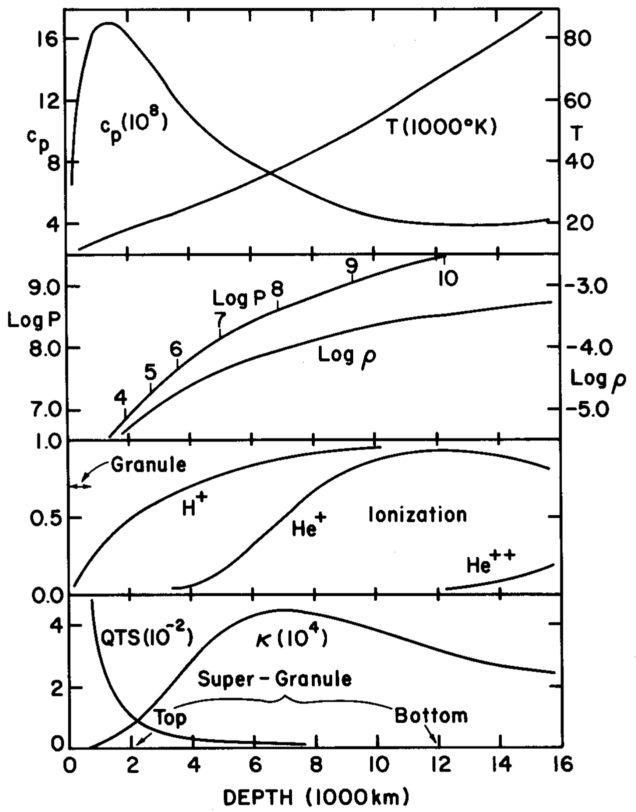


Figure 2.2: A solar model showing the physical quantities relevant to supergranules. The diagram is taken directly from Schwarzschild (1975).

What are solar supergranules? They are not seen directly in the granulation pattern of the Sun. In order to find them the horizontal velocity field is measured. Over time the velocity field is seen to converge in superstructures each several granules in size. This is taken as indication of some bigger, underlying granules, i.e. supergranules.

Schwarzschild used a typical diameter of 30000 km for the supergranules. Again he assumed the diameter/depth ratio of 3 which gave a depth of 10000 km. He estimated the top of the supergranules to lie at a depth of 2000km, arguing that any higher value would make their presence more obvious on the surface while being lower would make the supergranule pattern disappear altogether. So the "typical" supergranule is assumed to stretch from a depth of 12000 km up to 2000 km.

Now it is possible to look for relevant physical structures which control the extent of supergranules. The diagram showing the relevant physical quantites from the solar model is shown in figure 2.2. The estimated extent of the supergranules is shown at the bottom of the figure.

Note that the supergranules stretch over several pressure scale heights. This is in disagreement with the hypothesis that granules are on the order of 1-2 scale heights. It is also of note that QTS is low inside the supergranules since there we have the first ionization zones of hydrogen and helium. Ionization zones make ∇_{ad} low while the increased opacity tend to make ∇_{rad} high so one would expect QTS to increase in these zones. This is unexpected and implies that something is not accounted for. Schwarzschild notes this and warns that this is all very hypothetical as it is, but adds that since $\nabla_{\mathrm{rad}} > \nabla_{\mathrm{ad}}$ (which is a basic condition of convection) we could expect some convective activity in these layers.

He also speculates that the rule of convection cells being on the order of 1-2 pressure scale heights is only valid where QTS is very high. Since it is very low in the supergranules it could explain why they stretch over so many scale heights.

This analysis gave him the following quantities potentially controlling the scale of granules:

- Pressure and/or density scale heights, at least where QTS is high.
- High maxima of QTS
- High opacity
- The first hydrogen and helium ionization zones.

Armed with these results he looked for the same quantities in the model of a red giant.

2.2.3 Red-giant envelopes

Schwarzschild used an available red-giant model at that time (year 1974) which had a mass of 0.65 $M_{\odot}$. This mass is very low but the model was chosen since the availible data from it was very detailed. The most relevant physical parameters from the model are shown in figure 2.3. The red-giant was assumed to have radius of $160 \cdot 10^6$ km.

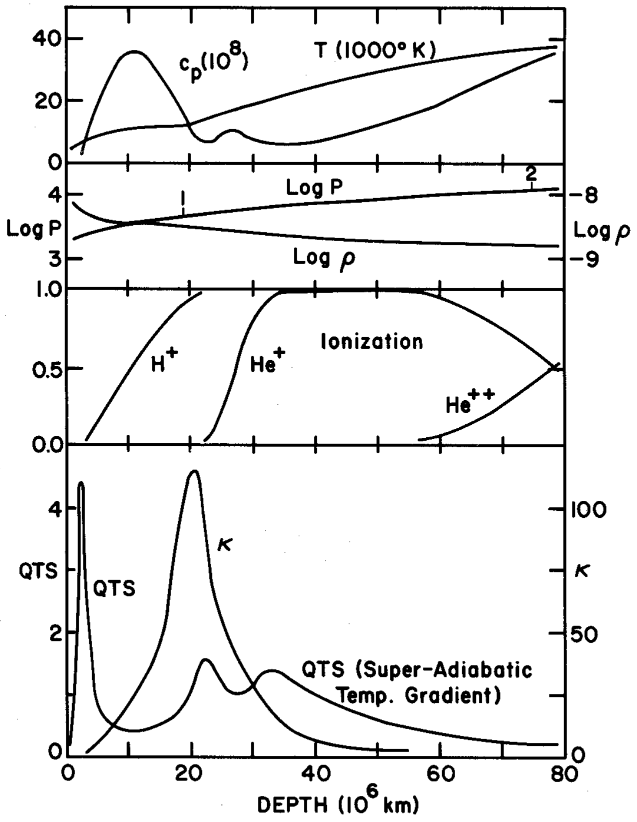


Figure 2.3: The red giant model. Here are shown the relevant physical quantities which potentially control the scale of granules. The diagram is taken directly from Schwarzschild (1975).

It is very important to notice the scale of figure 2.3 which is much larger than for figure 2.2 of the solar model. Figure 2.3 reaches 50% of the radius, while figure 2.2 reaches only 2.5%. This is the single most important factor to notice, all physical changes in the envelope of the red giant happen over much longer spatial scales relative to the radius. Just this fact alone implies the convective structure to be quite different and having bigger relative scales than on the Sun.

The pressure scale height is also very different. Down to 50% of the radius there are only roughly 2 scale heights in the red giant, while the Sun has 11 pressure scale heights over only the first 2.5%. The QTS has a similar maxima near the surface as in the Sun, but it is much stronger. Generally it declines with depth except for two local maxima in and near the opacity maximum. Schwarzschild thinks these two local maxima are artifacts caused by applying a theory of convection (i.e. the mixing-length theory) which is probably not suitable for red giant envelopes.

The QTS is strong in the red giant. Over the first 40% of the radius it is stronger than the subphotospheric maxima of QTS in the solar model. The paper states this as an effect of the very low densities in the envelope of the red giant, causing it to be convectively inefficient.

The minimum possible size of granules

First Schwarzschild asked what would be the minimum size of the granules this comparison implied. In figure 2.1 the size of granules in the Sun follow the QTS maxima. The same comparison in figure 2.3 should therefore give a possible size of the granules on the surface of the red giant.

Schwarzschild chose depth the depth $10 \cdot 10^6$km for the granules. This depth covers the QTS maxima and the first half of the hydrogen ionization zone and stops just before reaching the maximum of c_p. This is in accordance with the behaviour in the Solar case. However, the total pressure scale height is only a fraction of unity, i.e. much less than for the granules in the Sun model. Therefore it is reasonable to think of this depth as the absolute minimum the granules can have.

Using the diameter/depth ratio of 3 as for the Sun gives granules with diameter of $30 \cdot 10^6$ km. This gives roughly 400 granules covering the surface of the red giant which is wastly different from the 2 million or so granules covering the surface of the Sun. Still this value is the absolute maximum number of granules according to the data.

Finding a more realistic size of the granules

These estimates ignore the fact that QTS is very high down to deep layers of the red giant. In addition the pressure scale height is increasing slowly so there are only 2 scale heights at depth of 50% radius.

In the solar model the lower end of a convection cell was roughly where QTS became 10% of its maximum value. Doing the same estimate for the red giant gives the depth $60 \cdot 10^6$ km. Furthermore, since there are only few scale heights down to this depth this it is very likely that convection cells actually could reach so deep. Using the diameter/depth ratio of 3 as usual gives a convection cell with a diameter of $180 \cdot 10^6$ km, which is larger than the radius of the star.

This last estimate implies the surface is covered with only 12 convection cells.

Convection cells on the surface of a more realistic model of a $15\,M_\odot$ red supergiant

Schwarzschild ended his reasoning by using the same arguments on a model of a $15\,M_\odot$ red supergiant. This is a model made by Stothers (1972). Doing the same analysis gave 90 convection cells covering the whole surface.

The dominant size of convection cells on red giants

These estimates gave the possible spatial scales of convection cells in a red giant. However, giving that QTS is very strong in large parts of the envelope makes one suspect that the mechanism driving the biggest convection cells could be strong and dominant, therefore favouring giant convection cells.

Later in this report it will be shown how the cells in a supergiant model actually seem to be spread over a range of spatial scales.

2.3 Observations of large scale activity

2.3.1 Direct Imaging of a bright spot

Gilliland and Dupree (1996) presented the first direct imaging of the surface of Betelguese. The observation is done with Hubble in UV light. As mentioned before the observed radius is very dependent on wavelength and in UV the star has roughly two times the radius as in the visual. This makes it easier to resolve large structures on the surface. In this observation a huge bright spot was found. This is in line with bright spots detected using interferometric observations (see below).

2.3.2 Variations in light flux and radial velocity

If the pulsations of Betelgeuse were purely global in nature one would expect the light and radial-velocity curves to follow each other (Sanford, 1933). Decreasing diameter would imply increasing light due to increasing temperature. Such a pattern, however, is not convincingly observed.

Betelgeuse is a semi-regular variable star. It is possible to find a period of roughly six years in the measured data although it is by no means a clear period followed continuously by the star. Over extended periods, e.g. a whole decade, the star can become very irregular with very little hint of periodicity at all. But afterwards it returns into its semi-periodic state of roughly 6 years, although the cycles can be phase-shifted relative to former cycles.

Goldberg (1984) gives an overview of photometric observations of Betelgeuese since early 20th century. Goldberg compares the light curves with available radial velocity measurements of the star. The data he uses is from several different sources and therefore has varying quality.

Regular or not so regular pulsations

In the first quarter of the 20th century Stebbins and Huffer (1931) made a 14 year long photoelectrical (i.e. light flux) measurements of the variations in Betelguese. After doing the measurements they said that "there is no law or order in the rapid changes of Betelgeuse". But to get some hints about the behaviour they smoothed the data heavily and fitted it with a mean curve with a period of 5.4 years. However, Jones (1928) had made radial velocity measurements which he fitted with a period of 5.781 years.

Periods of irregularity

The activity changes over many years. Goldberg (1984) talks about periods where light and/or radial velocity curves are seemingly random. These periods stretch over many years, sometimes over a decade. Often this random behaviour is observed simultaneously in both the light curves and radial velocity, but sometimes the radial velocity behaves erratically while the light curve does not.

Goldberg theorises that these irregular periods could be caused by increased activity on the surface. That could give local variations strong enough to mask the global pulsations. When finally the activity decreases the pulsations again become dominant and regularity settles in. Still, the irregular component is always present and could be caused by the ever present local activity which never completely ceases. This is in line with Schwarzschilds conclusions above.

2.3.3 Optical interferometry

As mentioned above then the small apparent disk of Betelgeuse (55 mas) makes it hard to use telescopes for direct observation of structures. However, interferometers can be used to detect structures below this scale.

Several interferometric observers have reported large scale asymmetries on the surface of the star. Wilson et al. (1997) briefs four observations done over an 8 week interval. These observations found structures changing both in radius and angle which is necessary to confidentally detect asymmetries. In all of the observations they found evidence of asymmetric features. To explain the data a model is fitted to the parameters. Their approach was to search for the simplest possible model they could find which matched the data adequately well. The model

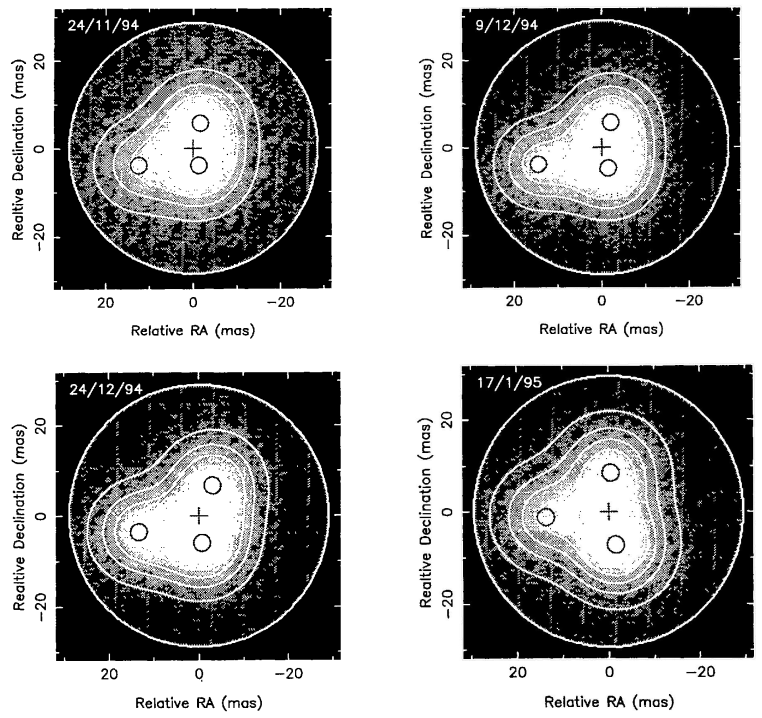


Figure 2.4: A reconstruction of interferometric observations reported in Wilson et al. (1997). These images are taken from the article and have been edited (inverted). They show the brightness distribution measured over the stellar disk at each of the four observations. The small circles denote the center of the bright spots used to model the data. Note how the spots change their position over time. These images represent a possible reconstruction and are not necessarily giving the true picture. The contours mark 30% to 90% of the maximum flux.

they settled on consisted of five components, a disk corresponding to the stellar disk, a larger symmetric component corresponding to a possible dust shell and finally three smaller bright spots which were well within the stellar disk. These spots are asymmetrically placed on the star, see figure 2.4.

