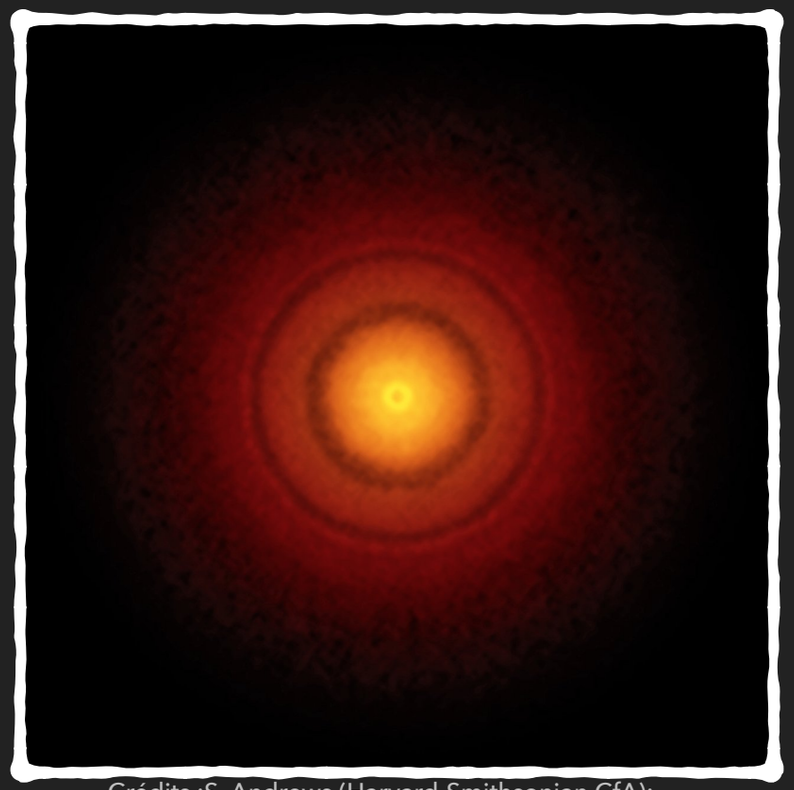


Crédits :S. Andrews (Harvard-Smithsonian CfA);
B. Saxton (NRAO/AUI/NSF); ALMA (ESO/NAOJ/NRAO)

Maxime Lombart
Supervisor: Guillaume Laibe

SMOLUCHOWSKI EQUATION AND PROTOPLANETARY DISCS

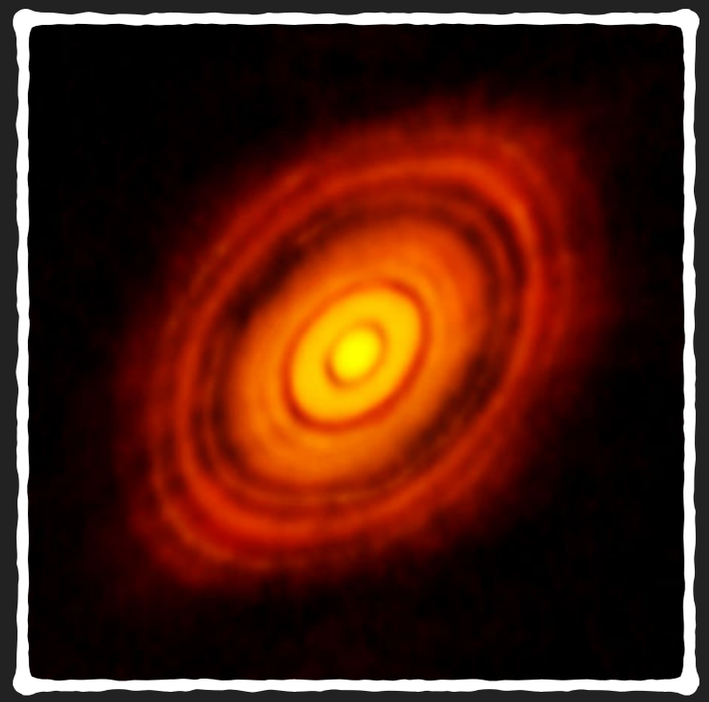
Workshop disc-ussions, 07/19/2019, Monash University

