Fracs : developing a versatile ray-tracer

Report on the project, what we have done and what we will do

A. Domiciano, P. Bendjoya, G. Niccolini

May 29, 2019

Outline

What this is all about ?

- First of all: this is not real radiative transfer (sorry . . .)
- Fracs stands for "Fast Raytracing Algorithm for Circumstellar Structures"
- Once it was called plastic (it isn't real, it's in plastic) and Armando came up with that (better) name.
  - Are you geek enough to remember BSG :-) ?
- In short : it is a raytracing tool (we integrate the RT equation along rays and produce data cubes).

It is a reborn old project

- There is an old version (since 2011) : it still compiles, run and works . . .
- We are working on a new version (started in 2015-2016, and "restarted" well, a few weeks ago . . .
- I tried hard to came up with results from the fresh new version before this talk but . . . . We are almost there

An historical perspective . . .

- Fracs is the name of a project that we started ages ago (Niccolini et al. 2011; Domiciano de Souza et al. 2011).
  - It was the evolution of a previous code plastic (started in 2008, according to the copyright notice in the code).
  - Many versions existed, even one running on GPUs.
- After a long pause, we plan for a revival of the project in 2015 or so, including new features and capabilities.
- I started coding again and some progress was done, but just for a short while.
- Here we are, . . . in 2019, I'm coding again ! :-)

Today

- Ideally, I would have wanted to present the new tool (but it's not ready yet)
- I'll say a few words about the "old" version,
- and report on the progress of the project: what we are doing and planning to do.

An historical perspective . . .

- Fracs is the name of a project that we started ages ago (Niccolini et al. 2011; Domiciano de Souza et al. 2011).
  - It was the evolution of a previous code plastic (started in 2008, according to the copyright notice in the code).
  - Many versions existed, even one running on GPUs.
- After a long pause, we plan for a revival of the project in 2015 or so, including new features and capabilities.
- I started coding again and some progress was done, but just for a short while.
- Here we are, . . . in 2019, I'm coding again ! :-)

Today

- Ideally, I would have wanted to present the new tool (but it's not ready yet)
- I'll say a few words about the "old" version,
- and report on the progress of the project: what we are doing and planning to do.

Outline

The real stuff

Nasty equation, how innocent you look ...

$$
\begin{aligned}
I_\lambda(\vec{r},\hat{n}) &= I_\lambda(\vec{r}_0,\hat{n})\,\mathrm{e}^{-\tau_\lambda(\vec{r}_0,\hat{\vec{n}},s)} + \\
&+ \int_0^s \eta_\lambda(\vec{r}_0+s'\hat{n},\hat{n})\,\mathrm{e}^{-\left(\tau_\lambda(\vec{r}_0,\hat{\vec{n}},s)-\tau_\lambda(\vec{r}_0,\hat{\vec{n}},s')\right)}\,ds' \\
\tau_\lambda(\vec{r}_0,\hat{n},s) &= \int_0^s \kappa_\lambda^{\mathrm{ext}}(\vec{r}_0+s'\hat{n})\,ds' \ .
\end{aligned}
$$

$$
\begin{aligned}
\eta_\lambda(\vec{r},\hat{n}) &= \kappa_\lambda^{\mathrm{abs}}(\vec{r})\,B_\lambda\big(T(\vec{r})\big) + \\
&+ \kappa_\lambda^{\mathrm{sca}}(\vec{r})\frac{1}{4\pi}\iint g_\lambda(\vec{r},\hat{n}',\hat{n})\,I_\lambda(\vec{r},\hat{n}')\,d^2\widehat{n}' \ .
\end{aligned}
$$

The real stuff

That's not all ...

- We integrate the RT equation for all pertinent rays (as you know the scattering term si not your friend)
- and then we solve for the Radiative Equilibrium (RE) condition

$$4\pi \int_0^\infty \kappa_\lambda^{\mathrm{abs}}(\vec{r}) B_\lambda\big(T(\vec{r})\big)\, d\lambda \quad = \quad 4\pi \int_0^\infty \kappa_\lambda^{\mathrm{abs}}(\vec{r}) J_\lambda(\vec{r})\, d\lambda \ .$$

- From this, we can compute the temperature (for dust, electrons or whatever)
- the medium emissivity depends on the radiation, and the radiation is determined from the emissivity : we need to Λ-iterate.

The real stuff

Let's get a bit nostalgic ...