The asymmetric features changed orientation between observations. The figure shows the center of the three spots that were used to model the observational data. These movements imply large-scale surface structures changing considerably over the timescale of the observations (8 weeks).

This is the smallest detail which can be observed using current interferometric techniques. Therefore it is impossible to know more about the actual geometrical form of these spots. In the article they chose to model the spots as gaussian disks (i.e. the brightness falls off radially in a gaussian manner). However the spots could be the brightest parts of an irregular patch changing form. It is impossible to know based on this data. These spots are usually interpreted as showing convective activity.

Observations of other supergiants have lead to similar results. Tuthill et al. (1999) lists several observations of five giants or supergiants of Mira variable type. As with Betelgeuse they detected asymmetrical structures in each of those stars. Their data could usually be fitted with models using one or more bright spots. This lends further support to the theory of large-scale surface structures being common on giants or supergiants.

Dust shells

Dust shells have been detected around Betelgeuse. These shells are most likely caused by the global pulsations of the stars as discussed above and e.g. in Goldberg (1984).

The dust shells are asymmetric (Bloemhof et al., 1984). The asymmetric distribution was not discussed, but any asymmetry implies non homogenous outflow of matter from the surface. One possible explanation could be cell structure on the surface affecting the matter outflow. In a way the asymmetries could be a “frosen” picture of the cell structure at the time of ejection.

2.3.4 Spectra

Arguments against giant convection cells are found in Gray (2000, 2001). He assumes that the spectra should show evidences for convection cells. If we assume that at any given time the surface is covered by relatively few convection cells then it could be reflected in the Doppler shifts of spectral lines. E.g. if only 2-4 cells are visible then each cell produces a distinct Doppler shift of the spectral lines. This would cause the same spectral line to appear at several different frequencies (one for each cell), such a few cells should therefore be easily detectable. These shifts would change with time and reflect the appearance and disappearance of the convection cells.

If on the other hand the surface is covered with many cells (several hundreds or more) the shifted lines become so many that the observed spectra would only show the smeared out superposition of the lines, it would be impossible to isolate the individual cells from the spectra.

Some words of thought: this interpretation of convection cells assumes some uniformity in the velocity of gas in each convection cell, furthermore it assumes all cells are continuously rising and falling and that they are sufficiently well represented in the intensity pattern (for the Doppler shifts to be clearly observable). These facts are by no means safe to assume and the models discussed later do not seem to support this picture.

Temporal change in width of spectral lines

In Gray (2001) the time variations of spectral lines are measured in order to find hints of giant convection cells. In his observations Gray did not find any of the features he expected assuming only few convection cells. He found that the line profiles are changing with time but in a very subtle way. The spectral lines changed widths but there were no discernible structures forming or disappearing. He even commented on how remarkably smooth they were. He concluded that the observed changes in spectra did not support the idea of only few giant convection cells existing simultaneously.

Gray measured a period of (subtle) narrowing of the line profiles. During this time the Doppler shift distribution decreased overall. He even found (more uncertain) evidence of another period where the Doppler shift distribution increased slightly. He could not explain how a star can change its overall Doppler shift distribution in this way but he speculated that if the roughly 4% shift in the measured Doppler distribution was due to statistical variation in the number of convection cells then it means that there were over 600 cells on the visible side of Betelgeuse alone. However, this figure is *very* speculative.

2.3.5 Bright spots, windows into deeper layers?

Betelgeuse shows strong brightness variations as discussed above. In Gray (2000) he describes how the spectral lines change depth with time. As a rule the lines deepen with decreasing brightness and weaken during brightness increase. To explain this and other observations he put forward the *Opacity Hypothesis.* He theorises that some disturbance in the atmosphere of the star spreads out and increases the continuous opacity more or less globally. This would cause less flux to go through and the star would dim. The increased opacity of the continuum would add to the opacity of the lines and cause them to deepen. The same effect in reverse would happen when the continuous opacity decreases again.

He uses this to explain the appearance of bright spots on the surface. The bright spots would be regions where the continuous opacity does not change. The spots brighten because the surface all around gets darker but they stay the same. And similarly they disappear when the opacity decreases and the surrounding surface increases in brightness back to the level of the spots.

In his view this could both explain the big large spot observed in some observations and the group of spots observed above.

He gives support for his hypothesis by referring to past observations. E.g. he states that the bright spot detected in Buscher et al. (1990) occurs at minimum brightness as expected. Other observations he mentions are done by Uitenbroek et al. (1998) and Klückers et al. (1997). In Burns et al. (1997) no spots are detected and the observations took place near a brightness maximum, i.e. when the opacity would be minimum and spots would vanish according to the hypothesis.

As discussed above Wilson et al. (1997) found three spots (or possibly bright irregular patches). These spots increase in relative flux while the star is dimming, again as expected.

Gray points out that interferometric observations usually interpret the data to indicate relative flux of the bright spot (or spots) to be up to 20% of the total (e.g. in Wilson et al. (1997) it is 20% and in Buscher et al. (1990) it is 10-15%), if the Opacity Hypothesis is correct this implies mean opacity change of up to 20% as well, which was the amount measured by Gray (2000).

2.3.6 Polarization

If the surface has bright asymmetrical features it would be expected to affect the polarization of light. One commonly mentioned mechanism is the illumination of scattering particles from below, e.g. the photosphere illuminates material in the chromosphere. This will cause polarization if the phenomenon causing the illumination is asymmetric. For a theoretical discussion about this effect see e.g. Al-Malki et al. (1999) where polarization of scattered light from an anisotropic stellar light source is discussed. The basic radiative transfer theory of polarized light due to Rayleigh scattering is treated by Harrington (1970), where a grey atmosphere is assumed.

In Hayes (1984) observations of optical linear polarization are briefed. He found that the light showed "ordered changes" of polarization. The polarization was shown to evolve with time, increase in strength and changing angle gradually. He concluded that these changes were sufficiently regular to be of statistical importance, i.e. they are most likely reflecting some evolving phenomenon in the star. He suggested that these changes could be due to convective activity in the photosphere, where the convection cells are creating bright patches of light illuminating the material above. But an alternative explanation could be magnetic activity in the star. He could not conclude which process was more likely based on his data.

Chapter 3

Modelling supergiants in 3D

After the discussion in previous sections it is obvious that Betelgeuse and probably most supergiants are dynamically active stars. Therefore, a dynamical simulation is needed to account for the observed phenomena. On the Sun it is usually most important to model the topmost granular layer in the photosphere. The fact that this topmost layer is very thin is used to simplify the modelling and since the granules are very small (relative the total surface area) it is sufficient to model a very small rectangular region on the surface with periodic boundaries.

A supergiant, however, has a radically different atmosphere which can be seen by comparing figures 2.2 and 2.3 as discussed in section 2.2.3. The scales of convection are deep reaching and we can expect convection cells which cover big fractions of the surface. This makes it impossible to get a decent dynamical model of a supergiant using the traditional techniques for the Sun.

3.1 Making 3D models with CO^{5}BOLD

In this report I have been working with data from CO^{5}BOLD. This is a simulator created in Fortran90 by my supervisor, Bernd Freytag. The code can simulate either an entire supergiant or, in the case of a Sun like star, a small patch of the surface. The data I have been working with is created using CO^{5}BOLD in supergiant mode. The description which follows of CO^{5}BOLD is based on information in Freytag (2001) and Freytag et al. (2002) while some information is based on discussions between me and Freytag. CO^{5}BOLD is an acronym for "COnservative COde for the COmputation of COmpressible COnvection in a BOx of L Dimensions, L=2,3".

3.1.1 Having a whole star in a box

As discussed above it is impossible to create a decent dynamical model of a supergiant by modelling only a part of it. A single convection cell is expected to be sufficiently big to influence most other parts of the stars. Therefore CO^{5}BOLD simulates the entire envelope of the star (with energy generation in the core as a boundary condition on entropy).

The whole star is enclosed in a cubic volume using a cartesian grid. The models I analyse in this report are done on an equidistant grid but the code supports non-equidistant grids as well. The six sides of the cubic volume are open for transmitting waves to minimise feedback effects from shocks and outflowing matter. All boundaries are equivalent. This method, to have the whole star enclosed in a cubic volume is sometimes referred to as having a *star-in-a-box.*

3.1.2 Gravity

The gravity is implemented using a constant external potential. No self-gravity of the gas is implemented. The potential is spherical and smoothed to avoid the singularity in the center

$$\Phi(r) = \frac{GM_*}{\left[r_0^4 + r^4/\sqrt{1+\left(\frac{r}{r_1}\right)^8}\right]^{1/4}}$$

Here r_0 and r_1 are smoothing constants which are chosen by the user where $r_0 < r_1$. M_* is the (initial) mass of the star.

In general, if

$$r_0^4 \ll r^4 \ll r_1^4$$

then

$$\Phi(r) \approx \frac{GM_*}{r}$$

which is the potential *outside* a homogenous spherical mass. Often the models use $r_0 = 0.2R_*$ (Freytag et al., 2002) where R_* is roughly the radius of the star. This means the potential reflects that most of the stars mass is kept within the core.

Closer to the center we have

$$\Phi(r \to 0) \to \frac{GM_*}{r_0}.$$

This is the maximum possible value of the potential. This is done to avoid the $1/r$ singularity in the center. Moving very far away gives the minimum value of the potential

$$\Phi(r \to \infty) \to \frac{GM_*}{r_1}.$$

This flattens out the potential and can be used to minimize the drop in density in the outer corner of the boxed volume (Freytag et al., 2002).

3.1.3 Operator Splitting

In essence the equations of radiative transfer and compressible hydrodynamics (with gravity) are non-linear and coupled. However, the code uses an approximative technique called *operator splitting* to make it possible to treat these as individual parts in the algorithm. This modularity has the added benefit of making the code easier to maintain.

3.1.4 Hydrodynamics

The hydrodynamical equations include compressible hydrodynamics under the influence of gravity but ignore radiation pressure. The operator splitting is used to reduce the problem of solving the full 3D hydrodynamical equations into three simpler 1D sub-steps, this is referred to as a *directional splitting.* The solver used is an approximate Riemann solver of Roe type which has been modified to include the physics used in the simulator.

Additionally a 3D viscosity tensor can be used for extra smoothing.

3.1.5 Radiative transport

The radiative transport is strictly LTE, i.e. non-LTE effects and scattering are ignored. Opacities are frequency-independent (i.e. grey) in the models analysed here although CO^{5}BOLD supports frequency dependence.

Short time-steps

In supergiants the radiative time scale is very short compared to the hydrodynamical one (Freytag et al., 2002). For simulations this means it is impractical to make the simulator solve for both radiative transfer and hydrodynamics in every step. The time steps would have to be short enough to include important changes in the radiative transfer, i.e. they would be controlled by the radiative time scale. This would mean the hydrodynamical quantities would be practically constant over many time steps.

Instead, in order to save processor power CO^{5}BOLD calculates the radiative transport in many sub-time steps for each hydrodynamical time step.

Short-characteristic scheme

The code supports both a short and a long characteristics scheme of radiative transfer although the models analysed in this report use strictly the short-characteristic scheme. In the short-characteristic scheme the rays are only sent between neighbouring grid cells during each iteration.

In a real star there would be (practically) an infinite number of rays leaving each point at every time. But that is impossible to do numerically. So instead the code sends only few rays at each sub radiative time-step from each grid cell. The directions of the rays are varied between the sub time-steps. This is done so that after many sub time-steps the rays are spread as evenly as possible into all directions.

This results in energy transfer between grid cells. The energy change has to be weighted back conservatively to the relevant grid cells, this is a very delicate step where one has to assume some notion of geometrical form of the grid cells in order to distribute the energy correctly. Incorrect weighting can lead to errors such as anisotropic spreading of energy (e.g. more weighting on axial directions.).

Opacities

The opacities are tabulated grey Rosseland means. The opacity tables are provided by Hans-Günter Ludwig who based it on PHOENIX (Hauschildt et al., 1997) and the OPAL opacity tables (Iglesias et al., 1992).

3.1.6 Energy generation in the core

The code simulates nuclear energy production in the core by keeping the entropy within radius r_0 (as defined in section 3.1.2) near a specified value. This keeps the (inner) energy up in the core.

3.1.7 The equation of state

The equation of state (EOS) takes the ionization of hydrogen and helium into account. This is done by using a lookup table which gives the pressure (and derivatives) given the density and internal energy.

3.1.8 Conserving quantities

When using a discrete grid care must be taken not to loose mass or energy when it flows between grid cells. The discretisized equations have been formulated so they conserve key quantities while still allowing source terms due to energy generation in the core. This makes sure that e.g. mass and energy can only change if there is flow through the boundaries of the volume.

This is particularly important when shocks are expected (as in the case of supergiants) and when using low-resolution grids (Freytag, 2001).

3.1.9 Using many processors in parallel

The code is parallelized, i.e. it can split the workload between many individual processors. The resulting 1D problems can be solved independently and are therefore ideal to for parallelizing. The code collects the 1D problems in chunks which are distributed between available processors. As of now the code only supports shared memory architectures.

Chapter 4

Facts about the analysed models

The models which are analyzed in this report are listed here for future reference.

4.1 Reference list of models

In the report I will refer to the models by their nicknames since their actual names are confusing to use in a written text. However, it is necessary to list the actual names since the nicknames are only used here and are not in context with other reports or information about the models. Here follows a list of the models alongside some important parameters and properties.

LORESI

LORESI is a nickname for model st35gm04n03. The initial condition is made by a one dimensional model which means it begins spherically symmetric. Therefore it takes some years (model-time) until instabilities due to inadequate initial conditions have settled down. Because of this most results are displayed using only frames nr. 95 to 259, which means the first six years are ignored. This choice was done after looking at figure 7.4 which shows the full sequence of frames, there the effect of the initial condition is obvious.

Resolution of grid:	127^3
Viscosity:	1.2
Side length of box:	$2 \times 815\,\mathrm{R}_\odot$
Reconstruction Method:	MinMod
Mass:	$5\,\mathrm{M}_\odot$
Radius of star:	$\approx 630\,\mathrm{R}_\odot$
$\log g$ (cgs units):	-0.46
Entropy within r_0:	$\approx 3.25 \times 10^9$ erg $\mathrm{K}^{-1}\mathrm{g}^{-1}$

Total luminosity:	45000 $L_\odot$
Number of frames:	259
Total model time:	ca. 16 years

MEDRESI

MEDRESI is a nickname for model st35gm04n04. This model uses a frame from LORESI as initial condition, which can potentially affect the analysis since some structures formed in LORESI could live on in MEDRESI.