Planetary formation

~8pc

NASA, ESA, M. Robberto (Space Telescope Science Institute/ESA) and the Hubble Space Telescope Orion Treasury Project Team

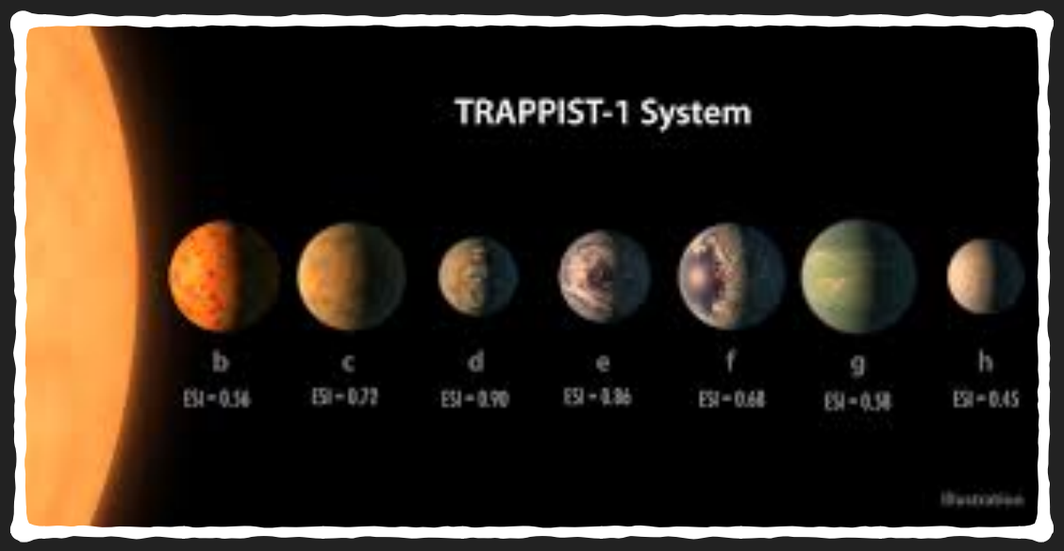
??



ALMA (ESO/NAOJ/NRAO)

~100AU

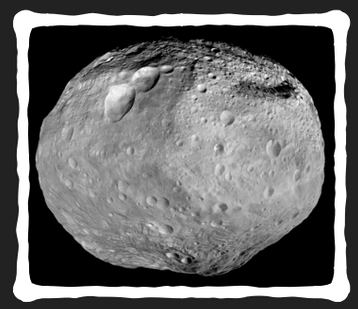
~10AU



NASA/R. Hurt/T. Pyle

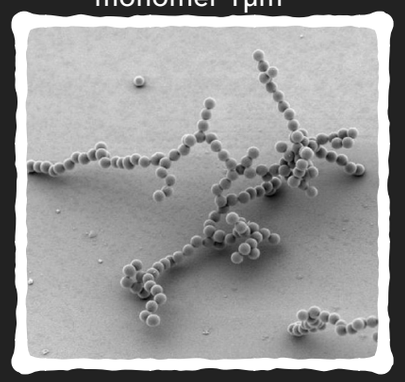
??

~1km



NASA/JPL-Caltech/UCAL/MPS/DLR/IDA

monomer 1μm



Blum et al. 1999

??

Coagulation

- Coagulation : aggregation of monomers into polymers
- Physical process in many fields

chemistry	→	polymerisation
biology	→	proteins growth
meteorology	→	raindrops formation
astrophysics	→	grains growth

Marian Smoluchowski

- The coagulation process is described mathematically by the **Smoluchowski equation** (1916).
- Safronov (1969), Friedlander (1977), Silk & Takahashi (1979), Dubovskii (1994), Aldous (1999), Lee (2001), Filbet & Laurencot (2003), Filbet (2008), ...