- A long time ago (but in this galaxy, not a far far away one ...), I was a Monte Carlo kind of guy :
  - I was in Nice and I developed *McTrancf* : a 2D radiative tranfer code based on the Monte Carlo method (Niccolini et al. 2003) and then I got bored, so ...
  - I was in Madrid and I developed *Tapas* : a 3D RT code using the Monte Carlo algorithm (Niccolini & Alcolea 2006) and I got bored again ...
  - All this is under a huge amount of dust I'm afraid.
  - Oh boy, how I hated model fitting with such tools ...
  - It was not uncommon at the time to see people publishing χ^2 by eye type of results.

What do we need a tool such as Fracs for ?

- We want to determine physical quantities characterising environments (not necessarily “circumstellar”): densities, temperatures, dust grains, morphology, sizes ...
- You can use self-consistent radiative transfer tools to compute you favourite observable (e.g. Hdust from A. Carciofi, a Monte Carlo code is used in our lab)
- However, it is time consuming (a few 10^{th} of minutes to a few hours) and model fitting is a tricky business in such cases.
- You can also use toy models, which is ok but you cannot get deep into the physics of you object.

A simpler approach

- So, I entered Armando and Philippe’s office and proposed : “Bon les mecs, on va faire des trucs simples ...” (nothing is simple).
- We focused on efficiency.

What do we need a tool such as Fracs for ?

- We want to determine physical quantities characterising environments (not necessarily "circumstellar"): densities, temperatures, dust grains, morphology, sizes ...
- You can use self-consistent radiative transfer tools to compute you favourite observable (e.g. Hdust from A. Carciofi, a Monte Carlo code is used in our lab)
- However, it is time consuming (a few 10^{th} of minutes to a few hours) and model fitting is a tricky business in such cases.
- You can also use toy models, which is ok but you cannot get deep into the physics of you object.

A simpler approach

- So, I entered Armando and Philippe's office and proposed : "Bon les mecs, on va faire des trucs simples ..." (nothing is simple).
- We focused on efficiency.

Fracs (old version)

Physical assumptions

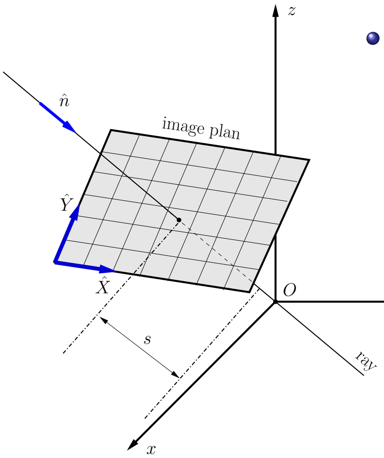
- We neglect the nasty scattering term in the RT equation.
- We do not compute the temperature structure but we prescribed it (basically, we have removed what make the transfer difficult).
- Since we were interested in a disc configuration, we assumed axisymmetry.
- Then, we integrate the RT equation (ray-tracing) along each rays (long characteristic) and we're done.
- Well, more or less . . .

Fracs (old version)

Numerical aspects

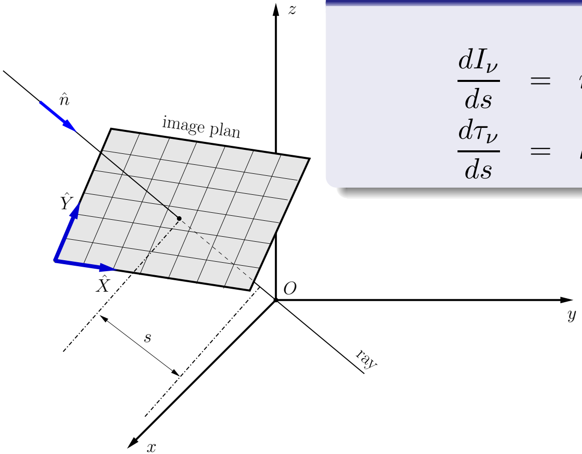
- A quadtree mesh
- Using left/right obvious symmetry.
- Not so obvious symmetry : from the upper part of the image, you can also obtain the lower part reconstructing one integral from the other $\rightarrow$ computating time is reduced by ~ 4.
- It's fast enough for model fitting : a few 10^{th} os second for an image at one wavelength.
- The main ingredient is the quadtree mesh and the point location algorithm (optimizing it you'll get faster)

