FRACS-Ideas for implementation

everybody

November 10, 2014

Abstract

Some notes about what is need to improve in the code and how to improve it. Anyone can add comments, corrections and notes on it.

1 What we want to implement [1]

1.1 Model grid

1. How to build the best grid in order to sample of high density (optical depth)?

1.2 Image grid

1. An optimal grid for the image can also be adopted in order to better follow the intensity distribution of the disc;

2. This can also help us to solve the problem of pixelized images in the regions of low flux.

1.3 Models at high Optical Depths

In order to model young stellar objects it is necessary to become FRACS compatiple with high opticla depths. Ideas:

1. To integrate the intensity contribution in the inverse direction (from the image back to the disc). We can stop the integration when the optical depth is too large and the intensity stops changing in a significantly way; e and flux in the high density parts? Or is it not useful for us in the case of the simplified radiative transfer from FRACS?

1.4 Velocity Fields

1. Including lines: we need to introduce a velocity field at each grid point, following a parametrized description (for example: Keplerian rotation);

1.5 Spectral Lines

1. Gilles's idea: to use a simplified description oflines seems very nice and better agrees with the FRACS philosophy. For example in [2], line is treated as a Gaussian profile at the visible disc surface, without any radiative transfer.

2. We can do something better, but, considering a given line profile at each grid point across the ray.

3. We should also check that the approximation of no scattering is valid (within a certain precision) for lines.

1.6 Other continuum opacities

1. Should we add also other continuum opacities to the model, like (free-free, bound-free)? It could be performed analytically, without to much difficulty if we impose the temperature and density profiles

2 Priorities on code implementation [3]

He already developped the linear interpolation of the log in each cells. However, if we want to avoid storing 2 times the values of kappa at cell vertices, we might want to use some implementation as described in [4] and [5].

Also, in [6] they use another way to interpolate the source function within each cells (but they can identify the neighbors easily since their mesh is regular).

It is important to identify the cells having an intersection with the physical boundary (specified by the user). In those cells, the integration must start/stop from the physical boundary. Otherwise, some structures (more or less the corners of the tori) appear in the images when the optical depth gets high.

We must integrate from the outer boundary into the medium. By doing so, we cannot longer used up/down "symmetries" specific to a disk configuration but we expect that the gain in computing time will be important. The criterion to stop the integration might be

1. stop when there is no further evolution of the specific intensity while integrating,

2. when the optical depth reaches a certain prescribed value.

He thinks that this must be more or less the same, but we must check this out.

It is not clear if the mesh refinment on the optical depth is a good idea. Mainly because since the optical depth depends on the frequency, ideally we should have a mesh for each frequency ... Otherwise, for frequency where the medium is optically thin we will loose a lot of computing by using too fine a mesh (scaled according to the most critical frequency where tau is the largest).

We also mentionned that it would be cool to make the code deals also with 3D configurations and let the user choose the geometry and mesh he wants. He coded that in 'tapas' (his Monte Carlo radiative transfer code) and we can pick it from there.

We might need to develop an adaptive mesh in the image plain itself using a quadtree as well. To do so, we can generate a prescribed number of points uniformly distributed on the projection of cell centers (ellipses) onto the image plane. Parallelizing this is not necessarily easy (if we want to) because of the tree data structures (load balacing is not that obvious) but there are some standard algorithms to do just that.

If we want priorities, he suggests :

1. Boundary fitting (to get rid of the nasty features in the images).

2. Integration from outside into the medium

3. Look carefully at the integration in each cells (interpolation, locating neighbors, ...).

4. Adaptive mesh in the image plane.

5. Arbitrary geometry and mesh.

Points 1 and 2 are relatively easy, point 3 is more critical especially for line transfer, he thinks.