Smoluchowski equation

- Model assumption : coalescence occurs instantly after the collision of two particles, a new sphere is formed.

n: number density of polymers
K: collision frequency (Kernel)

$K_{k,j} = \sigma_{k,j} \Delta v_{k,j}$

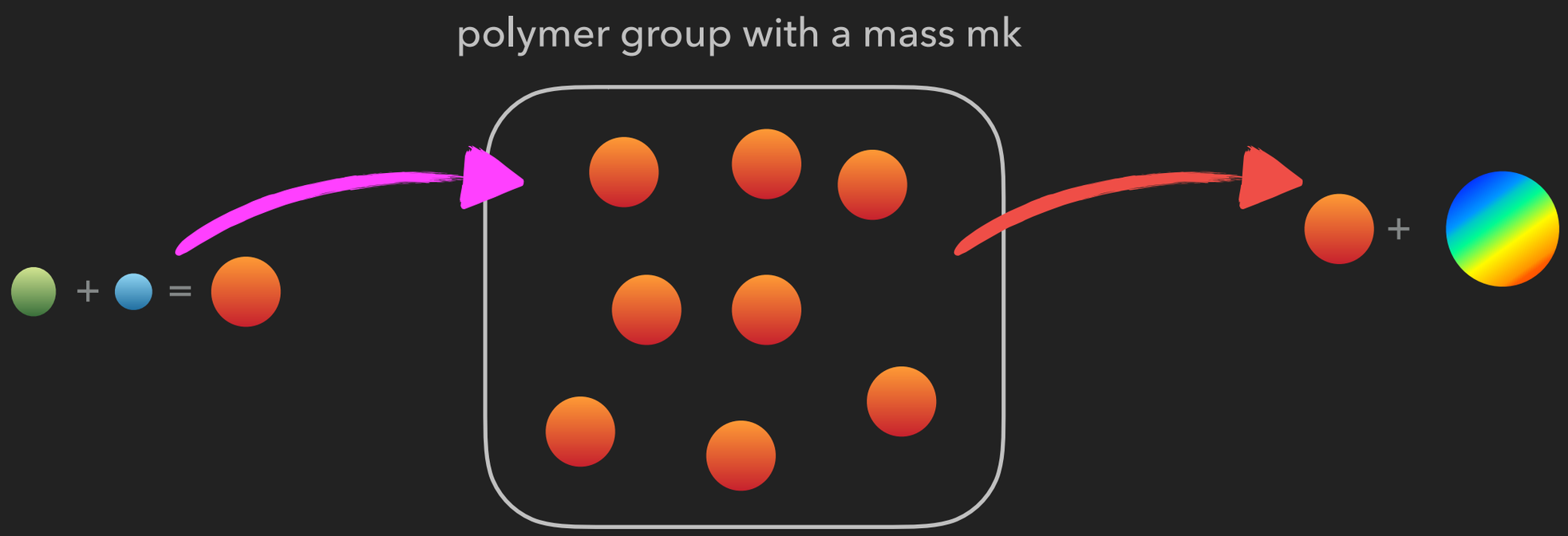
$$\frac{dn_k(t)}{dt} = \frac{1}{2}\sum_{j=1}^{k-1} K_{j,k-j} n_j(t) n_{k-j}(t) - n_k(t) \sum_{j=1}^{\infty} K_{k,j} n_j(t)$$

$$\frac{\partial n(x,t)}{\partial t} = \frac{1}{2}\int_0^x K(y, x-y) n(y,t) n(x-y,t) dy \; - \; n(x,t) \int_0^{\infty} K(x,y) n(y,t) dy$$

$m_k = k \times m_{min}$

x,y: masses

Mass conservation equation.