Resolution of grid:	171^3
Viscosity:	1.2
Side length of box:	$2 \times 813\,R_\odot$
Reconstruction Method:	MinMod
Mass:	5 $M_\odot$
Radius of star:	$\approx 630\,R_\odot$
Entropy within r_0:	$\approx 3.25 \times 10^9$ erg $K^{-1}g^{-1}$
$\log g$ (cgs units):	$\approx$ -0.46
Total luminosity:	46000 $L_\odot$
Number of frames:	162
Total model time:	ca. 10 years

MEDRESII

MEDRESII is a nickname for model st35gm04n06. I have only used this model to make some figures but have not used it for analysing.

Resolution of grid:	171^3
Viscosity:	0.6
Side length of box:	$2 \times 813\,R_\odot$
Reconstruction Method:	VanLeer
Mass:	5 $M_\odot$
Radius of star:	$\approx 600\,R_\odot$
$\log g$ (cgs units):	$\approx$ -0.42
Entropy within r_0:	$\approx 3.25 \times 10^9$ erg $K^{-1}g^{-1}$
$T_{\rm eff}$:	≈ 3340 K
Total luminosity:	43500 $L_\odot$
Number of frames:	96
Total model time:	ca. 6 years

4.1.1 Referring to individual frames in a model

Each model is a time sequence of frames (i.e. array of 3D voxels) containing the state of the model. To refer to an individual frame two numbers are needed, first number is the number of the dataset file containing the frame. Each dataset file is a collection of several frames. The second number is the frame number within the dataset file with the first frame in the dataset file starting with index zero.

Writing e.g. LORESI_22n3 means I am referring to frame 3 within dataset file 22 in model st35gm04n03.

Chapter 5

Convection cells in the models

5.1 The look of a supergiant?

How does the star look like in the simulations? Figure 5.1 shows output from the MEDRESII model. The left image shows the “surface” at optical depth unity (see section 5.2 for explanation). The colors represent T^4 which is proportional to the (bolometric) intensity via the Stefan-Boltzmann law (i.e. assuming black-body radiation implies $I \propto T^4$). The image at right shows an isosurface of the entropy which outlines the convection cells more clearly, sinking gas in the downdrafts tends to conserve its low entropy thus making a relatively clear boundary. All data above optical depth unity has been cut away.

5.1.1 Finding the convection cells

The question is if the convection cells are easy to detect in observations? The consensus seems to be for clearly distinct convection cells showing up as bright patches on the surface. However the simulations show a different picture. Figure 5.1 (left) shows that the surface pattern clearly has large scale bright and dark patches. But how well does it correspond to the actual convection cells? In figure 5.1 (right) the convection cells are easy to find (both figures show the same point of view). The biggest cells are found by tracing the deepest lanes of downdrafts, those cells are so large that only 2-4 cover the surface at any one time. Comparing the two figures side by side shows that it is hard to see the outlines of the convection cells simply by looking at the observed intensity alone. In particular low resolution observations would find it impossible to detect with certainty the biggest cells.

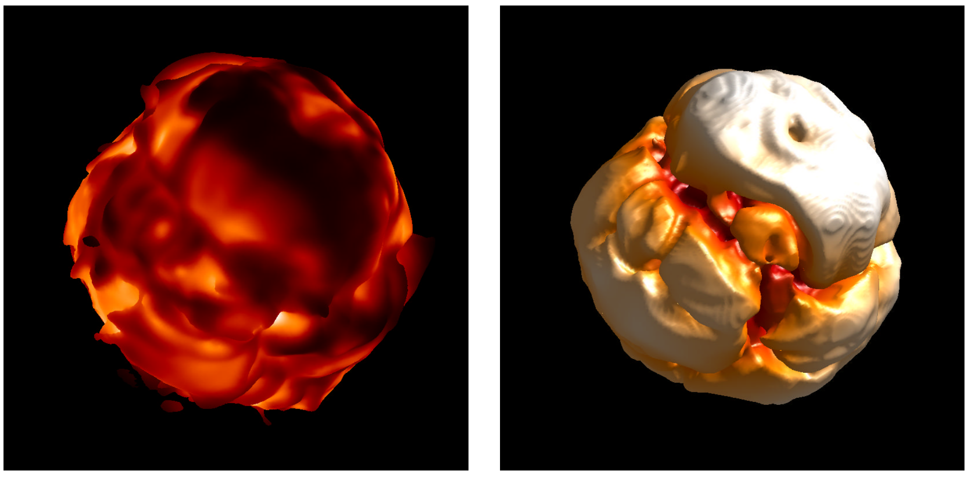


Figure 5.1: (left) Frame MEDRESII_14n3. The colors are T^4 (which corresponds roughly to the bolometric intensity) shown at optical depth unity. The image represents very roughly what an observer would see visually. Few giant convection cells are present but they are not clearly outlined with the intensity pattern. (right) Same frame as left except it shows an isosurface of entropy, the point of view is the same. Here the outlines of the convection cells are much clearer. Compare the outlines of the convection cells with the observed intensity pattern. All data above optical depth unity has been cut away.

What is a single convection cell?

One should question the very definition of a cell as a single, independent entity. This definition seems to be an oversimplification of reality. The models usually show that cells are closely connected structures. See figure 5.2 for an example. The figure shows isosurfaces of radial velocity inside the star (which has been cut in half). Radial velocity is a measure of updrafts and downdrafts which are used when definining convection cells. The top left and both bottom row images show updrafts, i.e. gas with radial velocities higher than 1 km/s flowing outward. The top-right image shows downdrafts with radial velocities higher than 1 km/s flowing inward.

The images do not show any obvious boundaries between different convection cells. The updrafts are connected with parts protruding up through the upper layers. Maybe it would be a better terminology to talk about the “convective structure”.

But how about the surface pattern of cells, is it at least possible to define a single convection cell in that? The surface pattern is formed by cells protruding up through the photosphere crossed by lines of downdrafts in between. The problem is similar here, cells are not preserved over time. In the models it is common to see a downdraft form and stretch like a finger into a cell, gradually cutting it apart.

This forms fragments which either live on or are suffocated by the surrounding downdrafts. Therefore cells are constantly splitting up making the pattern ever changing.

However, in the lack of a better understanding and an usable terminology I will stick with talking about cells as being clearly defined entities. Still, it should be understood in this light of cells being part of the immense convective structure which spans more or less the whole stellar envelope.

5.2 Optical depth in a 3D volume

I commonly talk about showing the models at optical depth unity. However it is not entirely clear what is meant since optical depth is defined along a specific line of sight. This concept was approximately generalized by Freytag in a practical way for a cartesian grid. He measures optical depths along each of the principal axes of the coordinate system. Each axis gives two values of optical depth, one for observers looking towards the negative axis and one for observers looking toward the positive axis. These values are weighted together in kind of a harmonic mean

$$\frac{1}{\tau(x,y,z)} = \sum_{i=1}^{6} \frac{1}{\tau_i(x,y,z)} \tag{5.1}$$

where $\tau(x,y,z)$ is the optical depth at gridpoint with location x, y, z. $\tau_i(x,y,z)$ are the measured optical depths along the principal axes. See figure 5.3 (left).

This equation weights τ towards lower τ_i. Very high τ_i will practically vanish. This is practical since we are primarily interested in the value of optical depth in the upper layers of the photosphere, i.e. where we would expect low values. In figure 5.3 (right) the gridpoint being measured is very close to the surface, here the optical depth through the nearest point on the surface (along the y-axis) is equal 2. The optical depths along the other axes will be very large, or for our purposes, practically infinite. Therefore equation 5.1 gives (here in 2D for simplicity)

$$\frac{1}{\tau} = \frac{1}{\tau_{y-}} + \frac{1}{\tau_{y+}} + \frac{1}{\tau_{x+}} + \frac{1}{\tau_{x-}} \approx \frac{1}{\tau_{y-}} + 0 + 0 + 0 = \frac{1}{2}$$

which gives $\tau = 2$. Therefore equation 5.1 will give small depths for points close to the surface which is as one would instinctively expect.

This equation can do strange things as producing lower τ than any one of the measured τ_i. This happens when the optical depth has low values alongside more than one axes, but the result are always within the same order of magnitude. This is usually not important since optical depth is not in it self an important quantity for the analysis done in this report. Instead it is used to get an approximation to the observed surface of the star, e.g. to get a hint of how large part of a convection cell

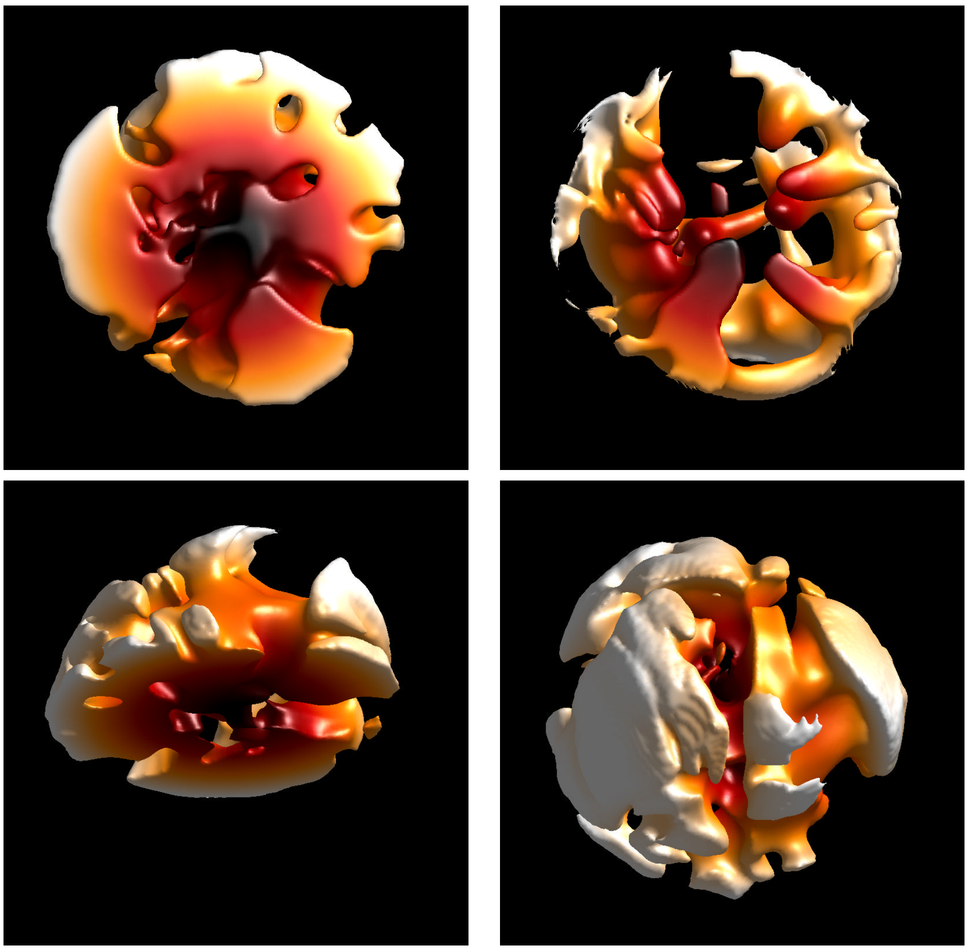


Figure 5.2: The image shows a slice through frame MEDRESI_24n3. The data shown is the radial velocity of the gas in the star. The star has been cut in half. Radius is used for coloring. Images at top-left and in the bottom row show gas which flows radially outward with a velocity of 1 km/s or greater. The image at top-right shows gas flowing radially downwards with velocities of 1 km/s or greater. The two bottom images are from different point of views in order to show the 3D structure of the updrafts. This hints that there is no clear boundary between convection cells, rather they must be seen as being part of the same huge convective structure.

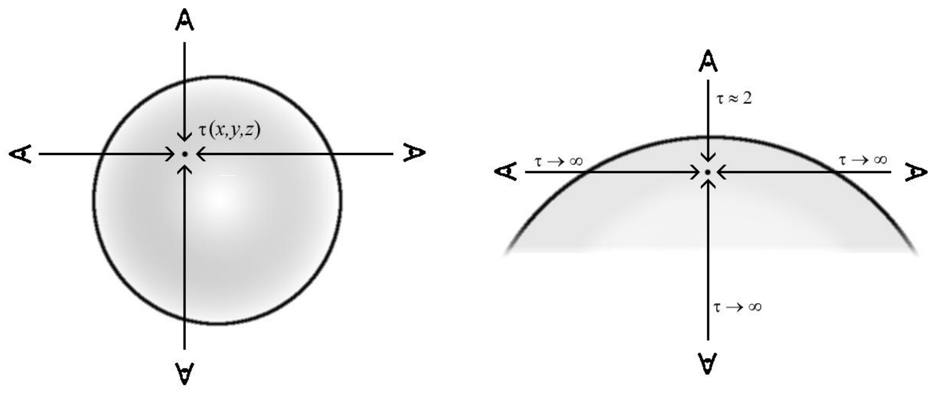


Figure 5.3: Calculating the optical depth for every gridpoint in the model, here only the 2D case is shown for simplicity. (Left) There are in total four axial (six in the 3D case) lines of sight towards a gridpoint at location x, y, z. Along each line of sight the optical depth is calculated, these optical depths are weighted using equation 5.1 to produce the optical depth $\tau(x, y, z)$. (Right) The special case when the gridpoint is close to the surface, $\tau(x, y, z)$ will be weighted towards the lowest value of optical depth.

would be visible, or to get an idea of the observed intensity pattern of the surface as already shown. However this definition should be used with care specially where $\tau \approx 1$ since there are quantities which change by large amount over this boundary.

For a justification of using equation 5.1 look at figure 5.4. The left image shows a 2D image where parallel light rays have been integrated directly in the simulator using the radiative transfer equation through the whole model, thus representing the bolometric intensity as a distant observer would see it. To produce the image at right a 3D volume was generated where each gridpoint was given optical depth according to equation 5.1. From that an isosurface at $\tau = 1$ was created and the bolometric intensity according to Stefan-Boltzmanns ($I \propto T^4$) law was colormapped on the surface.

These images are roughly similar. Still some areas show up different, specially near the edge of the disk where the image at right is generally brighter. This is due to limb darkening which shows up in the integrated image (left). The image at right is generated using equation 5.1 which does not assume any special point of view making it impossible to reproduce limb darkening. The fuzzy boundaries in the integrated image are due to the radiative transfer equation which correctly picks up light from gas in zones where $\tau < 1$. In addition there are some smaller differences in structures. But on the whole the images are sufficiently similar to validate equation 5.1.