3 First implementation – image.cpp

In order to eliminate the *nasty features*, visible in Figure 1, we started our implementation by changing some aspects in image.cpp routine, mainly in

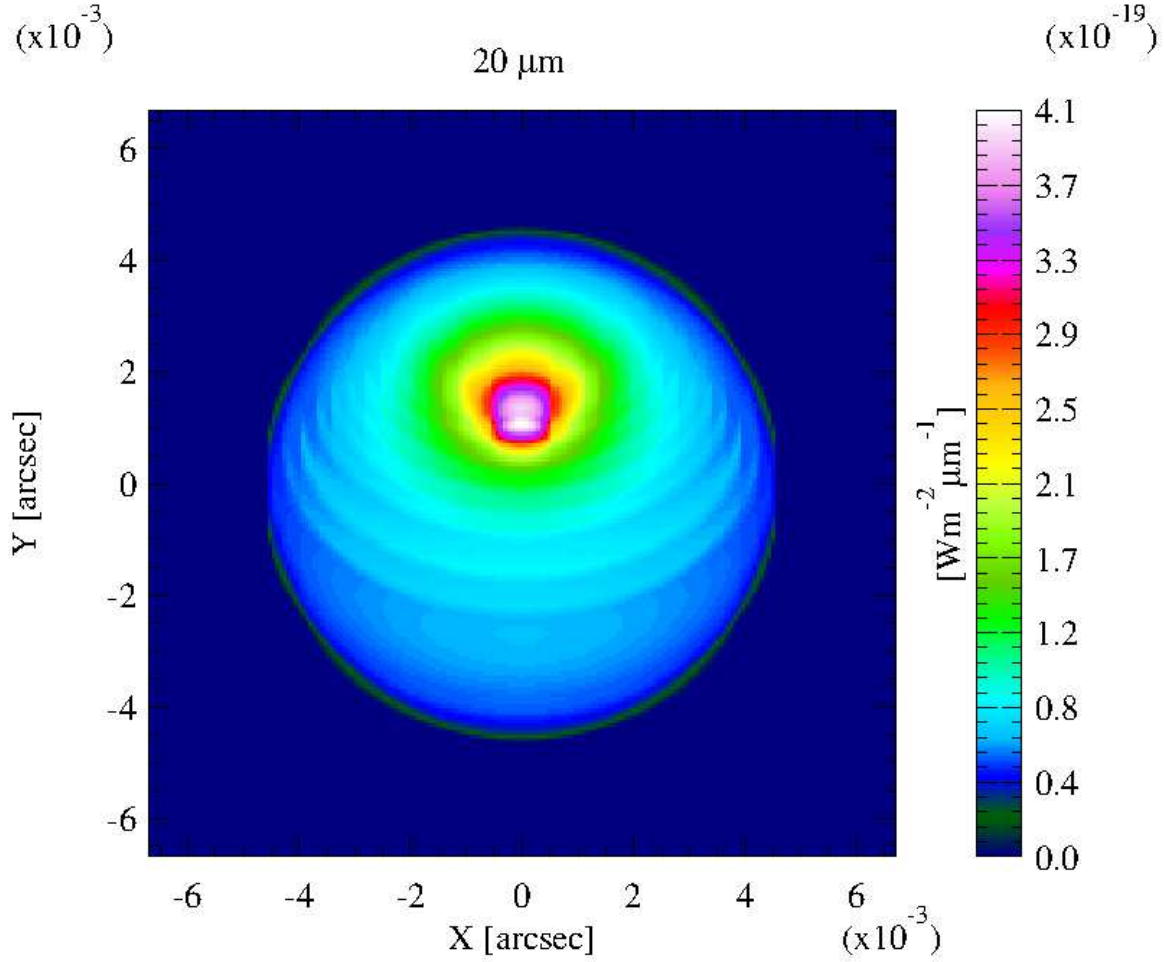

Figure 1: *Image with nasty features that we want to get rid of*

the way of interpolate. The actual version of the code treats the coordinate system following the description described in [7], which one of those figures we reproduce in Figure 2.

From Figure 2 we define $r_s(X, Y, i)$ as the position vector along a ray, given in the model system of coordinates by:

$$r_s(X, Y, i) = \begin{pmatrix} -X \\ -Ycos(i) + ssin(i) \\ Ysin(i) + scos(i) \end{pmatrix} \tag{1}$$

And, in the code, it is defined as:

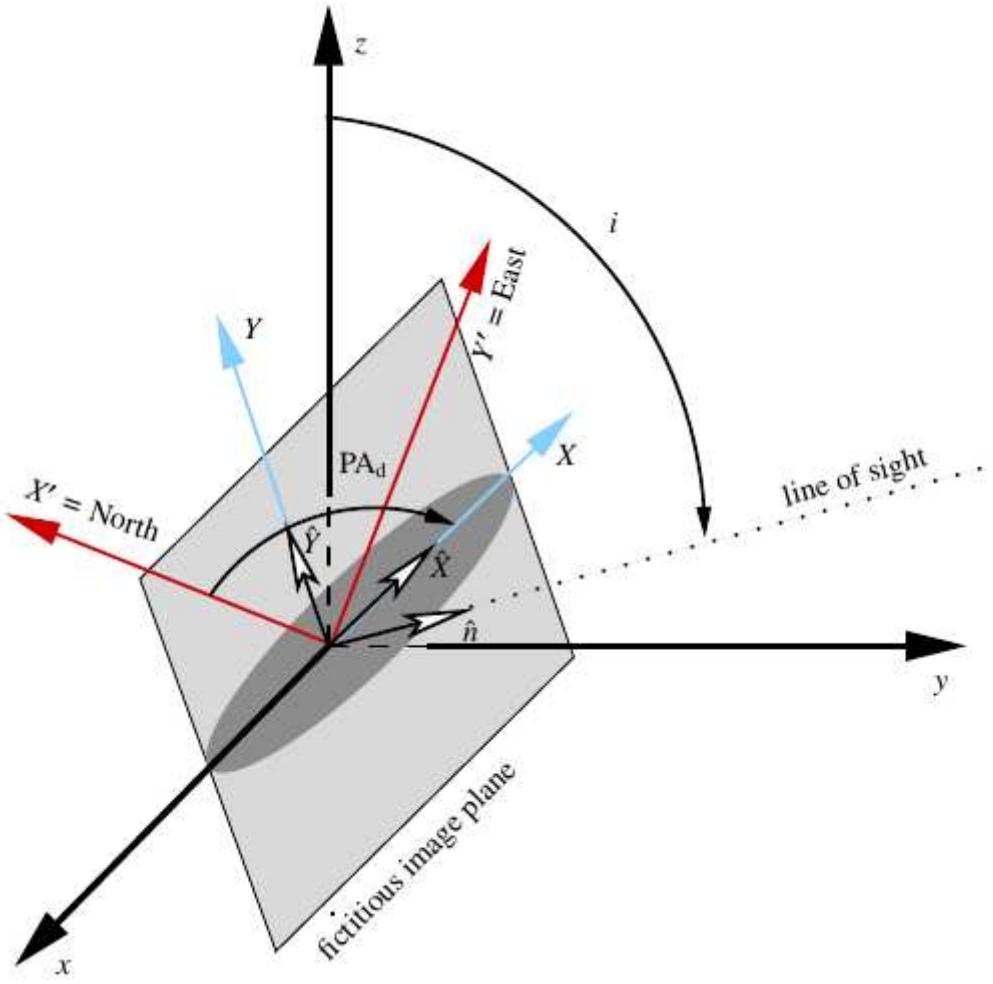

Figure 2: *Coordinate systems. The shaded ellipse represents a disc viewed by the observer [7]*

Listing 1: image.cpp - Setting coordinates

```
y=(iy+.5)*deltaY-halfSize;

double yOld=y;

//Rotation

y=cosTheta*yOld;
z=-sinTheta*yOld;
```

As a remember, in the code we're dealing with the ray direction from the center image to the outside (sign of coordinate). Our idea is to do the opposite: ray comming from outside going to inside.

In [7], for the calculation of s, we have:

$$\begin{aligned} \vec{r_s}^2 &= R_{out}^2 \\ X^2 + (-Y\cos(i) + s\sin(i))^2 + (Y\sin(i) + s\cos(i))^2 &= R_{out}^2 \\ s &= \sqrt{R_{out}^2 - (X^2 + Y^2)} \end{aligned} \tag{2}$$

In the same article, the authors define the optical depth as:

$$\tau_\lambda(X, Y, i; s) = \int_s^{\sqrt{R_{out}^2 - R^2}} \kappa_\lambda^{ext}(r'_s) ds' \tag{3}$$

and Intesity as:

$$I_\lambda(X, Y, i) = \int_{-\sqrt{R_{out}^2 - R^2}}^{\sqrt{R_{out}^2 - R^2}} \kappa_\lambda^{abs}(r_s) B_\lambda[T(r_s)] e^{-\tau_\lambda(X,Y,i;s)} ds \tag{4}$$

both of them in a analitically way.