Smoluchowski equation

$$\frac{dn_k(t)}{dt} = \frac{1}{2}\sum_{j=1}^{k-1} K_{j,k-j} n_j(t) n_{k-j}(t) \; - \; n_k(t) \sum_{j=1}^{\infty} K_{k,j} n_j(t)$$

$$\frac{\partial n(x,t)}{\partial t} = \frac{1}{2}\int_0^x K(y, x-y) n(y,t) n(x-y,t) dy \; - \; n(x,t) \int_0^\infty K(x,y) n(y,t) dy$$

- Non linear integro-differential equation.
- No analytical solution in general case.
- Physical constrains : mass conservation, positivity.
- Astrophysics constrains : any kernels.

Nintendo ™

Requirements from PHANTOM

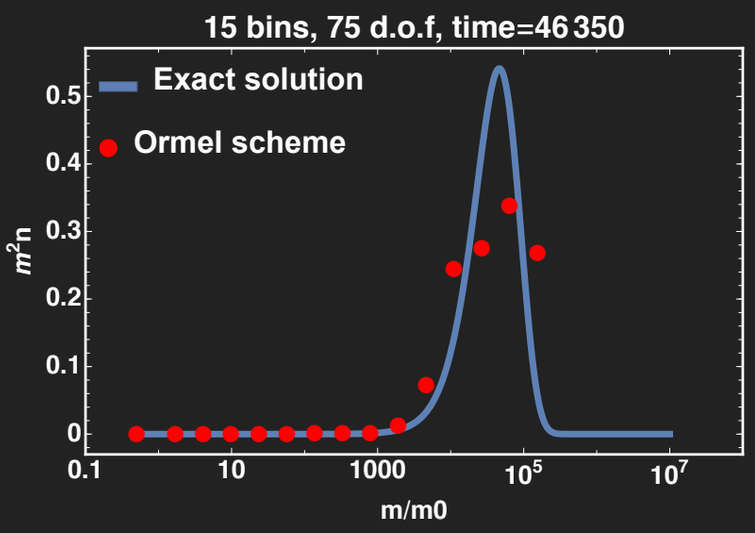
issue : need to have a fast and efficient algorithm.

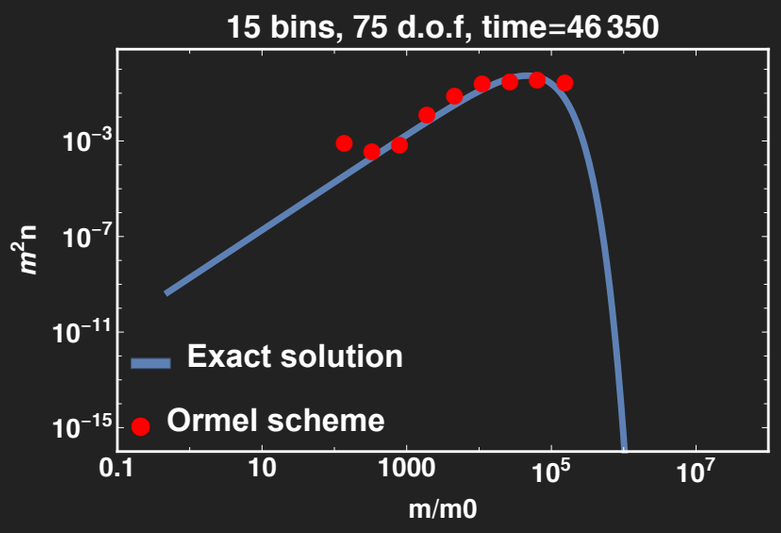
Requirements \ Algorithms	Ormel et al. 2007	Brauer et al. 2008	Finite volume scheme
▸ SPHERE/ALMA 12 orders of magnitude in mass < 15 grain sizes ▸ ~10-100 degrees of freedom: sub-bins or MC particles ▸ Good accuracy	?	?	?