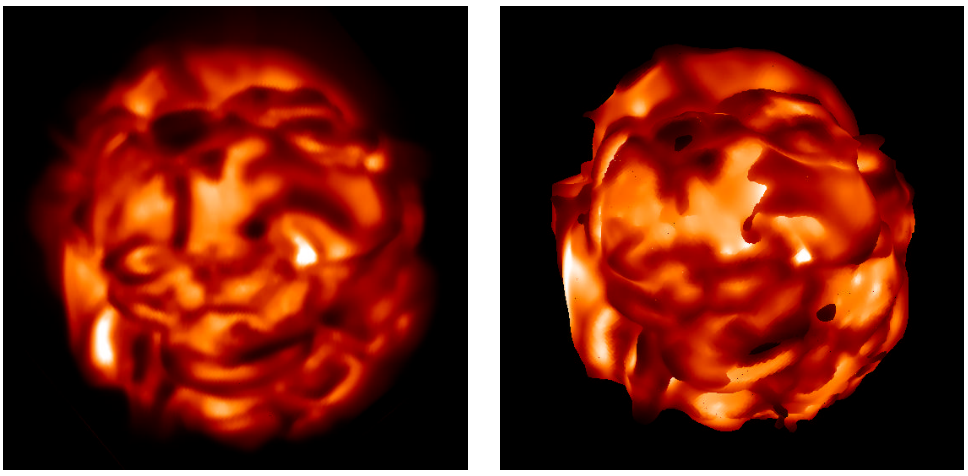


Figure 5.4: Comparison of two different ways to approximate the observed intensity. Here frame MEDRESII_14n3 is shown. (Left) The image is created by a module in the simulator which integrates parallel rays going through the model and would be valid for an observer placed very far away. Since the image is integrated using the radiative transfer equation it represents a more real view of the star. The produced image is only two dimensional. (Right) A 3D approximation of the surface at $\tau = 1$. A 3D array containing the optical depth at each gridpoint calculated according to equation 5.1 was used to create an isosurface at $\tau = 1$. The coloring is proportional to T^4 which is roughly proportional to the intensity. Both images show the same point of view. These two images are roughly similar, i.e. many main structures are similar in both. The left image is more fussy since the integration correctly picks up light from gas which has $\tau < 1$ causing edges to become very softened.

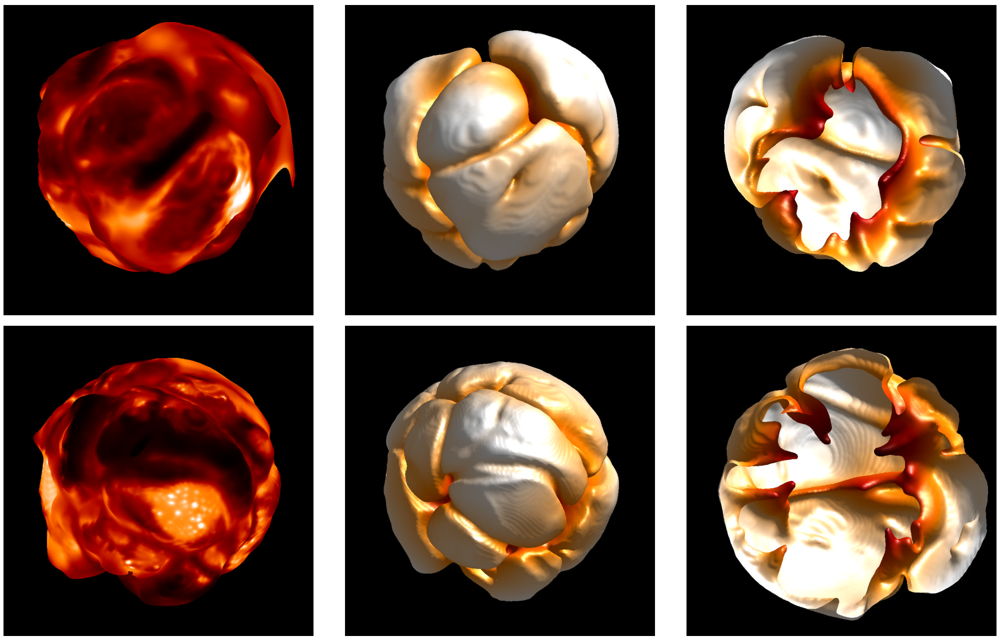


Figure 5.5: Comparison of models with different resolution. In the top row is frame LORESI_22n0 and the bottom row shows frame MEDRESI_24n3. The left column shows the intensity at $\tau = 1$, the middle column shows the convection cells using an isosurface of entropy, the right column is the same as the middle except it shows the cell structure from the "inside". When looking at the entropy images hints of smaller structures are seen in MEDRESI, although not very clearly. The downdrafts of MEDRESI are similar except they seem slightly more pointier. Coloring in the entropy images is the radius.

5.3 Changing resolution of the model

When doing models there is always the question of how well they reflect reality. One question is the importance of the basic numerical parameters of the model, e.g. resolution, algorithms, type of coordinate system etc. These are important since ideally these things should not change the results in a radical way. In this report some tools are developed to measure the importance of numerical resolution.

In figure 5.5 both LORESI and MEDRESI are shown. The leftmost columns shows the intensity at $\tau = 1$. The middle and right columns show the convection cell structure (using entropy isosurfaces), the middle column has the same point of view as the left column, but the column at right shows the convection cell structure from "inside". The right column is shown to point out differences in the downdrafts. The frames shown are LORESI_20n0 in the top row and MEDRESII_14n3 in the bottom row.

Chapter 6

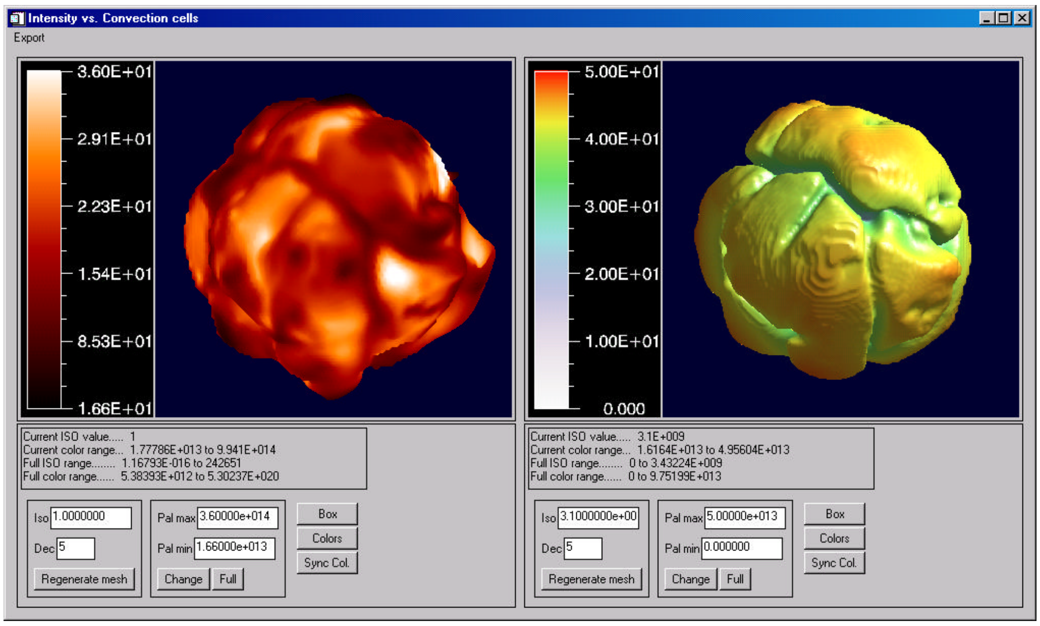
vis3D, the data visualizer

At first I needed some tool to help me get a feeling for the data. The institution uses the IDL development environment, however the tools coming with IDL are inadequate since they offer very limited ways of relating different datasets. Furthermore the tools are limited to 3D arrays of only 20^3 gridpoints while I needed to be able to visualise 3D arrays of 171^3 gridpoints and more. Therefore I decided to make my own 3D visualization tool in order to assist me in analysing the models. The program I made is called *vis3D* and is a general 3D array visualisation tool written in IDL. The oldest version of IDL it has been tested with is 5.4 but it probably works with versions down to 5.0 (since it uses object structures introduced into that version of IDL).

However, this program is still in alpha stage and needs some heavy polishing if I would for some reason decide on distributing it for general use. But for my needs it served its purpose well.

6.1 A typical vis3D session

Figure 6.1 shows a typical session in vis3D where few different quantities in frame LORESI_20n0 are shown together. The screenshot is from the Windows version of IDL 5.4. The left side of the dual-view is showing the intensity (i.e. T^4) mapped on an $\tau = 1$ isosurface, the right view is showing an entropy isosurface to visualise convection cells and the coloring is radius from the center. The data can be rotated around with the mouse and both views always show the same point of view thus making it easy to relate data in different 3D arrays. Lights are used in the right view to make it easy to see the 3D structure. In the left view the actual color is more important so no lights are used.



```
Current ISO value.....   1
Current color range...  1.77786E+013 to 9.941E+014
Full ISO range........  1.16793E-016 to 242651
Full color range......  5.38393E+012 to 5.30237E+020
```

Iso 1.0000000
Dec 5
Pal max 3.60000e+014
Pal min 1.66000e+013

```
Current ISO value.....   3.1E+009
Current color range...  1.6164E+013 to 4.95604E+013
Full ISO range........  0 to 3.43224E+009
Full color range......  0 to 9.75199E+013
```

Iso 3.1000000e+00
Dec 5
Pal max 5.00000e+013
Pal min 0.000000

Figure 6.1: A typical session in vis3D using the dual-view mode. The views have synchronized point of views, the mouse is used to rotate either one of the views and the other view follows. Note that in the right view lights are used to enhance 3D structures while in the left view color is more important so no lights are used.

6.2 List of features

6.2.1 Coloring Isosurfaces

A 3D view in vis3D is usually generated from a pair of (equally sized) 3D arrays. The 3D geometry is an isovalue of one 3D array while the colors are taken from a separate 3D array. The view can be rotated around with the mouse.

As of now only vertex coloring is supported. No texture maps are implemented which makes it hard to use complex colordata on a flat, single polygon surface. However, see the section about Tips and Tricks below for a way around this.

6.2.2 Synchronized Multiple Views

Using an idea by Freytag vis3D can visualize up to four different views simultaneously. Each view has its own individual combination of 3D arrays to use for isosurfaces and coloring. However, rotating the geometry in one view rotates all views synchronously. This ensures the same point of view is always used. This is very useful when comparing different quantities of the same frame, e.g. in figure 6.1 mentioned above the intensity pattern is compared with the convection cells.

6.2.3 Simplification of Geometry for Speed

The user can specify a decimation level for each view individually. The decimation level simplifies the geometry which increases speed when rotating with the mouse. Unfortunately this simplification also removes important details. To solve this vis3D uses the simplified version as the user rotates the model which makes it faster and as soon the user stops interacting with the model the program renders the 100% detailed version. This works quite well.

This technique was necessary since the complexity of e.g. a 3D array with 171^3 elements can be overwhelming for the 3D hardware. This made interaction tediously slow if one didn't want to loose details.

6.2.4 Exporting

It is possible to export individual views to several common 2D image formats (e.g. JPEG, PNG, TIFF). There is also a 3D VRML exporter included, e.g. making it possible to put the models on the web for others to view.

6.2.5 Choosing Colors and Color Mapping

Colors can be chosen for each view either via starting parameters or at runtime via a palette dialog. So far only the native color palettes in IDL are supported. The range of physical values mapped to the palette can be chosen for each view individually. If needed the chosen colors and palette values can be copied from one view to the other views.

6.2.6 Turning User-Interface Elements On or Off

The user interface can be simplified via input parameters. The elements possible to turn on and off are the colorbars, the input fields (buttons, input fields) including the informative text field and the menu bar. In essence the program can be forced to showe only the bare views.

6.2.7 Automatic Exporting

vis3D can export images without needing to show the user-interface. This can be useful in scripting when making e.g. animations. Via input parameters it is possible to specify rotation, coloring, isovalues, export format and more.

6.3 Using the user interface

See figure 6.2 for pointers to different elements of the UI.

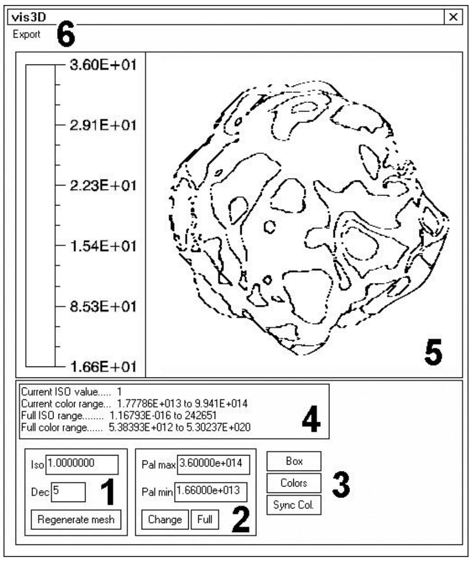
Controlling the 3D geometry (1)

Here the parameters of the 3D geometry are changed. In `Iso` a new isovalue is typed in for the 3D array and in `Dec` a new decimation (i.e. simplification of geometry) value is given where 100% is no simplification (use 5-20% for good speed/quality tradeoff). Press `Regenerate Mesh` to generate new 3D geometry after choosing the new parameters.

The simplified version of the geometry is only used when interacting with the 3D view using the mouse. A full detailed 100% version is always displayed after user interaction.

Controlling color range (2)

This controls the range of physical values which are mapped to the color palette. Write new mininum/maximum values in the fields and press `Change` to remap the



```
Current ISO value.....  1
Current color range...  1.77786E+013 to 9.941E+014
Full ISO range........  1.16793E-016 to 242651
Full color range......  5.38393E+012 to 5.30237E+020
```

Iso: 1.0000000
Dec: 5
Pal max: 3.60000e+014
Pal min: 1.66000e+013

Figure 6.2: The main elements in the user interface of vis3D. Only one view is shown. The numbers refer to corresponding paragraphs in section 6.3.

colors. Press `Full` to make the palette automatically span the full range of physical values the isosurface spans in the space of the color 3D array.

Note that pressing the button `Sync Col.` will copy the current mapping to all other views alongside with the color palette in use.

Displaying a boundary box (3)

The button `Box` displays a wireframe 3D box. The box marks the boundary of the current iso 3D array.

Changing the color palette (3)

The button `Colors` brings up a dialog where a new color palette can be chosen. The possible color palettes are inbuilt in the IDL package. Some basic operations such as inverting and changing scales of the palettes are possible.

Synchronize colors between views (3)

The button `Sync Col.` copies the current color palette of the view to the other views. In addition the palette mapping is copied as well.