In the numerical case we have to re-write intensity as:

$$\Delta I_\lambda = \int_{s_i}^{s_i + \Delta s} \kappa_\lambda^{abs}(s) B_\lambda[T(s)] e^{-\tau_\lambda(s)} ds \tag{5}$$

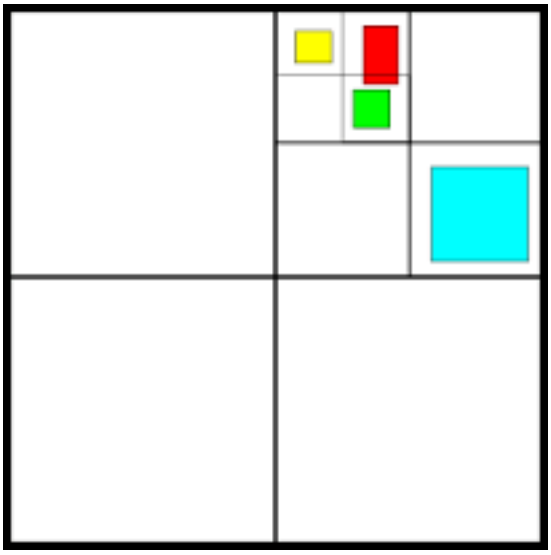

Figure 3: *Coordinate systems in mesh.cpp [7, 4]*

From linear interpolation and following the 3, we have:

$$\frac{d_{max} - d_{min}}{\Delta z} = (z_0 - z_{min}) + d_{min} \tag{6}$$

where,

$$\begin{aligned} d_{max} &= \frac{d_{11} - d_{01}}{\Delta \rho}(\rho - \rho_{min}) + d_{01} \\ d_{min} &= \frac{d_{10} - d_{00}}{\Delta \rho}(\rho - \rho_{min}) + d_{00} \end{aligned} \tag{7}$$

Substitute Eq(7) in Eq (6), we find the coordinates of each vertices of the cell. In FRACS, such definition is made at mesh.cpp:

Listing 2: mesh.cpp - Setting coordinates

```
double rMax=position(0)+halfSize;
double rMin=position(0)-halfSize;
double zMax=position(1)+halfSize;
double zMin=position(1)-halfSize;

double rMax2=rMax*rMax;
double rMin2=rMin*rMin;
double zMax2=zMax*zMax;
double zMin2=zMin*zMin;


double d00=holder->getDensity(rMin2,zMin2);
double d01=holder->getDensity(rMin2,zMax2);
double d10=holder->getDensity(rMax2,zMin2);
double d11=holder->getDensity(rMax2,zMax2);

weights.resize(4);

if(d00!=0. && d01!=0. && d10 !=0. && d11!=0) {
    weights(3)=(d00+d11-d10-d01)/(size*size);
    weights(2)=(d10-d00)/size;
    weights(1)=(d01-d00)/size;
    weights(0)=d00;
  } else weights=0.;

}
```

In [6], the authors have a different way of rewriting Eq(3):

$$\tau_\nu(s) = \int_0^s \kappa_\nu^{ext}(r_0 + s'_{\vec{n}})ds' \tag{8}$$

Numerically, it means:

$$\Delta\tau_\nu(s) = \int_{s_i}^{s_i+\Delta s} \kappa_\nu^{ext}(s)ds \tag{9}$$

where in [6], they assume $\kappa_\nu^{ext} \approx A + B(s - s_i)$. In this sense, we can rewrite the Eq(9) as:

$$\Delta\tau_\nu(s) = \int_{s_i}^{s_i+\Delta s} (A + B(s - si))ds = A\Delta s + B\frac{\Delta^2 s}{2} \tag{10}$$

In the same way, the Intensity is written as:

$$\Delta I_\nu = \int_{s_i}^{s_i+\Delta s} \kappa_\nu^{ext}(s) S_\nu(s) e^{-\tau_\nu(s)} ds \tag{11}$$

where, the Source function is written as Ce^{D_s}. So, we have:

$$\Delta I_\nu = C \int_{s_i}^{s_i+\Delta s} [A + B(s - s_i)] e^{D(s-s_i)} e^{-\tau_\nu(s)} ds \tag{12}$$

Assuming $(s - s_i) = l$, we can rewrite as:

$$\Delta I_\nu = C \int_0^{\Delta l} [A + B(l)] e^{Dl - Al + B\frac{l^2}{2}} dl \tag{13}$$

Problems here: time consuming evaluating log function → quadratic function must be evaluated by Gamma Function. Also we have to check Taylor expansion ...