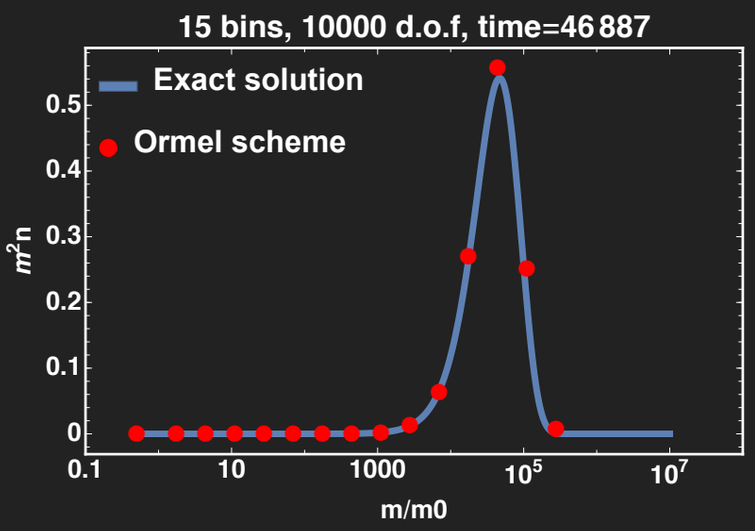
- Analytical solution with simple kernels.
- To benchmark algorithms (tests with constant kernel).

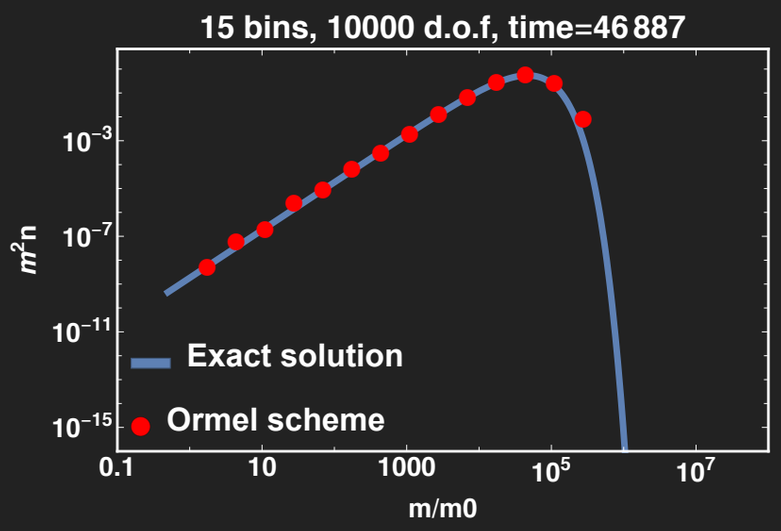
Monte-Carlo algorithm (Ormel et al. 2007)