Raytracing



- For each pixel we consider a ray going through the medium
- We need to integrate the radiative transfer equation along each ray

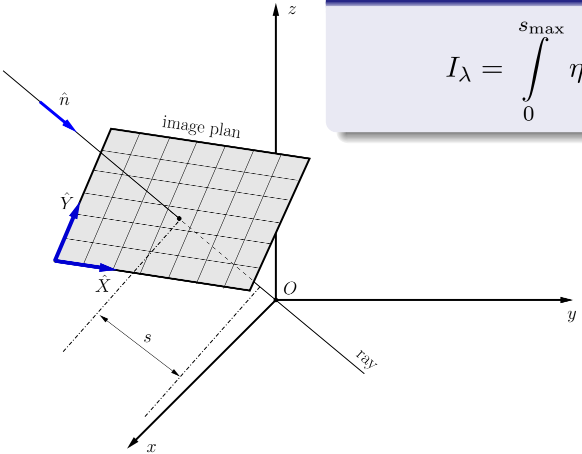
$$\hat{n}.\vec{\nabla} I_\lambda(\vec{r},\hat{n}) = -\kappa_\lambda\, I_\lambda(\vec{r},\hat{n}) + \eta_\lambda(\vec{r},\hat{n}) \ .$$



We can treat the RT eq. as an ODE:

$$\frac{dI_\nu}{ds} = \eta_\nu(\vec{r}_s)\,\mathrm{e}^{-\tau_\nu(s)} \,,$$
$$\frac{d\tau_\nu}{ds} = \kappa_\nu(\vec{r}_s) \,.$$

Raytracing



or as an integral

$$I_\lambda = \int_0^{s_{\max}} \eta_\lambda(s)\, \mathrm{e}^{-\tau_\lambda(s)}\, ds \ .$$

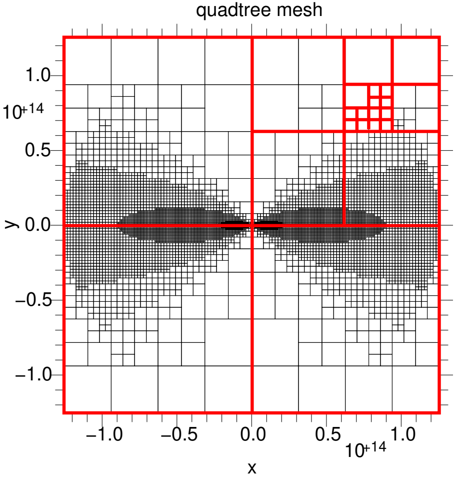
Quadtree mesh

- You put the computation effort where needed:
  - you save time,
  - you save memory.
- We used a quadtree mesh (see Kurosawa & Hillier 2001, for a description) with cylindrical coordinates.
- Similar to Barnes & Hut (1986) tree code use in many body problem.
- We divide recursively each cell in four child cells until e.g.

$$\frac{\iiint\limits_{V_\xi} [\kappa_\lambda(\vec{r})]\ d^3\vec{r}}{\iiint\limits_{V_{\rm tot}} [\kappa_\lambda(\vec{r})]\ d^3\vec{r}} \quad < \quad \eta\ . \tag{1}$$

- N.B: This type of criterion is used in the literature but it is not the best one I think (I'll get back to this latter).

Quadtree mesh



N.B. the cells are tori.

- This is a rough grid ($\eta \sim 10^{-3}$) to avoid representation problems
- Monte Carlo volume integration
- Uninteresting comment : coding this type of mesh is actually fun

Example: the wind model of a B[e] star

- The density is described as

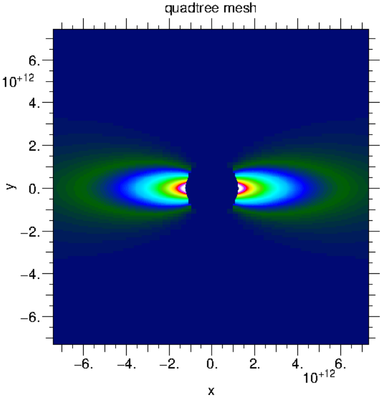
$$n(r,\theta) = n_{\rm in}\left(\frac{R_{\rm in}}{r}\right)^2 \frac{1+A_2}{1+A_1}\,\frac{1+A_1\ (\sin\theta)^m}{1+A_2\ (\sin\theta)^m}\ ,$$