Information about the current 3D arrays (4)

The informative text field shows some basic data about the current 3D arrays. The term `ISO` refers to the 3D array used for generating the 3D isosurface also referred to as the *iso 3D array*, and `color` refers to the 3D array used for coloring, or the *color 3D array.*

Current ISO value
The isovalue used to generate the surface from the iso 3D array.

Current Color range
The range of values which the iso-surface spans in the space of the color 3D array.

Full ISO range
The full range of values in the iso 3D array.

Full color range
The full range of values in the color 3D array.

The 3D view and colorbar (5)

The mouse can be used to drag the 3D objects around in the views with the left mouse button. Optionally a colorbar is displayed within the view. The colorbar needs polishing, it does not round off the numbers as it should and if the numbers are too long it can happen that only asterixes are displayed instead of numbers.

Export menu (6)

The export menu has the following three options

2D image format
Exports to several common 2D image formats. The available formats are depending on the version of IDL, in IDL 5.4 the supported formats are JPEG, TIFF, PNG, PPM, SRM and BMP. There is a bug in the TIFF exporter which makes the image upside-down.

VRML
Exports any view to 3D VRML format.

Print Rotation matrix
Prints the current transformation matrix to the IDL command line of the views. The matrix is printed in a valid IDL format and can be copy/pasted directly into scripts. This rotation matrix can be used as input to the parameter `ROTATIONMATRIX` to duplicate the current rotation of the view for

another session. This is useful in scripting to repeatedly display some interesting parts of the data. Note that the rotation matrix does work for all 3D arrays having exactly the same dimensions. Therefore it is possible to find an interesting place in e.g. the temperature data, get the rotation matrix and use it on the corresponding entropy data. The entropy data will be rotated to the same view.

However, using differently dimensioned array with the same rotation matrix moves the origo off center and everything will simply be wrong.

6.4 Input parameters

In order to set a keyword parameter in IDL use "/" in front of it, e.g. `/SMOOTH_ON` turns on smoothing.

6.4.1 Passing 3D arrays

Start vis3D with the pairs of 3D arrays as the first parameters:

```
vis3d, iso1, col1, [iso2, col2,] [iso3, col3,] [iso4, col4]
```

Make it able to use up to four pairs of 3D arrays, one for each view. As of now it takes 3D arrays in only by reference (due to change).

The iso 3D arrays are `iso#` and color 3D arrays are `col#`.

As of now a pair of 3D arrays must be given for each view. It is mandatory to give at least one pair.

6.4.2 Displaying the 3D data

`ISOVALUES`
: An array of isovalues. Has one element corresponding to each view. An iso-value must be given for each view.

`DECIMATION`
: An array of decimation values in percentages, one element for each view. Use integer values in the range 1..100. If not given the default value 5% is used for all views.

`IDLPAL`
: Array of IDL index numbers to predefined palettes, one element for each view. Palette number one is default (black to white gradient).

`PALETTERANGE`
: Gives the range of values mapped to the palette in each view. This is an

array of the format [[min1,max1], [min2,max2], [min3,max3], [min4,max4]] where min#, max# is the palette range for view #. One min#,max# pair must be given for each view.

`SCALE`
: Scaling of the displayed 3D objects. Only positive floats are allowed, bigger numbers scale up the 3D object. If set equal to 1.0 the 3D object is scaled exactly so it will never cross the edge of the view, even when rotated. All views share the same value for scaling.

`SHOWAXIS`
: If set the boundary box is displayed in all views.

`ROTATEANGLE`
: The initial angle of the view. It gives the angle of rotation around the axis given in `ROTATEAXIS`.

`ROTATEAXIS`
: The axis of rotation, given as a 3-element vector in the format [x,y,z]. The angle of rotation is given in `ROTATEANGLE`.

`ROTATIONMATRIX`
: Set equal to a full 4x4 transformation matrix. A transformation matrix has information about both angle and translation of the 3D object in 3D space. Is meant to be used in conjunction with the export menu command “Print Rotation Matrix”.

`LIGHTS`
: Controls lighting in each view. Set this to an array with one element for each view. Elements which are equal 0 turn off lighting, 1 turns on lighting.

`VIEWCOLOR`
: The background color of the views. A three-element array of the form [R, G, B]. Same color is used for all views.

6.4.3 Smoothing parameters

These parameters control the smoothing of both the 3D geometry and the color values. Use with much care since it can radically destroy important details.

`SMOOTH_ON`
: If set it turns on smoothing. Default values will be used for the keywords `SMOOTH_ITERATIONS` and `SMOOTH_LAMBDA`. Note that only if `SMOOTH_ON` is set then smoothing is performed even if the `SMOOTH_*` keywords are given.

`SMOOTH_ITERATIONS`
: Number of iterations for the IDL **`mesh_smooth`** function used to smooth the geometry. See the IDL manual for further information.

`SMOOTH_LAMBDA`
: Lambda number for the IDL **`mesh_smooth`** function. See the IDL manual for further information.

`COLORSMOOTH_ON`
: If set it turns on smoothing of color values. The colorsmoothing uses the IDL command **`mesh_smooth`** to smooth the color values.

`COLORSMOOTH_ITERATIONS`
: The iterations of the IDL command **`mesh_smooth`**. See the IDL manual for further information.

`COLORSMOOTH_LAMBDA`
: The lambda value for the IDL command **`mesh_smooth`**. See the IDL manual for further information.

6.4.4 Controlling the user interface

`XSIZE, YSIZE`
: These keyword parameters control the width and height of the views in display pixels.

`TITLE`
: A string containing the title of the vis3D main window.

`SIMPLE_GUI`
: A macro which sets the parameters `NO_CONTROLS`, `NO_MENU` and `NO_COLORBAR`.

`NO_CONTROLS`
: If set then no controls are visible (buttons, textfields etc.).

`NO_MENU`
: If set the menu is not shown.

`NO_COLORBAR`
: If set the colorbars are not shown.

`QUIET`

`VERBOSE`
: These keyword turn off resp. on informational messages by the program when running.

These keywords work by setting or clearing the system variable `!QUIET`. Therefore, vis3D will react to the value of `!QUIET` even if `QUIET` or `VERBOSE` were not mentioned at all in the parameter list. Use this with care since it affects other applications as well.

6.4.5 Exporting images (e.g. for script controlling)

The Export mode is a separate mode in vis3D. It is activated using the keyword `EXPORT_ON`. In this mode no user interface is displayed at all, instead vis3D creates the 3D images in the background, exports to disk and exits. This mode is primarily thought for generating images for e.g. animations using scripting.

All keywords having the prefix `EXPORT` are only used in export mode. If `EXPORT_ON` is not set these keywords are simply ignored.

It is possible to use many views in Export mode, in that case all views are combined together horizontally into one image file, including colorbars if requested.

`EXPORT_ON`
: Turns the Export mode on. The following keywords are *mandatory* when in Export mode:

    `EXPORT_FILENAME`

    `EXPORT_TYPE`

    `EXPORT_DIMENSIONS`

`EXPORT_FILENAME`
: The filename of the exported image *excluding filename extension.* The extension will be automatically added by the program.

`EXPORT_FILENR`
: A integer which will be postfixed to the filename. This is e.g. used for indexing frames in an animation.

`EXPORT_PATH`
: Path for the exported image file. Defaults to current directory. Can be both a relative or absolute path.

`EXPORT_TYPE`
: A string containing the type of the exported image. Currently supported types are 'BMP', 'JPEG', 'PNG' and 'TIFF'. This string will also be used as a suffix for the filename.

  Give the string 'nofile' to suppress writing the image file to disk. This is used in conjunction with the keyword `EXPORT_GETIMAGE`.

`EXPORT_DIMENSIONS`
: A 2-element array of the form [xsize, ysize] giving the dimensions of the exported image. The maximum allowed dimensions are [1600, 1200].

`EXPORT_GETIMAGE`
: Set this equal to a variable in which to copy the generated image. The image will be stored as an array with dimensions [3,xsize,ysize], the array is of type BYTE. There are three color planes (for each elemental color: red, green, blue). The first dimension indexes which color plane to access. The second and third dimensions store the 2D image itself. This format is standard for many image manipulating commands in IDL, see the IDL manual for further information.

  The keyword `EXPORT_GETIMAGE` was included to make generating mpeg animations easier using the already inbuilt mpeg-modules in IDL. The image format of the array is used by these modules.

  Set `EXPORT_TYPE` in order to suppress writing of the image file to disk.

6.5 Tips and tricks

vis3D is just a simple 3D array viewer. To accomplish some tasks we prepared the 3D arrays in different ways.

6.5.1 Texture mapping the sides of a box

Alongside creating the 3D arrays CO^{5}BOLD also creates 2D-image files. These image files are views of the 3D arrays from three different directions, i.e. along the principal axes of the coordinate system. I mentioned these images previously in section 5.2. These images are created by integrating parallel rays going through each gridpoint in the side of the box volume. We needed a simple way to compare these 2D images to the 3D arrays preferably within vis3D. But vis3D does not support texture mapping, only vertex coloring. Therefore a 2D image does not map too well on a flat plane since there are not enough vertices to hold the color values of the pixels.

To solve this Freytag made a small program which caused extra vertices to be created on the plane in order to map a 2D image on it. Here follows a simplified explanation of the procedure:

Start with a 3D array containing only horizontal layers. Each horizontal layer is homogenous and has a higher value than the layer below, lets assume the layers have the values 0,1,2,3,4… etc. The isosurfaces of this 3D array will of course be flat planes. When IDL creates isosurfaces it interpolates between neighbouring gridpoints in the 3D array, e.g. in our case if we want an isosurface with the isovalue 3.4 then IDL will interpolate between layers having the values 3 and 4 and the isosurface will be located between those two.

This can be used to texturemap images on planes. By perturbing the values of some of the horizontal layers in the 3D array it can be controlled where the interpolation will place the vertices. Therefore the pixel values of the image are used to perturb the gridpoints in some horizontal layers. When an isosurface is created its vertices will be displaced from the original plane with the displacement depending on the pixel values. This curves the plane where the pixel values change causing vertices to be created. To actually color these vertices the 2D image is used again as a color map. Usually the 2D image is copied to another, equally sized, 3D array. This 3D array is both used to perturb the original layered 3D array and as the color map.

This is the basic idea. However, we had six images instead of one, each corresponding to one side of the cubical volume. Two 3D arrays were prepared. The first 3D array has parallel, homogenous, layers with different values on each side. (corresponding to the horizontally layered 3D array above), creating an isosurface from this array gave a cube with flat sides. The other 3D array had the 2D images copied to each side. This 3D array was used both to perturb the other 3D array and for colormapping the the isosurface afterwards. This created nice results.

An example of the output is shown in figure 6.3.

6.5.2 Masking arrays

In chapter 7 I will describe how masking is used in the CCS algorithm. Masking prooved to be very useful method to cut and slice through the model as needed. It can be used to hollow out a shell from the 3D arrays or cut away extra layers (common use was to cut everything above $\tau = 1$), or cut the star in half to see the surface in relation to the inner processes.

To do this one creates a 3D mask in the same way as the masks in the CCS algorithm, using the value 1 for gridpoints belonging to the mask and 0 for those outside of it. This mask is multiplied with the 3D arrays to cut unwanted pieces from them. For avoiding errors it is a good rule to first cast the mask to the number type of the 3D array (i.e. double, long, int...).

Note that the color 3D arrays should *not* be cut using the masks since it will cause errors in the cut out boundaries.

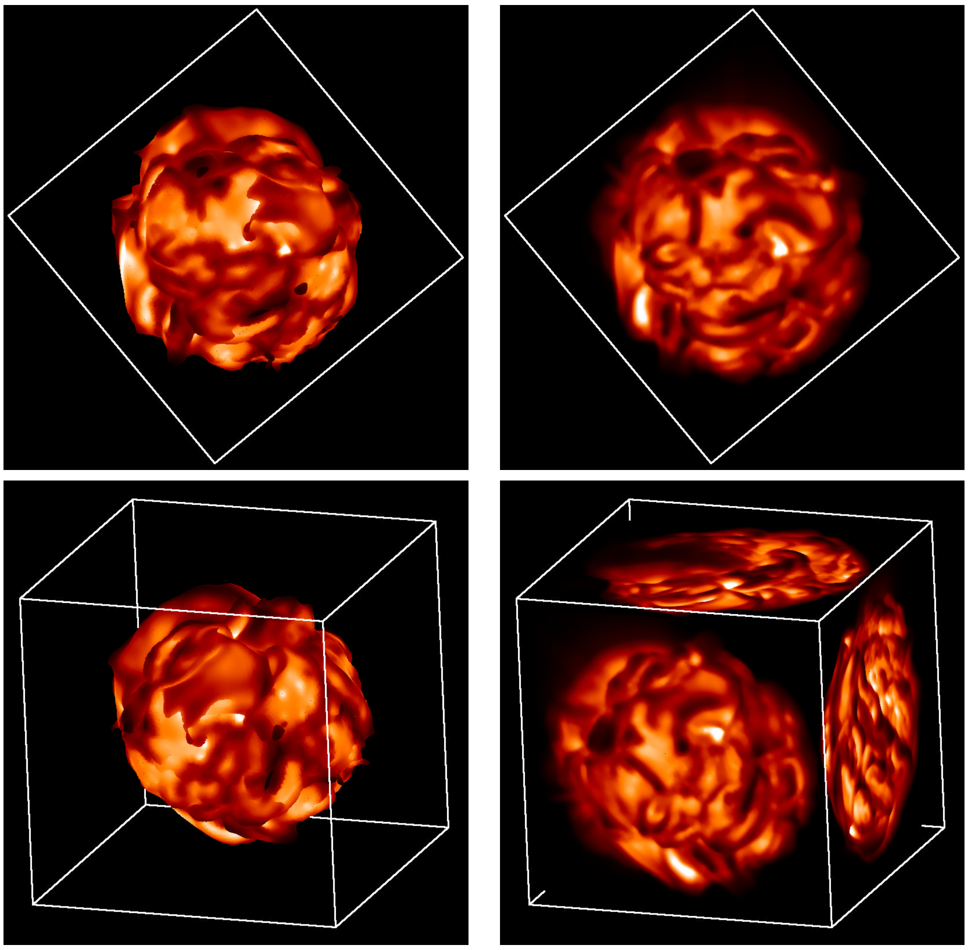


Figure 6.3: An example of texture mapping using specially prepared 3D arrays in vis3D. The top row shows one side of the mapped rectangle compared with the corresponding 3D model. The bottom row displays the view slightly rotated showing how each side of the 3D array has been mapped with the 2D image corresponding to that particular point of view.