$$\int mn(m,t)\mathrm{d}m = \int m^2 n(m,t)\mathrm{d}(ln(m))$$

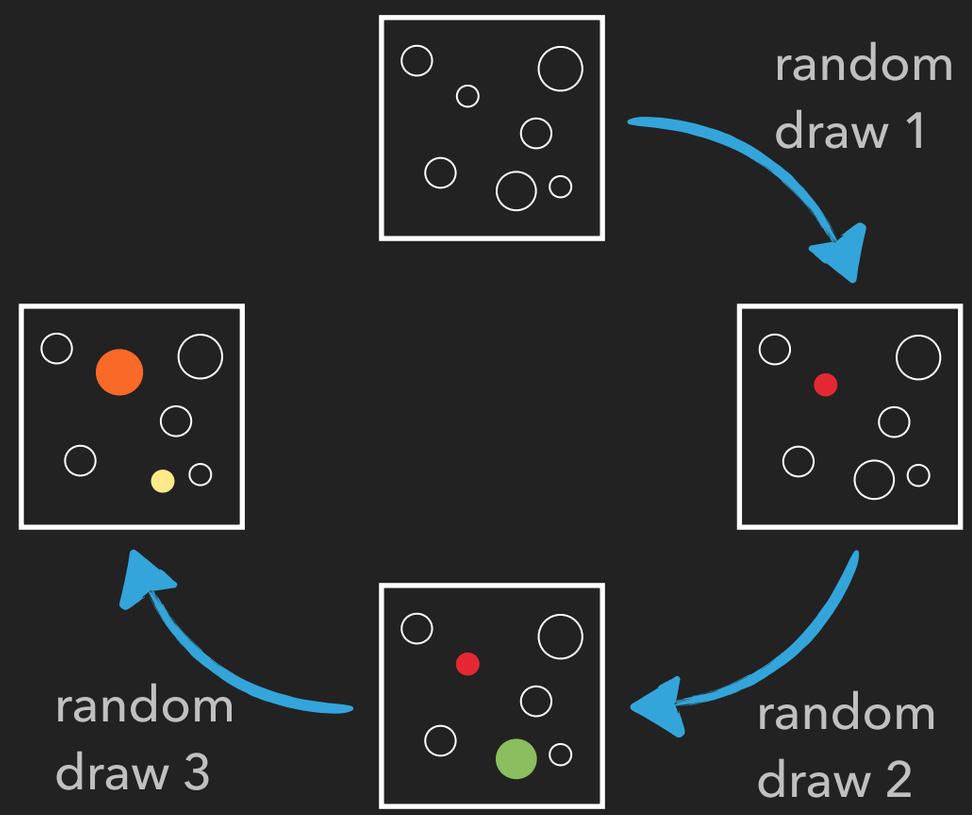








Probability density function of collisions $P(t,i,j)$

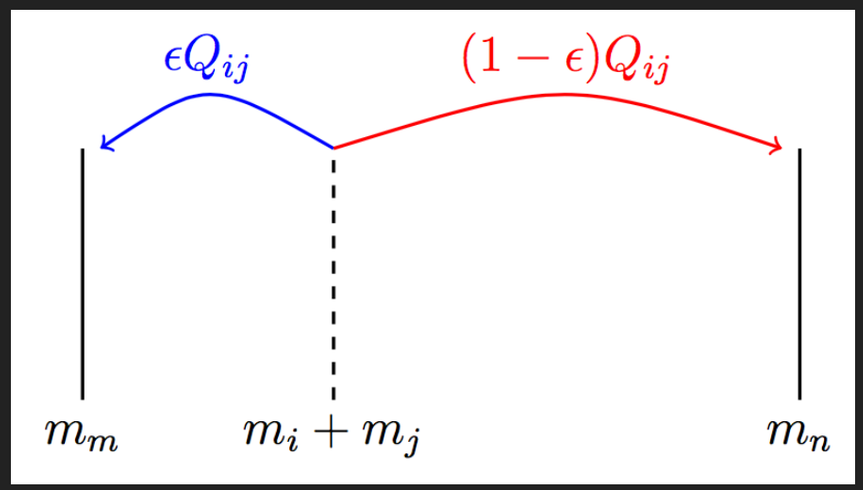


- Accuracy for a large number of particles → long calculation time
- Will be useful to compare with non simple kernels.

Brauer et al. 2008 algorithm

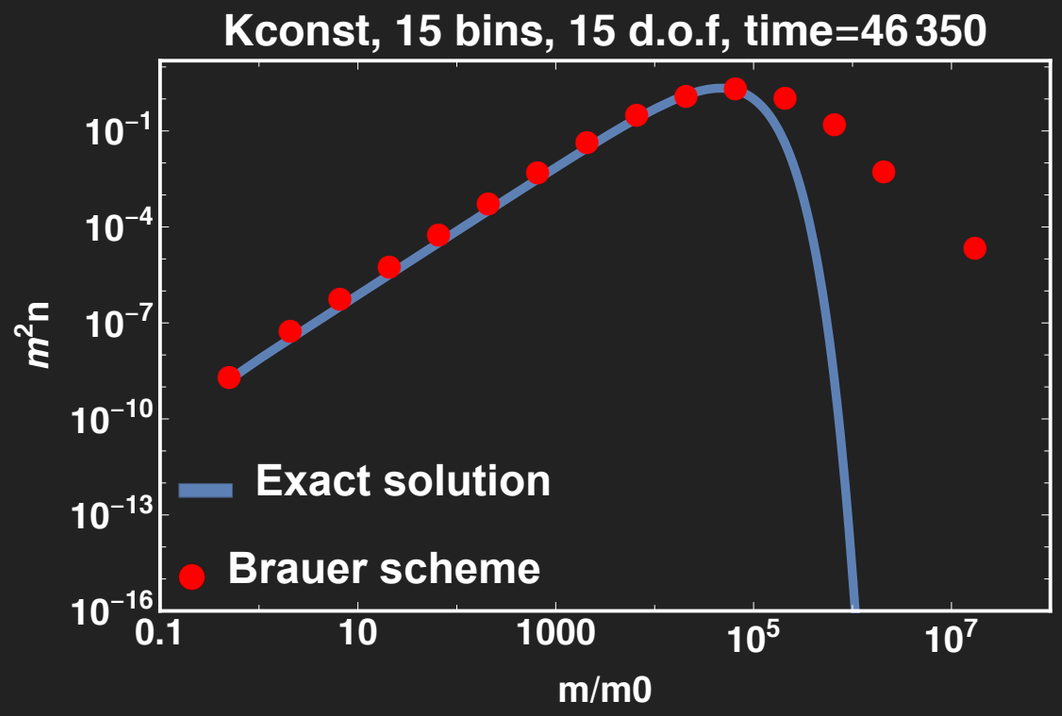
- Kovetz, Olund 1969, Brauer et al. 2008, Birnstiel et al. 2012