- Dust grain opacities are computed from the Mie theory
- Dust temperature is parameterised by

$$T(r) = T_{\rm in}\left(\frac{R_{\rm in}}{r}\right)^{\gamma}\ .$$

- Emission from the central source

$$I^{\rm s}_{\lambda} = I^{\rm s}_{\lambda_0}\left(\frac{\lambda_0}{\lambda}\right)^{\alpha}\ .$$

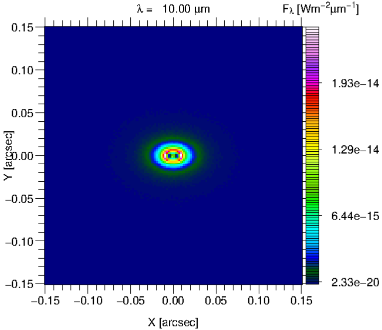


Model vs model

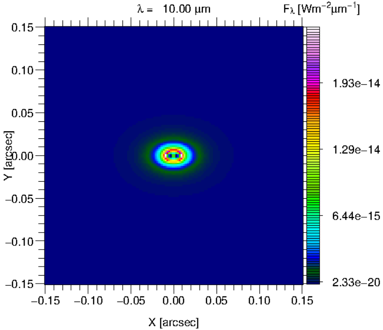
Parameters	Values	Unit
A_1	150	-
A_2	-0.8	-
R_{in}	30	R_{s}
n_{in}	0.015/0.15	m^{-3}
T_{in}	1500	K
γ	0.75	-
$I^{\mathrm{s}}_{\lambda_0}$	6500	$\mathrm{W\,m^{-2}\,\mu m^{-1}\,str^{-1}}$
α	3	-
$\mathrm{PA_d}$	125	deg
i	20/50/90	deg
a_{min}	0.5	μm
a_{max}	50	μm
β	-3.5	-

- 10 free parameters
- red : better than 10 %,
- blue : 10 to 25 %,
- green : larger than 25 %, less than 100 %,
- gray : larger than 100 %.

Monte Carlo code vs ray-tracer (unfair comparison !)



Monte Carlo (Tapas, $\sim$ 3hours)

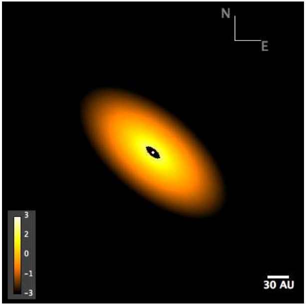


Ray-tracer (Fracs, $\sim$ 0.2 s)

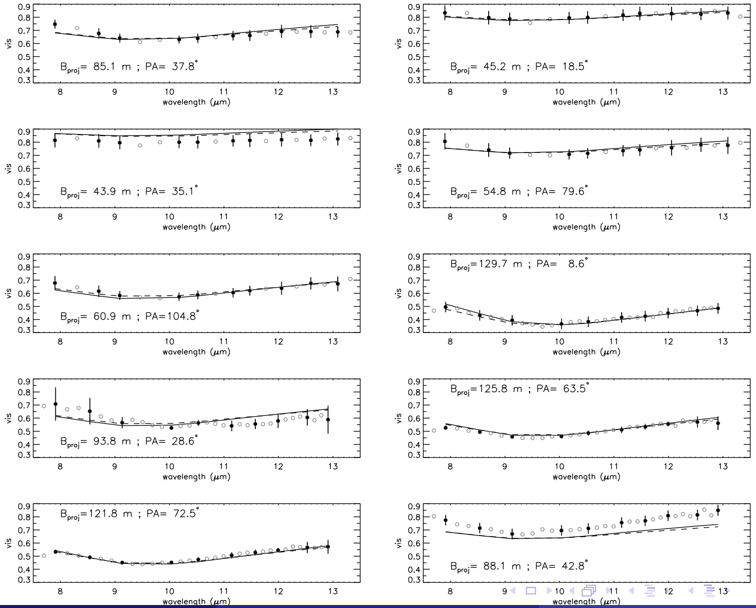
Domiciano de Souza et al. (2011)

$$n(r,\theta) = \begin{cases} n_{\rm in} \left(\frac{R_{\rm in}}{r}\right)^2 & ;\, 90^\circ - 0.5\Delta\theta_{\rm d} \leq \theta \leq 90^\circ + 0.5\Delta\theta_{\rm d} \\ 0 & ;\, \theta < 90^\circ - 0.5\Delta\theta_{\rm d} \text{ and } \theta > 90^\circ + 0.5\Delta\theta_{\rm d} \end{cases}$$