4 Line Emissivity Problem [3]

The problem with line emissivity is that it is proportionnal to the line profile and that it is very sharp.

This line profile is the result of three contributions :

1. A Lorentzian profile that is the "natural" line width : the energy level has a certain lifetime, and the line a related width in energy/frequency via Heisenberg inequality. Classically, if we considered an electron linked to the nucleus with a damped (because the e radiates) harmonic oscillator excited by a plane wave, we also do get the Lorentzian.

2. The doppler shift due to the random motion of particles in the gas and the microturbulence → This gives us a Gaussian profile.

In fact, the actual profile is the convolution of both profile. In practice, the Lorentzian is much sharper than the Gaussian and we can assume a Gaussian profile for the line.

If Φ_ν is the line profile, because of the (macroscopic) velocity, the line is centered on $\nu_{ij}(1 + v.n/c)$, where ν_{ij} is the frequency of the line. Because

of the Doppler shift, the frequency dependence is transform in a spatial dependence of line emissivity.

If we assume that v varies linearly within a cell we can identify where we have the line profile maximum in the portion of the considered ray crossing the cell. We just need to add integration points there. Otherwise, we could completly miss the line.

We need a faster way to generate the grid based on the gradient of some quantities (estimated from finite differences) and generates the mesh in a faster way based on the gradient of some quantities.

All those considerations described above were based on [8]. In that paper, the authors made notes on molecular line transfer (See Section 2 in this paper), where starting by the radiative transfer equation:

$$\frac{dI_\nu}{ds} = -\alpha_\nu I_\nu + j_\nu \tag{14}$$

where I_ν is the intensity of radiation, s is the length along the ray. Also, in [8], we have a set of balance equations for level populations:

$$n_u \left[\sum_{l<u} A_{ul} + \sum_{l \neq u} (B_{ul}\bar{J}_{ul} + C_{ul}) \right] = \sum_{l>u} n_l A_{lu} + \sum_{l \neq u} n_l (B_{lu}\bar{J}_{ul} + C_{lu}) \tag{15}$$

where n_u and n_l are level populations and A_{ul} and B_{ul} are the Einstein coefficients. C_{ul} are coefficients of collisional excitation and ul indices specifies the transition from the upper level (u) to lower level l. Equations (14) and (15) are coupled by the emission and absorption coefficients j_ν and α_ν:

$$j_\nu = \frac{h\nu_{ul}}{4\pi} n_u A_{ul} \Phi_{ul}(\nu) \tag{16}$$

$$\alpha_\nu = \frac{h\nu_{ul}}{4\pi} (n_l B_{lu} - n_u B_{ul}) \Phi_{ul}(\nu) \tag{17}$$

And also by the mean intensity ($\bar{J}_{ul}$), defined as:

$$\bar{J_{ul}} = \frac{1}{4\pi} \int_{4\pi} d\Omega \int_{0}^{\infty} I_\nu \Phi_{ul}(\nu) d\nu \tag{18}$$

where $\Phi_{ul}(\nu)$ is the line profile function and Ω is the spatial angle. The line profile function can be expressed as:

$$\Phi_{ul}(\nu) = \frac{c}{b\nu_{ul}\sqrt{\pi}} exp\left\{ -\frac{c^2[\nu - \nu_{ul} - (\mathbf{v}\cdot\mathbf{n})\nu_{ul}/c]}{2} \right\} \tag{19}$$

in the approximation of total redistribution over frequencies and a Maxwellian turbulent velocity distribution. Here ν_{ul} is the central frequency of the transition $u \to l$, $\mathbf{v}$ is the regular velocity, $\mathbf{n}$ is the unit vector associated with $d\Omega$, and b is a parameter related to the kinetic temperature T_{kin} and the most probable value of the microturbulent velocity V_{turb} by the expression:

$$b^2 = \sqrt{\frac{2kT_{kin}}{m_{mol}} + V_{turb}^2} \tag{20}$$

The intensity I_ν can be expressed in units of the brightness temperature, T_B by means of the Planck Equation:

$$I_\nu = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{kT_B}} - 1} \tag{21}$$

