A treatise on the state of KOSMA-τ 3D

Craig Yanitski

This document has been created to record what major issues already existed in KOSMA-τ 3D—created by Silke Andree-Labsch and further developed by Christoph Bruckmann—when I started working on it and how they were fixed. While it may seem slightly unnecessary, these changes in the code greatly affect how it calculates the emission of a PDR model. I will not describe trivial alterations in here like converting it to `python3`, making it run out of the terminal, or any of the changes I made to save on execution time (of which there are a few).

1 From Silke's thesis...

The PDR emission is calculated from a probabilistic approach to the fractal PDR structure. The PDR is modeled from multiple instances of the spherical KOSMA-τ PDR simulations (hereafter called *masspoints*). The probability is applied to how many (if any) of these masspoints can be seen. For observed element (read: *molecule*). the intensity and optical depth depend on the observing velocity (v_{obs}) and the velocity of the masspoint (v_i).

$$I_{x_i}(v_{obs}) = \sum_{j=1}^{n_M} k_{j,i}\overline{I_{j,line}}\exp\big[-\frac{1}{2}\big(\frac{v_i - v_obs}{\sigma_{line,j}}\big)^2\big]$$
$$\tau_{x_i}(v_{obs}) = \sum_{j=1}^{n_M} k_{j,i}\overline{\tau_{j,line}}\exp\big[-\frac{1}{2}\big(\frac{v_i - v_obs}{\sigma_{line,j}}\big)^2\big]$$

In this case, the intensity and optical depth are summed over the observed number ($k_{j,i}$) of each masspoint (j) at a particular velocity (i). This is combined with the probability of seeing a combination, p_{x_i}, where x_i refers to an individual combination. Since we summed over the visible masspoints, this probability is a product of the probabilities of seeing each masspoint.

$$p_{x_i} = \prod_{j=1}^{n_M} B(k_{j,i}|p_{j,i}, N_{j,i})$$

Henceforth I will denote $B(k_{j,i}|p_{j,i}, N_{j,i})$ as $p_{m_j,i}$ for masspoint m_j at velocity v_i. Finally the emission of each element is combined, and the contribution from combinations at different velocities summed-over.

$$\langle I\rangle_i(v_{obs}) = \sum_{x_i} p_{x_i} I_{x_i}$$
$$e^{-\langle\tau\rangle_i(v_{obs})} = \sum_{x_i} p_{x_i} e^{-\tau_{x_i}}$$

$$\langle I\rangle(v_{obs}) = \sum_i I_i(v_{obs})$$
$$\langle\tau\rangle(v_{obs}) = \sum_i \tau_i(v_{obs})$$

2 The issues

On the implementation side, the probability of seeing different masspoints is calculated for each combination of masspoints (depending on how many can be seen). The probability is determined by the masspoint and its velocity: from the size of the masspoint compared to the projected area of the voxel ($p_{vox} = \frac{\pi R_j^2}{s^2}$ for voxel side length s) one can determine the expected number of masspoints that can be seen at a certain velocity ($\mu_{j,i}$) and its "standard derivation [*sic*]" ($\sigma_{j,i}$).

$$\begin{aligned}\mu_{j,i} =& N_j p_{vox} \\ \sigma_{j,i} =& \sqrt{N_j p_{vox}(1 - p_{vox})}\end{aligned}$$

This is used to determine the combinations, as the number of masspoints seen is $\mu_{j,i} \pm 3\sigma_{j,i}$ to account for $\geq 98\%$ of the mass at that velocity. The combinations are then summed over, using their respective probabilities p_{x_i}. The issue with this is that it was implemented in a way such that the contribution in accumulated in each combination. For an example, let m_j refer to masspoint j in a combination. If there is a combination of $m_1 = 10M_\odot$ and $m_2 = 100M_\odot$ masspoints, the code yields I_{m_1} for m_1 and $I_{m_1} + I_{m_2}$ for m_2, while they are added separately with their individual probabilities. This is an erroneous treatment of the procedure, and contrasts what is written in the thesis.