Adopted distance	$d = 1.7$ kpc		$d = 2.5$ kpc	
Model parameters	value	error	value	error
$I^{\rm s}_{\lambda_0}$	2.1	$^{+0.1}_{-0.1}$	4.2	$^{+0.1}_{-0.1}$
α	2.4	$^{+0.2}_{-0.2}$	2.4	$^{+0.2}_{-0.2}$
γ	0.92	$^{+0.07}_{-0.07}$	0.85	$^{+0.05}_{-0.05}$
$R_{\rm in}$ (AU)	11.0	$^{+2.0}_{-2.0}$	13.9	$^{+2.4}_{-2.4}$
i (°)	60.5	$^{+1.5}_{-1.5}$	59.3	$^{+1.5}_{-1.5}$
$\mathrm{PA_d}$ (°)	139.8	$^{+1.0}_{-1.0}$	139.3	$^{+1.0}_{-1.0}$
$n_{\rm in}$ (m^{-3})	0.09	$^{+0.06}_{-0.06}$	0.11	$^{+0.07}_{-0.07}$
$\Delta\theta_{\rm d}$ (°)	7.5	$^{+4.4}_{-4.4}$	5.9	$^{+4.4}_{-4.4}$



VLTI/MIDI visibilities



Outline

Why we decided to write a new version

- New 3D mesh with neighbour search and more efficient tree traversal.
- free-free and bound-free continuum opacities
- line emission
- New integrator
- It should be flexible (adding new kind of features or new physics must be simple)
- MATISSE data are here !
- The program should be controled from python
  - This will simplifies post-processing (no need to communicate via config files)
  - We want to use emcee, the Markov chain Monte Carlo python package (Foreman-Mackey et al. 2013).
  - boost::python to the rescue.

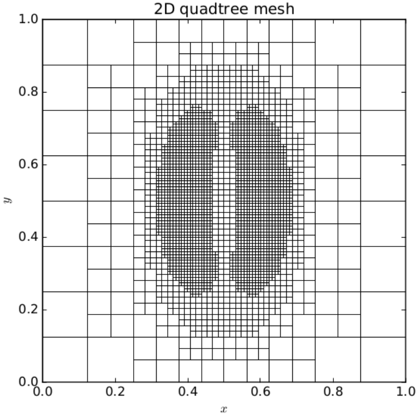
A new mesh implementation

Features

- The algorithm is described in Frisken & Perry (2002)
- Arbitrary number of dimensions (we use octrees)
- The algorithm uses binary locational codes (represent the position of the lower vertices)
- The tree is traversed following these codes.
- It allows a very efficient way to determine cell neighbours
- I fully implemented the mesh, as well as related tools (builder, smoother, writer, etc . . .)

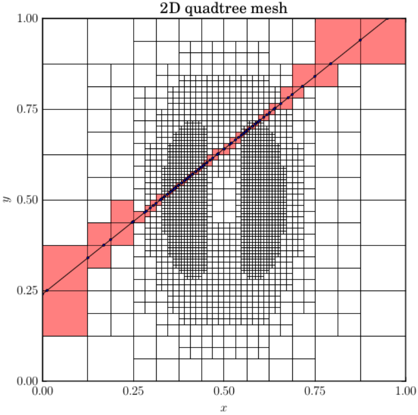
A new mesh implementation

Example : a quadtree refined on the gradient of a Gaussian + smoothing.



A new mesh implementation

Raytracer : give it a point $\vec{r}$ to start from and a direction $\hat{n}$ and it'll do its job.