Also, we can use the definitions mentioned in [9], where the Source function (Eq(11)) is defined as:

$$S_\nu = \frac{j_\nu^c(R,\theta) + j_\nu^l(R,\theta,\bar{v})}{\alpha_\nu^c(R,\theta) + \alpha_\nu^l(R,\theta,\bar{v})} \tag{22}$$

which ray element is described in Figure 4.
In our case, we can re-write Eq(22) as:

$$S_\nu = \frac{\kappa_\nu^{abs} B_\nu + j_\nu(R,\theta,\bar{v})}{\kappa_\nu^{ext}} \tag{23}$$

where our $j_\nu(R,\theta,\bar{v})$ will be dependent of $\Phi_{ul}(\nu)$ (Eq(19)).

2

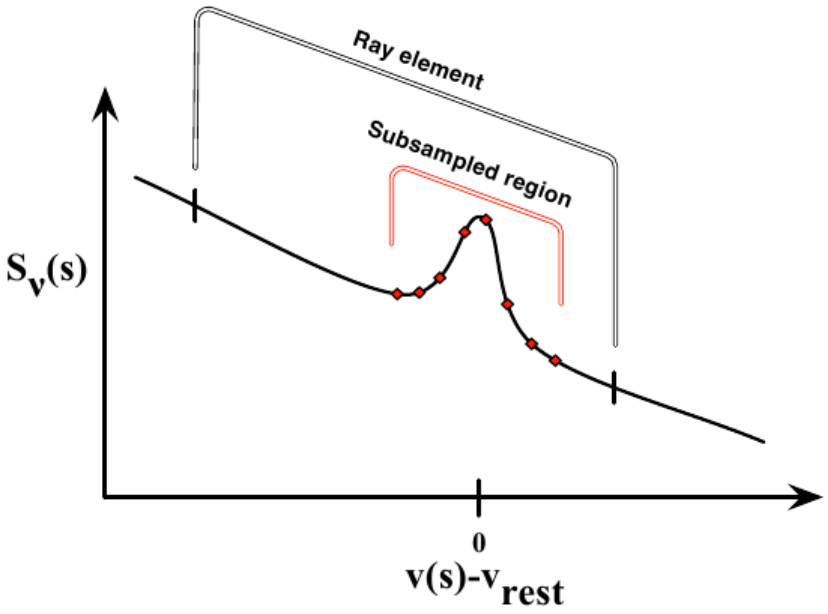

Figure 4: *Sketch of the RADlite ray element subsampling scheme. The illustration shows the line and continuum source S_ν, as function of radial velocity $(v(s))$, which in turn is a function of location s along the ray. Normally, the integral of the transfer equation is only evaluated at each end of the ray elementm as indicated in the figure, However, lines with narrow local broadening may be completely missed by integration. In such cases, the line is localized within the ray element and the integrand evaluated in a sufficient number of points across the line[9]*

References

[1] Armando's e-mail.

[2] Domiciano *et al*, **A resolved, au-scale gas disk around the B[e] star HD 50138**, http://arxiv.org/abs/1409.7394

[3] Gilles' e-mail.

[4] Frisken, S.; Perry, R., **Simple and Efficient Traversal Methods for Quadtrees and Octrees**, TR2002-41, 2002.

[5] Samet, H., **Implementing Ray Tracing with octrees and neighbor funding**, 1989.

[6] Woitke, P. *et al*, **Radiation thermo-chemical models of protoplanetary disks**, A&A 501, 383–406 (2009)

[7] Nicollini, G., Bendjoya, P., Domiciano, A., **Fast ray-tracing algorithm for circumstellar structures (FRACS) I. Algorithm description and parameter-space study for mid-IR interferometry of B[e] stars**, Astronomy and Astrophysics (2010) 00, http://arxiv.org/abs/1009.5616

[8] Pavlyuchenkov, Ya. *et al*, **Molecular emission line formation in prestellar cores**, ApJ, 689:335-350, 2008

[9] Pontoppidan, K. M. *et al*, **A New Raytracer for Modeling AU-Scale Imaging of lines from Protoplanetary Disks**, ApJ, 704:1482-1494, 2009