Beyond this, there is also an error with how the probability is treated. The probability of seeing a particular combination is $p_{x_i} = \prod_{j=1}^{n_M} p_{m_j,i}$ for all the masspoints n_M in the combination. Similar to how the emissions were accumulated over the combination, the probability is accumulated over all combinations. This forces the averaged expression to become:

$$\begin{aligned}\langle I \rangle_i(v_{obs}) =& \sum_{x_i} \left(\sum_j \prod_{j=1}^{j} p_{m_j,i} I_{m_j} \right) \\ e^{-\langle \tau \rangle_i(v_{obs})} =& \sum_{x_i} \left(\sum_j \prod_{j=1}^{j} p_{m_j,i} e^{-\tau_{x_i}} \right)\end{aligned}$$

The probability $p_{m_j,i}$, as before, is the contribution of m_j to p_{x_i}. This is clearly incorrect. Using the same two-masspoint example and considering the combinations ((0,0), (0,1), (1,0), (1,1)), this method would have a product of all of the probabilities, when we are not looking for a probability of seeing all combinations at once (nor is that conceptually possible).

The final issue regards a bug in the implementation of the intensity calculation. It does not have to do with the the exact properties being calculated, but rather in the way the code is written. The intensities are all wrong due to this bug. The issue is that when the emission from each masspoint is accumulated, what is saved is actually a reference to the emission rather than the value itself. This is fine if both masspoints have zero emission, but it causes a problem if there is an emission. With the previous set of combinations, the second combination (0,1) should have zero emission for the first masspoint and emission from the second masspoint. Applied to the Milky Way model with a resolution of 1000 pc, the corresponding probabilities are 99% and $\sim 10^{-4}$. The bug in the code makes the first masspoint have the emission of the second, and the second masspoint has the emission of the first in the next combination. Combined with the probability in the averaging equation, this bug gives a probability of 99% to seeing a masspoint, and the emission of each voxel is greatly overestimated. For concrete numbers, the bug gives a CO intensity of 14 to a voxel that should only have a CO intensity of $\sim 10^{-2}$.

3 The solution

This should be fairly evident now, but I just want to clarify what I have changed. Obviously I had to fix the aforementioned bug by inserting a `copy` statement. As for the mathematical errors, I adjusted the probability calculation to be a product of the probability of seeing the masspoints in a given combination, so it is truly a probabilistic approach of a certain combination of masspoints being in the line-of-sight. This allows the average emission of the ensemble to be calculated from the total emission from each combination. Thus only the final sum of the masspoint emissions is saved (removing the error of an accumulating intensity).

It is perhaps a bit worrisome that difference in emission is so large in the Milky Way model. The probability scales with the ratio of clump radius and voxel width, so a smaller resolution should have higher combinations of clumps. The discrepancy in emission thus should not be so large for a smaller model, such as the Orion Bar in Silke's thesis.

The main improvement I made to this code is completely rewriting it. Some of the most basic principles are preserved, but I needed to make it class-based to increase encapsulation. This bypasses the coding error of saving a reference to the emission rather than copying its value. The main hierarchy is the `Model` class has-an instance of `VoxelGrid`, which contains however many `Voxel` instances is needed for the specified shape and resolution. To agree with the disk-model of Christoph Bruckmann, each voxel has two instances of `Ensemble`: one for the dense clump and one for the diffuse interclump. Each ensemble has multiple instances of `Combination` to account for what emission can be seen. Finally, each combination has the specified number of `Masspoint` instances, where the intensity is interpolated from the `KOSMA-`τ grid files, and this is summed-up as specified for each voxel. This is the basic structure, but some other classes were created to assist the computation. The `Shape` class is owned by `Model`, and this specifies how the voxels are arranged with an instance of `Dimensions`. `Model` also has-an instance of `Orientation`, which is used to integrate the intensity for a given orientation of the model. There is also a `Observations` class that takes an argument specifying the location of the tables of PDR information. This is given as an argument to `Interpolate`, which pre-interpolates the data as functions, which can be called much more quickly than interpolating the data each time. Finally, the information we require is not all of the emissions given in the `KOSMA-`τ grid, so there are the `Molecules` and `Dust` classes, owned by `Model`. These are separated so `Masspoint` can distinguish them when calculating the emissions, since dust is not velocity-dependent.