Chapter 7

Measuring convection cells

All the models in chapter 5 showed huge convection cells. However, it would be interesting to get some quantitative data about e.g. the scales of the cells. Questions as how many are the big cells and how large? Are all cells similar, i.e. are they close to some mean size or are we dealing with cells occuring on few different spatial scales? This last question is interesting since past papers often implicitly assume the granules to have similar sizes, see e.g. Gray (2000, 2001). Another interesting question for observers concerns e.g. the evolution and timescales of the spatial distribution of granules.

Having a way to quantify these questions strengthens the link between modelmakers and observers. It is easier to use the models to explain observations and using observations to improve the models.

In this chapter I go through a method which begins to tackle those things. The main questions here are: What is the distribution of spatial scales? How many cells can be expected to be on the surface at any given time?

I used this method to analyse both the LORESI and MEDRESI models in order to get a hint of the change in spatial scales when increasing the numerical resolution. It was of particular interest to see if the large scale structures survived when going to a higher resolution or if they would disappear in favour of smaller scale structures.

7.1 Tools to measure convection cells

First we need to find relevant quantities and create tools to measure them. Using these tools it is possible to get quantitative measurements which can be compared and interpreted.

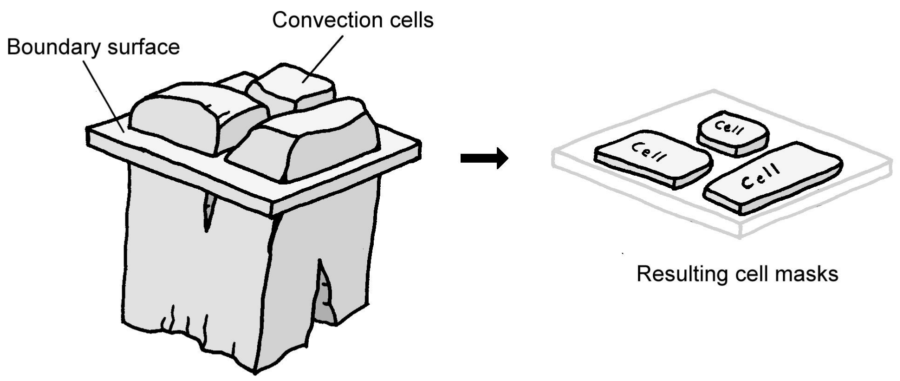


Figure 7.1: The process of masking cells. The cells (left) protrude through the boundary surface. Some suitable quantity is used to mask the insides of the cells *on* the boundary surface (shown at right). This image only shows the basic idea of using two masks to find the convection cells on a surface, the actual look of the masks can be quite different.

Measure the "total" volume of a convection cell?

Here we are interested in the size of the convection cells. Mostly we are interested in finding an indicator of the size of each cell. The original idea was to count the convection cells and measure their volumes. But as discussed in section 5.1.1 it is hard to define a whole convection cell so we had to find another way of measuring them.

7.2 The Cell-Cross-Section (CCS) algorithm

Instead of finding volumes we measure the area the cells have when crossing some defined boundary surface. The boundary has to be chosen so that relevant physical quantities change predictably on it. Here we were interested in areas which are "inside" and "outside" of a convection cell. We searched for a boundary surface and a corresponding quantity which could be used to mask which areas of the surface are inside the cells, see figure 7.1.

7.2.1 Choosing a suitable boundary

The boundary surface is just an isosurface of value a_1 of some chosen quantity q_1. This isosurface has to be chosen so that another quantity q_2, can be used to indicate where the cells protrude through it. On every point on the isosurface the value of q_2 is checked. A treshold value a_2 is used which marks the boundary between inside and outside a convection cell, e.g. if $q_2 > a_2$ it implies outside of a cell, otherwise it is inside.

So, the first step is to find a good pair of quantities, q_1 and q_2, and some suitable treshold values a_1 and a_2.

Using $\tau = 1$ as the isosurface

The obvious choise for the boundary surface is to use the isosurface $\tau = 1$. This would correspond to the observable area of the convection cells. A good matching quantity was the entropy s with the treshold value 3.1×10^9. Entropy marks the boundary between downdrafts and updrafts fairly well in the upper photosphere because the updrafts contain high temperatures and high entropy while the downdrafts are gas which has cooled down and lost entropy. This made a very sharp jump in entropy values over the boundary between an updraft and a downdraft, which is precisely what defines a convection cell.

There is a problem, at optical depth unity entropy changes very sharply. The entropy decreased quickly above the surface $\tau = 1$ which made the algorithm sensitive to inaccuracies due to limited resolution in the model and errors in the calculation of the $\tau = 1$ surface. A common problem was caused by the surface "bulging" slightly too high up, protruding into layers with lower entropy and causing the bulge to be incorrectly left out of the cell mask. This created holes and noise in the cell masks, see figure 7.2.

Using temperature and pressure

To find an alternative we instead used an isosurface $T = 12000\,\text{K}$ in combination with pressure. The best combination was usually to define the inside of a cell on the surface where $\log_{10}(P) < 3.34$. See figure 7.3 which shows that pressure tends to be higher in the downdraft regions on a given T-isosurface.

Using such a high temperature means the boundary surface is somewhat below $\tau = 1$. At this depth the relevant quantities changed less rapidly between gridpoints. This made it easier to find a good treshold value on pressure, which also lessened the problem of unwanted holes because of fluctuations.

7.3 Implementing the CCS algorithm

Here I will go briefly through the method of creating the boundary surface masks and corresponding cell mask. The individual steps are:

1. Create a boundary surface mask.

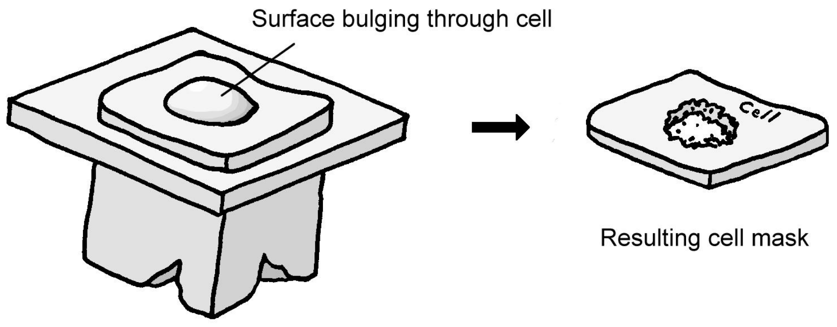


Figure 7.2: The problem of a protruding boundary surface. The convection cells are created by using entropy. In places the boundary surface can "bulge" up through the convection cells as shown. This results in a hole in the cell mask. Furthermore, due to the limited numerical resolution of the model the hole is usually not clearly defined, gridpoints are changing between being "inside" or "outside" the cell causing noise of isolated, or clustered, gridpoints inside the hole. Each such isolated gridpoint will be measured as an individual cell.

2. Find the cell mask.
3. Combine results from step 1 and 2 to mask cells on the boundary surface
4. Measure the area of the masked cells, count them and register.

To make the explanations little less abstract I will use temperature and pressure as described above. However, the explanations are equally valid for any other suitable quantities.

7.3.1 The structure of the model data

Here is a short list of terms necessary to understand the the explanations below. The important terms are *model*, *frame*, *3D array*, *mask* and *gridpoint*.

A *gridpoint* is an individual element in a 3D array.

A single *3D array* contains the complete state of a single physical quantity in a frame.

A *frame* has the complete state of all physical quantities in the model at a specific point in time. Each physical quantity is stored as a 3D array.

A *model* contains all the frames.

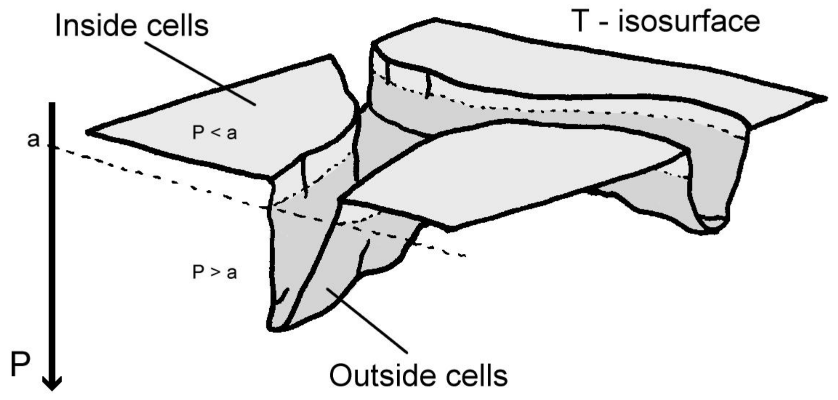


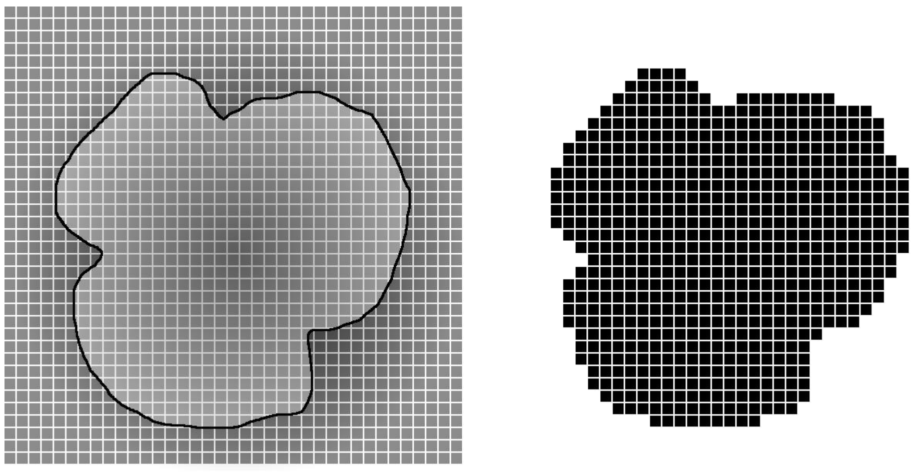
Figure 7.3: Using pressure to mark regions on a temperature isosurface. Ideally the pressure is only dependent on depth. Downdraft regions are cooler than updrafts at same depth which causes the isosurface of temperature to have "valleys" in the downdrafts. Therefore, when following the isosurface the pressure will be higher inside downdrafts than in the convection cells. This difference is used to cut away the downdraft regions from the isosurface. The figure shows how the isosurface is cut below a constant pressure a.

A *mask* is a 3D array where each gridpoint has either the value *true* or *false*, gridpoints having true values are said to belong to the mask. Masks can be combined together into new masks using boolean operations (e.g. AND, NOT, OR...). Boolean operations compare corresponding gridpoints in the masks to create new masks.

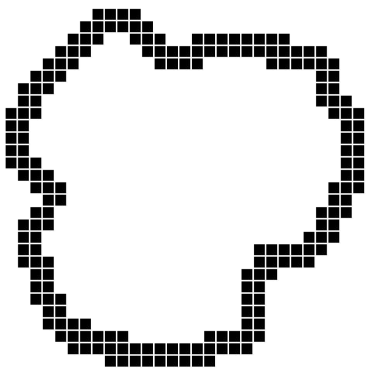
7.3.2 Create the boundary surface

The boundary surface is a thin shell around the temperature isovalue. The thin shell is created as a mask.

The first step is to use the isovalue to create a mask from the original temperature 3D array, the mask will contains all gridpoints which are both on and inside the isosurface. The left image below shows the original 3D array containing the temperature values (the images are shown as 2D slices for simplicity). The rough outline of the isovalue for the boundary surface is marked on the image. The right image shows the mask as created from the left image.

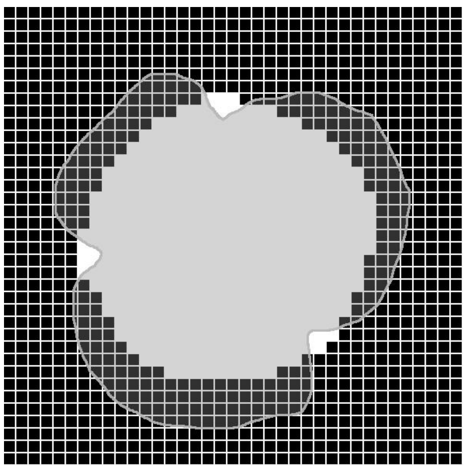


The next step is to apply some filtering algorithm which creates a shell out of the mask. It creates a shell only few gridpoints thick around the boundary of the mask, see the image below.



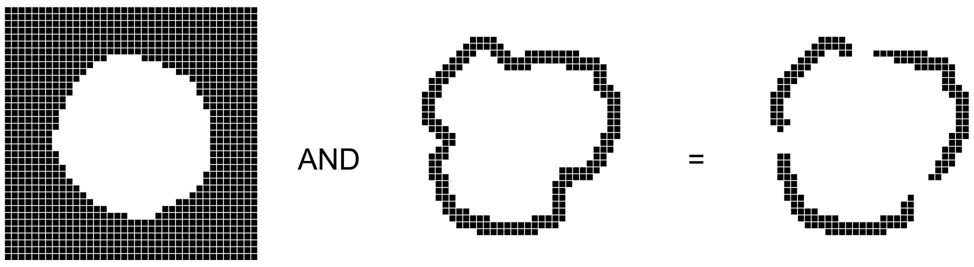
7.3.3 Filter the convection cells

A similar process is used to find the convection cells. A mask is created from the pressure 3D array where $\log_{10}(P) \leq 3.34$. This results, for the pressure, in a fairly smooth mask which cuts through cell tops on the temperature isosurface. The pressure-mask is shown with the temperature isovalue overlaid for comparison. Note how the pressure-mask only masks the cells while the downdrafts are mostly below the mask.



7.3.4 Masking the cells in the boundary surface

The two masks created above are combined in order to mask the cells on the the boundary surface. This is done with a boolean AND operation. This results in a mask containing isolated regions corresponding to the cells on the boundary surface.



In practice some extra smoothing is often needed to minimize noise in the mask.

7.3.5 Measure the area of the marked regions

The number of gridpoints in each marked region is measured. With luck the steps so far should have succeeded in keeping the marked regions fairly isolated from each other. A single region is simply a region of connected gridpoints in the mask. An algorithm is applied which goes through the whole mask, finds every isolated region and counts the number of gridpoints it contains. The output is a list of the counted gridpoints in every found region.

The area of a region is taken as being proportional to the number of gridpoints it has. This is justified by creating the boundary surface shell with (roughly) equal thickness all over so the volume of each region will be proportional to the surface area.