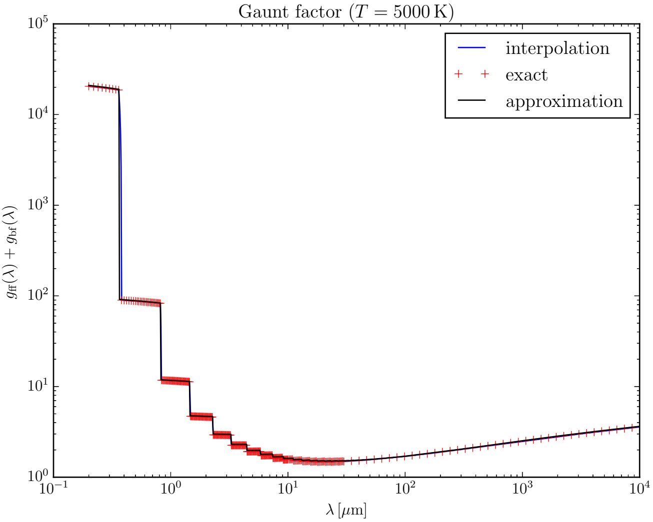
Continuum opacities

- Dust opacities are already implemented (using the Mie theory).
- Free-free and bound-free absorption coefficients

$$\kappa_{\mathrm{ff+bf}}(\lambda,T) = \frac{4e^6}{3m_{\mathrm{e}}^2hc(4\pi\epsilon_0)^3}\left(\frac{2\pi m_{\mathrm{e}}}{3k}\right)^{\frac{1}{2}}\frac{Z^2n_{\mathrm{e}}\,n_{\mathrm{i}}}{T^{\frac{1}{2}}}\left(\frac{\lambda}{c}\right)^3\left(1-\mathrm{e}^{-\frac{hc}{\lambda kT}}\right)\left(\overline{g_{\mathrm{ff}}}(\lambda,T)+g_{\mathrm{bf}}^{\mathrm{tot}}(\lambda,T)\right)$$

- The nasty and tricky part is the computation of the Gaunt factors (see Armando for more details).
- A paper has been recently submited using these opacties in a specific raytracer (V. Hocdé et al. on Cepheids).
- The code already exits in python (the equivalent code in c++ is on the way)

Continuum opacities



Continuum Integration

If we assume

$$\kappa_\nu(s) \approx \kappa_\nu^i + \frac{\kappa_\nu^{i+1} - \kappa_\nu^i}{\Delta s_i}(s - s_i)$$

$$S_\nu(s) \approx S_\nu^i \left(\frac{S_\nu^{i+1}}{S_\nu^i}\right)^{\frac{s-s_i}{\Delta s_i}}$$

The optical depth is then quadratic

$$\Delta\tau_\nu(s) \approx \kappa_\nu^i(s - s_i) + \frac{1}{2}\frac{\Delta\kappa_\nu^i}{\Delta s_i}(s - s_i)^2 \ ,$$

Continuum Integration

The integral for the intensity in one cell is given by

$$\Delta I_\lambda^i = S_\lambda^i \, \mathrm{e}^{-\tau_\lambda^i} \int\limits_0^{\Delta s_i} \left[\kappa_\lambda^i + \frac{\Delta \kappa_\lambda^i}{\Delta s_i} \, s \right] \mathrm{e}^{\left[\frac{1}{\Delta s_i} \ln\left(\frac{S_\lambda^{i+1}}{S_\lambda^i} \right) - \kappa_\lambda^i \right] s - \frac{1}{2} \frac{\Delta \kappa_\lambda^i}{\Delta s_i} s^2} \, ds \ .$$

Remembering that we want to compute

$$I_\lambda = \int\limits_0^{s_\mathrm{max}} \kappa_\lambda(s) S_\lambda(s) \, \mathrm{e}^{-\tau_\lambda(s)} \, ds \ .$$

Continuum Integration

It can be simplified in

$$\Delta I_\nu^i \approx S_\nu^i \, \mathrm{e}^{-\tau_\nu^i} \int\limits_0^{\Delta s_i} \left[\kappa_\nu^i + \frac{\Delta \kappa_\nu^i}{\Delta s_i} \, s \right] \mathrm{e}^{\left[\frac{1}{\Delta s_i} \ln\left(\frac{S_\nu^{i+1}}{S_\nu^i} \right) - \bar{\kappa}_\nu^i \right] s} \, ds \; ,$$

with

$$\bar{\kappa}_\nu^i = \frac{\kappa_\nu^{i+1} + 2\,\kappa_\nu^i}{3} \; .$$

up to $O\left(\Delta s_i^4\right)$ terms. This integral can be done analytically (similar trick in Woitke et al. (2009) but not of the highest order)

Line emission

The basics

- A spectral line is characterised by the absorption coefficient and its emissivity:

$$\begin{aligned} \kappa_\nu &= n_l\,\sigma_{ul}\,\Phi_{ul}(\nu) \\ \eta_\nu &= \kappa_\nu\,B_\nu(T_{\rm ex})\ . \end{aligned}$$

- The cross section (corrected for stimulated emission) σ_{ul} is given by:

$$\sigma_{ul} = \frac{c^2\,A_{ul}}{8\pi\nu_{ul}^2}\,\frac{g_u}{g_l}\,\left(1 - {\rm e}^{-\frac{h\nu_{ul}}{kT_{\rm ex}}}\right)\ ,$$

where A_{ul} is the *Einstein* coefficient for spontaneous emission.

Line emission

Line profile

- The line profile $\Phi_{ul}(\nu)$ describing Doppler broadening due to thermal motion and microturbulence is

$$\begin{aligned}\Phi_{ul}(\nu) &= \frac{c}{\sqrt{\pi}\, b\, \nu_{ul}} \mathrm{e}^{-\frac{1}{b^2}\left[\frac{c}{\nu_{ul}}(\nu_{ul}-\nu)-\vec{v}.\hat{n}\right]^2} \\ b^2 &= \frac{2kT_{\mathrm{kin}}}{m_{\mathrm{mol}}} + v_{\mathrm{turb}}^2 \,.\end{aligned}$$

- That stuff is just the *Maxwellian* distribution in disguise.
- $\frac{c}{\nu_{ul}}(\nu_{ul} - \nu)$ is the velocity at which we observe.
- $\vec{v}.\hat{n}$ is the projection of the macroscopic velocity of the gas along the line of sight.

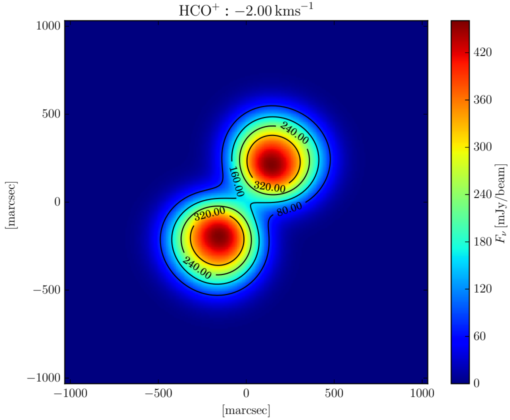
Along the ray

- When we integrate, ν (or the velocity of observation) is fixed and we integrate in space.
- Remember: $\Phi_{ul}(\nu) = \frac{c}{\sqrt{\pi}\, b\, \nu_{ul}} \mathrm{e}^{-\frac{1}{b^2}\left[\frac{c}{\nu_{ul}}(\nu_{ul}-\nu)-\vec{v}.\hat{n}\right]^2}$
- And $\vec{v} = \vec{v}(s)$, s being the distance along the ray.
- The size is roughly :

$$\Delta s \approx \frac{b}{\mid \hat{n}.\frac{d\vec{v}}{ds}(s) \mid}.$$

Example : IRAS 15103-5754

A specific raytracer for lines (written in python)



- Water-fountain observed with ALMA
- Emission of a toroidal structure
- Gómez et al. (see 2018)

Outline

Let's recap . . .

- Fracs is a raytracer.
- Project started in 2008
- Reborn (prematurely) in 2015 (mesh, raytracer).
- Complete rewrite
- Now working on continuum opacities (including gas)
- I expect to have a working version "quite soon" (at least for the continuum) . . .
- Post-processing is on the way as well.
- For the lines, I expect to work again on water fountains with my IAA colleagues. It is a good playground.

For further reading . . .

Barnes, J. & Hut, P. 1986, Nature, 324, 446

Domiciano de Souza, A., Bendjoya, P., Niccolini, G., et al. 2011, A&A, 525, A22

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306

Frisken, S. F. & Perry, R. N. 2002, Journal of Graphics Tools, 7, 1

Gómez, J. F., Niccolini, G., Suárez, O., et al. 2018, MNRAS, 480, 4991

Kurosawa, R. & Hillier, D. J. 2001, A&A, 379, 336

Niccolini, G. & Alcolea, J. 2006, A&A, 456, 1

Niccolini, G., Bendjoya, P., & Domiciano de Souza, A. 2011, A&A, 525, A21

Niccolini, G., Woitke, P., & Lopez, B. 2003, A&A, 399, 703

Woitke, P., Kamp, I., & Thi, W.-F. 2009, A&A, 501, 383