7.3.6 Getting a model-independent measure of area

The last remaining problem is comparing these results between models with different resolutions or even different spatial scales? The measured area is the number of gridpoints, this means it will get higher in models with higher resolution. We need some sort of normalised values which are comparable between models. We do this by normalising the number of gridpoints in each cell against the total number of gridpoints in the mask of the boundary surface shell. This gives the area of the cell as a fraction of the total surface area. An added bonus is that we should get similar areas even if we choose different quantities for defining the boundary surface. This could result in the boundary surface lying at different depths, still the normalised areas of the cells are likely to be similar.

These values are easy to compare between models.

7.3.7 Implementing CCS in IDL

Appendix A has listed the IDL source code for creating the cell masks and also the code for counting the number of gridpoints in every isolated region of a mask.

Parameters used by the IDL routines

For reference I list here the complete set of control parameters used for the IDL procedure `get_cellmask5` (listed in appendix A) when producing the results of this paper. For both models the input parameters were

- `TMIN` = 12000
- `PMAX` = 3.34
- `MINVOX` = 62
- `NEIGHBOURHOOD` = 5
- `SHELLTHICKNESS` = 9

7.4 Results of the CCS algorithm, the different spatial scales

The CCS algorithm is applied on every frame in the models. The raw output from the CCS algorithm is shown in figure 7.4 for model LORESI and in 7.5 for MEDRESI. Time is on the horizontal axis. The vertical axis is the normalised area (i.e. as a fraction of total surface area) of the measured cells, area of 1.0 corresponds to the whole surface. Each point stands for a single measured cell. The vertical axis is logarithmic (base 10).

7.4.1 Different types of cells

In the figures from the CCS algorithm it is possible, without too much imagination, to spot three distinct behaviours of the cells roughly depending on their size. At the top we have the biggest cells which form relatively stable tracks. These cells can occasionally split up which is seen as a branching of the tracks but they seem to always keep above a certain size the whole time. However, near the middle there are cells which are shorter lived. Their tracks suddenly appear and then "fall" down in size and disappear, showing a characteristic falling curve. When looking for these cells in the models I found them form through fragmentation of the big cells. A small part of a big cell was cut from it by a growing downdraft and the newly formed fragment quickly shrunk in size until it disappeared.

The third group in the figures are the individual cells which do not seem to have any obvious relation to any other cells. The smallest of those which are collecting near the bottom are probably mostly numerical noise caused by the imperfect filter algorithm and/or the choice of physical quantities for the boundary surface and cell marking. However, bigger individual cells can form when two regions connect

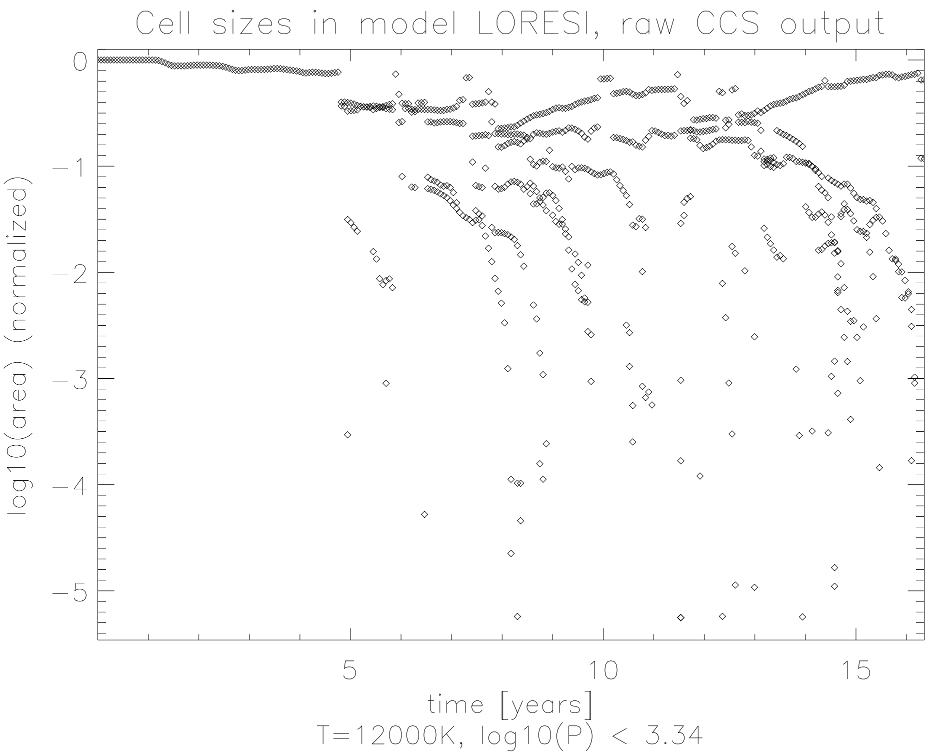


Figure 7.4: The raw output from the CCS algorithm for model LORESI. The model time is shown on the horizontal axis and normalised area is on the vertical axis (base 10 logarithm). Each point stands for a single isolated region or cell on the surface. Note how the cells seem to be roughly grouped in three distinct groups. The biggest cells are long lived and form relatively stable tracks at the top. Medium sized cells are seen as "falling" tracks near the middle, they appear and quickly die. The smallest cells are seen as individual points near the bottom, some are noise due to the discrete nature of the filtering algorithm which temporarily creates islands and then rejoins them.

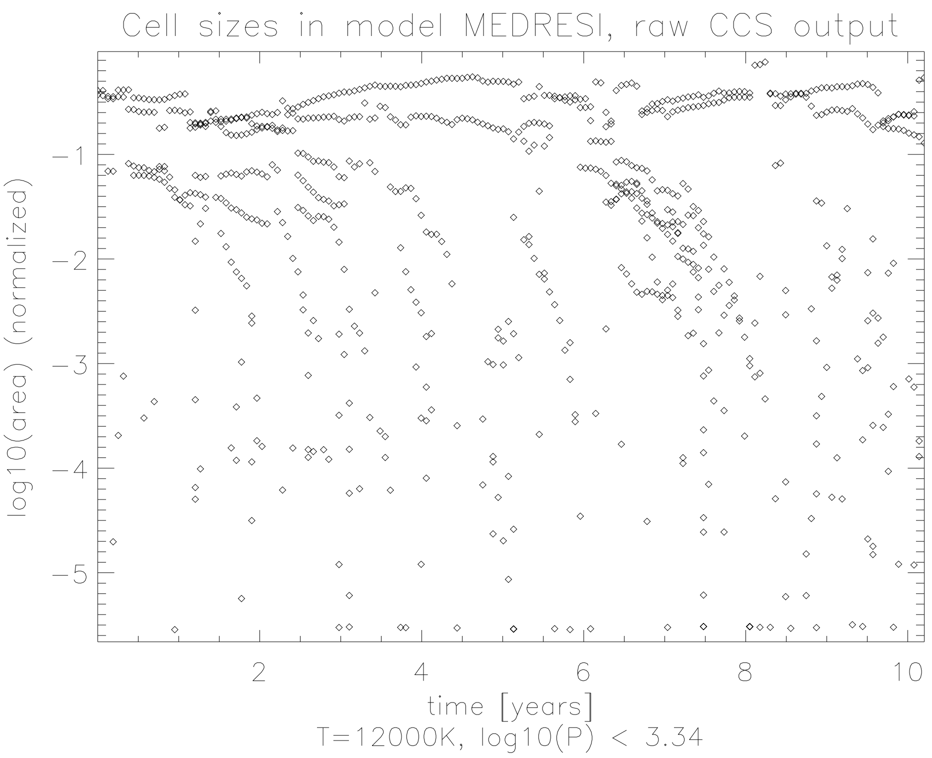


Figure 7.5: The raw output from the CCS algorithm for model MEDRESI. Model time is shown on the horizontal axis and normalised area is on the vertical axis (10-logarithmic). See description under figure 7.4.

with a temporary bridge in the mask. These bridges appear due to the discrete nature of the filter algorithm. During the masking step the gridpoints are defined to be either inside or outside a cell, there is no gradual change allowed. In places it can happen that some parts of a cell are very loosely connected together, i.e. in places the physical data is on the border of being defined as being ”outside” the cell rather than inside. Due to fluctuations the region can suddenly get pushed above the limit and be defined as ”outside” over 1-2 frames. This causes the small part to loose connection to the big cell and be temporarily counted as a separate small cell. These show up as smaller, isolated cells in the results.

However, some of these individual cells are actually part of longer tracks, e.g. of the short lived “falling” cells mentioned above. Since the frames of the model are not continuous but rather discrete snapshots in time then cells changing by orders of magnitude in size between frames show up as separate entities without any obvious connection to previous tracks in the plots.

7.4.2 The spectrum of spatial scales

Figures 7.4 and 7.5 look interesting enough but they are complicated to compare and really say too much at once. So the next step in the simplification is to create a plot showing the spectrum of the different spatial scales of the cells. See figure 7.6 for spectra of spatial scales in both models. The figure shows the relative frequency of cells of different scales. The figures are created using binning, the area is splitted into number of intervals (here 15) intervals (i.e. bins) and the number of cells within each interval is counted, the results are presented as the mean number of cells in each bin.

The spectra show what was already seen in the previous figures that the bigger cells are more dominant in the models.

There is a problem with representing the data in a binned form. Using different number of bins can produce quite different spatial scale-spectra, e.g. using more bins leads to fewer cells in each bin on all scales. This is a little arbitrary and to avoid the dependence of the number of bins we integrate the spectra.

7.4.3 The integrated spectra of spatial scales

Figure 7.7 is similar to the previously shown spectra except that each point shows the number of cells which are larger than or equal to that point, this is therefore an integrated version of figure 7.6 where the integration is from big cells to smaller.

The graph shows that both models produce roughly the same number of cells larger than 10% (however the exact proportions differ). But when going to smaller scales the models diverge, MEDRESI clearly increases its production of cells relative LORESI. In other words the high resolution model produces relatively more cells

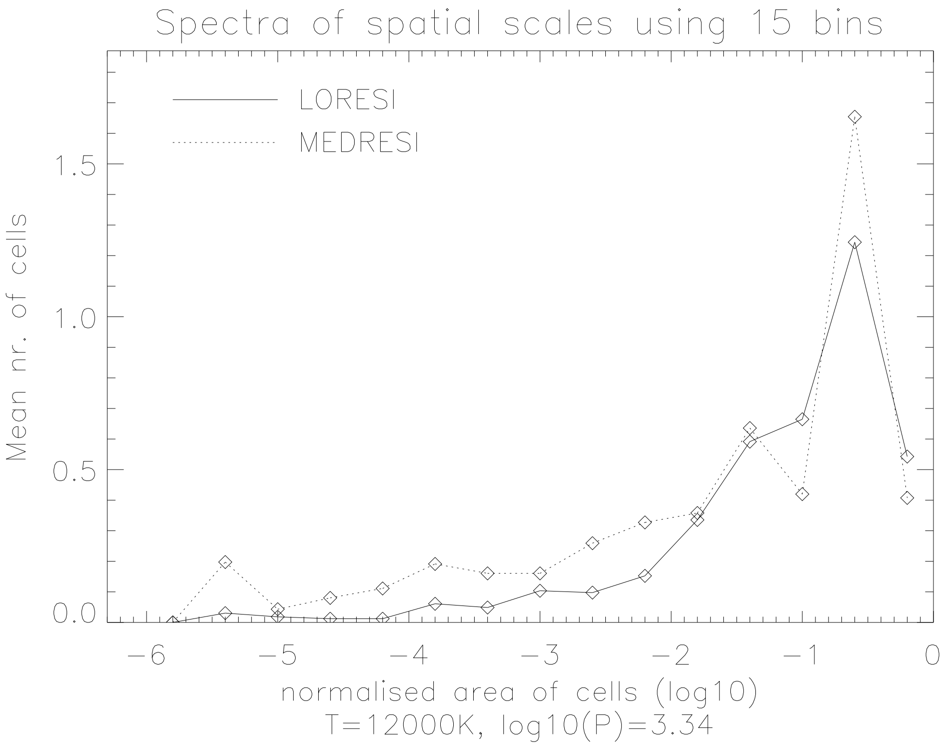


Figure 7.6: The spectra of spatial scales for cells in both LORESI and MEDRESI. Solid line is LORESI and dotted is MEDRESI. The 10-logarithm of cell sizes is on the horizontal axis and the mean number of cells is on the vertical axis. The spectrum is created by splitting the size-intervals into 15 bins and counting the cells within each bin. The measurement points are centered on each bin and the spacing between bins is equal on the logarithmic scale. MEDRESI produces more fragments on almost all scales. But the general behaviour is similar, in particular we have ever present big cells in both models (i.e. of sizes larger than roughly 10%)

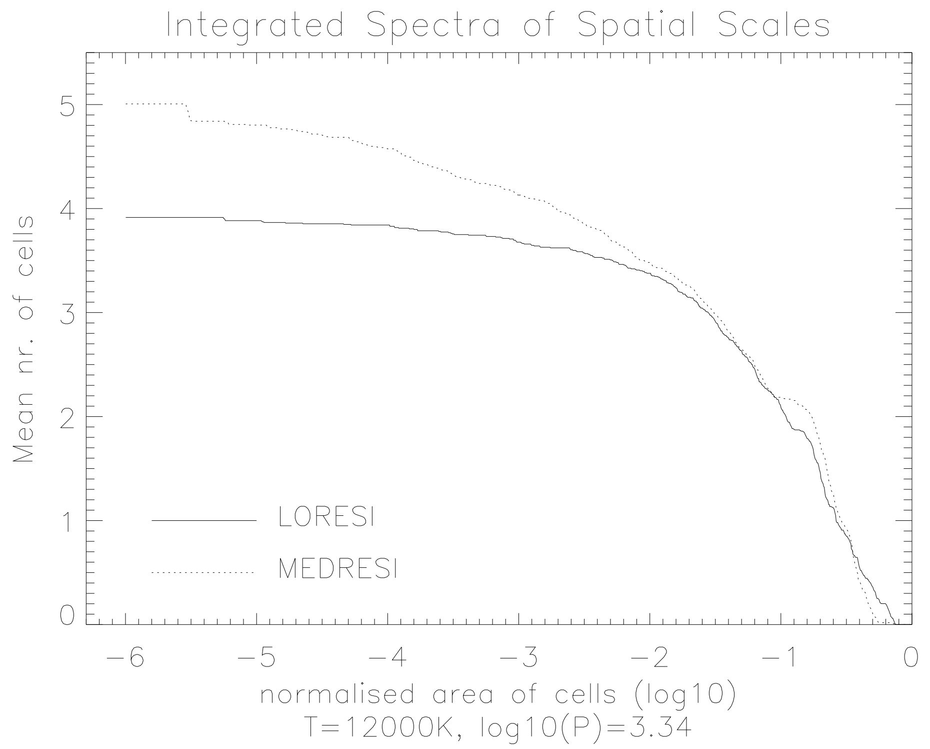


Figure 7.7: The integrated spectra of spatial scales for both LORESI and MEDRESI. The axes are the same as in figure 7.6. Every point on the graph gives the mean number of cells for all cells which have areas larger than or equal to that point, i.e. the spectra in figure 7.6 has here been integrated from right to left except that no special binned intervals are being used. Note that both models produce roughly as many cells which are larger than $\approx 10\%$ (although the proportions differ). But for smaller scales the models diverge, meaning that MEDRESI produces more cells which are smaller than $\approx 10\%$.

on smaller scales while both models produce similar amount of large scale cells.

7.4.4 Getting a comparable quantity

In order to compare different models it is useful to get some simple quantities, e.g. single numbers which are easy to interpret. Can such a quantity be taken from the method described here? We have so far only analysed two models and more would be needed. But to begin somewhere we look for potentially meaningful points in the figures.

The mean number of cells larger than or equal to 10^{-1} or 10% can be of interest since both models produce similar amount of these cells. It is also lying on an approximate boundary between the more permanent giant granules and the smaller shorter-lived ones (see figures 7.4 and 7.5).

Finding another number representing the total number of cells over all scales could also be of interest, for that I chose the size 10^{-4}. I don't want to go to smaller scales since by looking at figures 7.4 and 7.5 it seems that below this size the meaning of the points becomes more ill defined, i.e. whether they are noise or actual cells.

For both models the mean numbers of cells at those two boundaries are

	LORESI	MEDRESI
n_{-1}	2.1	2.2
n_{-4}	3.8	4.6

Another quantity of interest could be the ratio of n_{-1} and n_{-4} for each model since it gives a hint of the relative number of cells at different scales

$$\left(\frac{n_{-4}}{n_{-1}}\right)_{\text{LORESI}} = 1.8$$

and

$$\left(\frac{n_{-4}}{n_{-1}}\right)_{\text{MEDRESI}} = 2.1$$

These ratios reflect the relatively larger number of cells smaller than 10% in the MEDRESI model.

Chapter 8

Conclusions

Here is a short list of general conclusions based on the analysis of the LORESI and MEDRESI models.

- Cells are not represented by a single typical size, instead they exist on different scales, differing by orders of magnitude, see all plots in chapter 7.
- Using higher numerical resolutions increases the number of cells on small scales without destroying cells on large scales. This hints at a hierarchical structure of smaller cells superposed on the bigger cells, see figure 7.7.
- On average there are 4-5 cells covering the surface including 2 cells which cover more than 10% of the total surface area, see figure 7.7.

But how well does this compare to the results by Schwarzschild (1975)?. He concluded that the surface of a supergiant would be dominated by very few cells, i.e. 12 to 90 cells. The models prefer an even lower number of 4-5 cells but given the uncertainty of the scaling arguments used in his article this feels like an excellent agreement (specially when compared to the two million or so granules on the Sun). However, Schwarzschild seeks out a single dominant scale of cells while the models give hint of cells existing over a range of different spatial scales.

8.1 Outlook

It would be useful to apply the same method to analyse more models, Freytag is now making models with still higher resolution of 235^3 gridpoints, as well as models with 171^3 gridpoints. These have to be analysed in order to get better statistics of the different spatial scales.

The connection between the velocity field and the spatial scales of cells is also of interest. This could e.g. be used to determine which scale could potentially be dominant in the Doppler shifts. The CCS algorithm can be easily extended for such an analysis since additional statistics can be collected for each cell.

Appendix A

IDL source for the CCS algorithm

A.1 Creating the cell mask

This is the IDL source code for the filter `get_cellmask5` which takes in the temperature and pressure 3D arrays and returns a mask containing the cells to be measured and counted. The number 5 in the name refers to this being the fifth algorithm which was developed.

A.1.1 Minimizing noise in the data

In the IDL implementation I try to minimize the noise caused by ill-defined regions, i.e. regions which are only few gridpoints and are most likely just some irrelevant noise. The parameter `MINVOX` is used for this. The code will take away any gridpoints from the resulting region-mask which have fewer than `MINVOX` neighbours around them. The parameter `NEIGHBOURHOOD` gives the size of the neighbourhood, the neighbourhood is a cubical volume centered on each gridpoint with side length `NEIGHBOURHOOD`. The IDL command `smooth` is used to find the neighbours, therefore I refer to the section describing the smoothing algorithm in the IDL manual for further information.

A mask containing the whole boundary surface shell is returned through the variable `shell`, the calling script uses it to count the total number of gridpoints in the shell in order to normalise the cell sizes.

Note that in the comments of the code I usually refer to gridpoints as *voxels.*

```
; +
; get_cellmask5
;
; Takes in two 3D arrays, T and P and returns a
; mask which covers all convection cells found
; which intercept the T=Tmin isosurface.
;
; The mask elements are 1 inside a cell, and 0
; outside of it. The mask returned is of type
; BYTE as well as the shell
;
; KEYWORDS:
;
;  TMIN
;    a shell is made around T=TMIN
;
;  PMAX
;    the T shell is cut with values
;    of alog10(P) le PMAX
;
;  GETSHELL
;    make it equal a variable to get the shell
;
;  SHELLTHICKNESS
;    approximate thickness of the
;    shell in voxels
;
;  NEIGHBOURHOOD
;    postprocessing of mask. Side length
;    of box to count nr. of voxels in.
;
;  MINVOX
;    voxels with at least MINVOX nr. of
;    voxels in their neighbourhood in
;    the mask will survive, otherwise deleted.
;
; There are no default values. Some typical
; values are
;    TMIN=12000.0
;    PMAX=3.34
;    SHELLTHICKNESS=9
;    NEIGHBOURHOOD=5
;    MINVOX=62
;
; Sigurdur Finnsson 25.mars 2003
;-
;

function get_cellmask5, T_in, P_in, GETSHELL=shell, $
         TMIN=Tmin, $
         PMAX=Pmax, $
         SHELLTHICKNESS=shellsmooth, $
         NEIGHBOURHOOD=neighbourhood, $
         MINVOX=minvox
```

```
   ; pass T and P by value
   T=T_in
   P=P_in

   ; The minimum allowed value of the smooth function
   ; so only points with at least minvox nr. of
   ; voxels in their neighbourhood
   ; are equal to one
   s_min = double(minvox) / neighbourhood^3

   ; create a thin shell around a specific
   ; temperature value
   Tvol=T ge Tmin
   Tshell=smooth(DOUBLE(Tvol),shellsmooth)
   Tshell=(Tshell ge 0.01) AND (Tshell le 0.99)

   ; no values are outside Tvalue
   Tshell=Tshell AND Tvol

   ; create a mask out of other pressure
   ; to cut through Tshell with
   P=alog10(P)
   Pvol=float(P le Pmax)

   ; the final mask which masks the cells
   mask=Tshell AND Pvol

   ; smoothing the mask to get rid of noise voxels
   masksmoothed=smooth(float(mask), neighbourhood)
   mask=masksmoothed ge s_min

   ; smooth the shell using same values for consistency
   Tshell2=smooth(float(temporary(Tshell)), neighbourhood)
   shell=temporary(Tshell2) ge s_min

   return, mask ; return the generated mask

end
```

A.1.2 Finding isolated regions in a mask

This code takes a mask as an input, finds all isolated regions in it and counts the number of gridpoints in each. It returns an array with the size of each region in the order they were found.

```
; findregions.pro
;
```

```
; takes a 3D array in vol and counts how many
; separate volume-patches exists
; within the array, plus how many gridpoints
; each of them contain.  Cells with
; value 1 are considered part of a volume patch,
; other values are outside.
;
; Returns a 1D-array ("vector") where each element
; corresponds to a found
; volume-patch. The number is the size of the
; found volume-patch in nr. of voxels.
;
; WARNING! Don't use on arrays equal to or
; exceeding 1000x1000x1000 elements
; ------------------------------------------------
; As of now the returned array is of type LONG,
; which means it can handle  numbers up to ca. 2e9,
; this can be a issue when using 3D arrays which are
; larger than 1000x1000x1000, then the size of each
; volume patch can exceed biggest
; number possible. Then this function should be
; extended using LON64 (64 bits).

function findregions, vol_in

   ; pass vol_in by value
   vol=BYTE(vol_in)

   ;regions which are found are cleared with value val
   val=0B

   ;search for some element which is nonzero
   i=0 & j=0 & k=0

   ;index into returned data vector
   cellsizes=LONG(-999)

   arrsize=size(vol, /DIMENSIONS)
   zeroplane=make_array(arrsize(0), arrsize(1), /BYTE, value=0B)
   zeroline=make_array(arrsize(0), /BYTE, value=0B)

   for k=0,(arrsize[2]-1) do begin

   if(NOT ARRAY_EQUAL(vol[*,*,k], zeroplane, /NO_TYPECONV)) then begin
     for j=0,(arrsize[1]-1) do begin

       if(NOT ARRAY_EQUAL(vol[*,j,k], zeroline, /NO_TYPECONV)) then begin
           for i=0,(arrsize[0]-1) do begin
```

```
              if (vol[i,j,k] ne 0B) then begin
                 print, 'Found region at: ', i, j, k
                 tmpi=i & tmpj=j & tmpk=k

                 ;must create a pointer since region
                 ;has to be used again and again, and
                 ;usually with different dimensions each
                 ;time. IDL tends to complain if region
                 ;is a normal array.
                 region=ptr_new(search3d(vol, tmpi,tmpj,tmpk, 1B, 1B))

                 ;zero the volume-patch so it wont be found again
                 vol[*region]= val

                 ;volume of the found volume-patch
                 cellsizes=[cellsizes, N_ELEMENTS(*region)]

                 ptr_free, region
              endif ;Endof checking each element in a line

           endfor

        endif  ; Endif zeroline
     endfor

  endif ; Endof zeroplane

  endfor

   ; cut away the extra cell I had to add in the beginning
   cellsizes=cellsizes(1:*)
   return, cellsizes

end
```

Bibliography

M. B. Al-Malki, J. F. L. Simmons, R. Ignace, J. C. Brown, and D. Clarke. Scattering polarization due to light source anisotropy. I. Large spherical envelope. *Astronomy & Astrophysics*, 347:919–926, July 1999.

E. E. Bloemhof, C. H. Townes, and A. H. B. Vanderwyck. Diffraction-limited spatial resolution of circumstellar dust shells at 10 microns. *Astrophysical Journal Letters*, 276:L21–L24, January 1984.

D. Burns, J. E. Baldwin, R. C. Boysen, C. A. Haniff, P. R. Lawson, C. D. Mackay, J. Rogers, T. R. Scott, P. J. Warner, D. M. A. Wilson, and J. S. Young. The surface structure and limb-darkening profile of Betelgeuse. *Monthly Notices of the Royal Astronomical Society*, 290:L11–L16, September 1997.

D. F. Buscher, J. E. Baldwin, P. J. Warner, and C. A. Haniff. Detection of a bright feature on the surface of Betelgeuse. *Monthly Notices of the Royal Astronomical Society*, 245:7P–11P, July 1990.

B. Freytag. Poster: Radiation Hydrodynamics Simulations in 3D with COBOLD. In *Workshop: Stellar Atmosphere Modeling*, September 2001. Tübingen, Germany.

B. Freytag, M. Steffen, and B. Dorch. Spots on the surface of Betelgeuse – Results from new 3D stellar convection models. *Astronomische Nachrichten*, 323:213–219, 2002.

R. L. Gilliland and A. K. Dupree. First Image of the Surface of a Star with the Hubble Space Telescope. *Astrophysical Journal Letters*, 463:L29+, May 1996.

L. Goldberg. The variability of alpha Orionis. *Publications of the Astronomical Society of the Pacific*, 96:366–371, May 1984.

D. F. Gray. Betelgeuse and Its Variations. *Astrophysical Journal*, 532:487–496, March 2000.

D. F. Gray. Betelgeuse: Giant Convection Cells. *Publications of the Astronomical Society of the Pacific*, 113:1378–1385, November 2001.

J. P. Harrington. Polarization of Radiation from Stellar Atmospheres. The Grey Case. *Astrophysics & Space Science*, 8:227–+, 1970.

P. H. Hauschildt, E. Baron, and F. Allard. Parallel Implementation of the PHOENIX Generalized Stellar Atmosphere Program. *Astrophysical Journal*, 483:390–+, July 1997.

D. P. Hayes. Variations of Betelgeuse's optical linear polarization over four consecutive observing seasons - 1979-1983. *Astrophysical Journal Supplements*, 55: 179–188, June 1984.

C. A. Iglesias, F. J. Rogers, and B. G. Wilson. Spin-orbit interaction effects on the Rosseland mean opacity. *Astrophysical Journal*, 397:717–728, October 1992.

H. S. Jones. The radial velocity variations of a Orionis and a Scorpii. *Monthly Notices of the Royal Astronomical Society*, 88:660–+, June 1928.

V. A. Klückers, M. G. Edmunds, R. H. Morris, and N. Wooder. Reality and the speckle imaging of stellar surfaces - II. The asymmetry of Alpha Orionis. *Monthly Notices of the Royal Astronomical Society*, 284:711–716, January 1997.

R. F. Sanford. The Variation in the Radial Velocity of α Orionis from 1923 TO 1931. *Astrophysical Journal*, 77:110–+, March 1933.

M. Schwarzschild. On the scale of photospheric convection in red giants and supergiants. *Astrophysical Journal*, 195:137–144, January 1975.

J. Stebbins and C. M. Huffer. Photo-electric studies of five variable stars. *Publications of the Washburn Observatory*, 15:178–213, 1931.

R. Stothers. Convective Envelopes and Radial Pulsation of Massive Red Supergiants. *Astronomy & Astrophysics*, 18:325–+, May 1972.

P. G. Tuthill, C. A. Haniff, and J. E. Baldwin. Surface imaging of long-period variable stars. *Monthly Notices of the Royal Astronomical Society*, 306:353–360, June 1999.

H. Uitenbroek, A. K. Dupree, and R. L. Gilliland. Imaging Spectroscopy of Betelgeuse in the Ultraviolet. In *ASP Conf. Ser. 154: Cool Stars, Stellar Systems, and the Sun*, pages 393–+, 1998.

R. W. Wilson, V. S. Dhillon, and C. A. Haniff. The changing face of Betelgeuse. *Monthly Notices of the Royal Astronomical Society*, 291:819–+, November 1997.