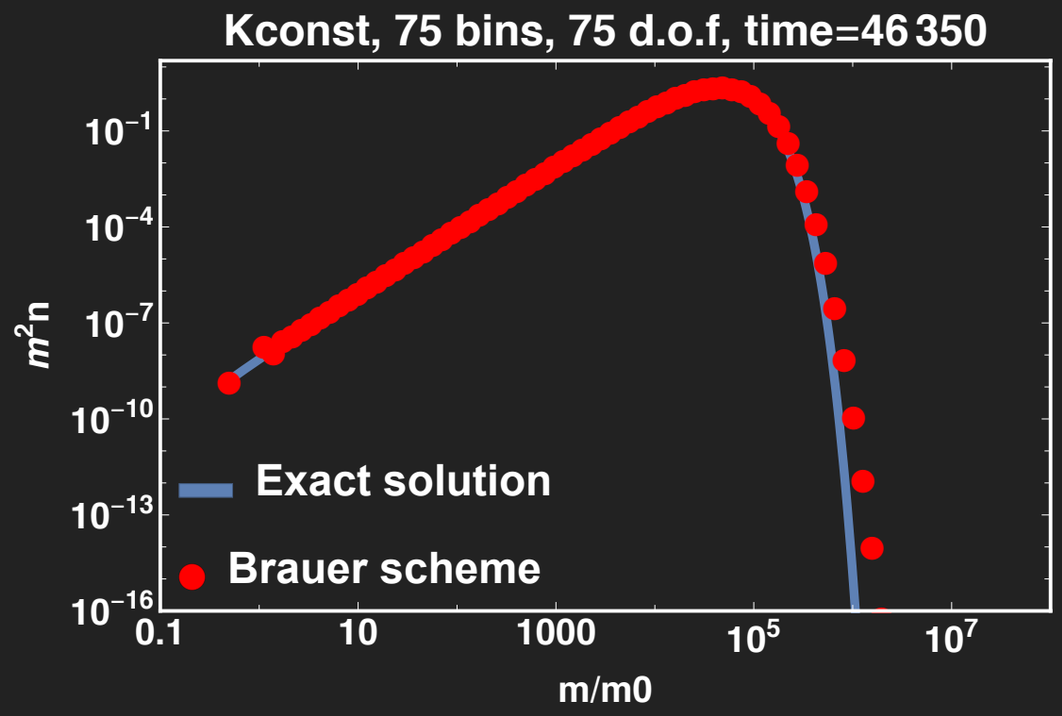
$$\frac{dn_k(t)}{dt} = \frac{1}{2}\sum_{j=1}^{k-1} C_{j,k-j} K_{j,k-j} n_j(t) n_{k-j}(t) - n_k(t) \sum_{j=1}^{N} K_{k,j} n_j(t)$$

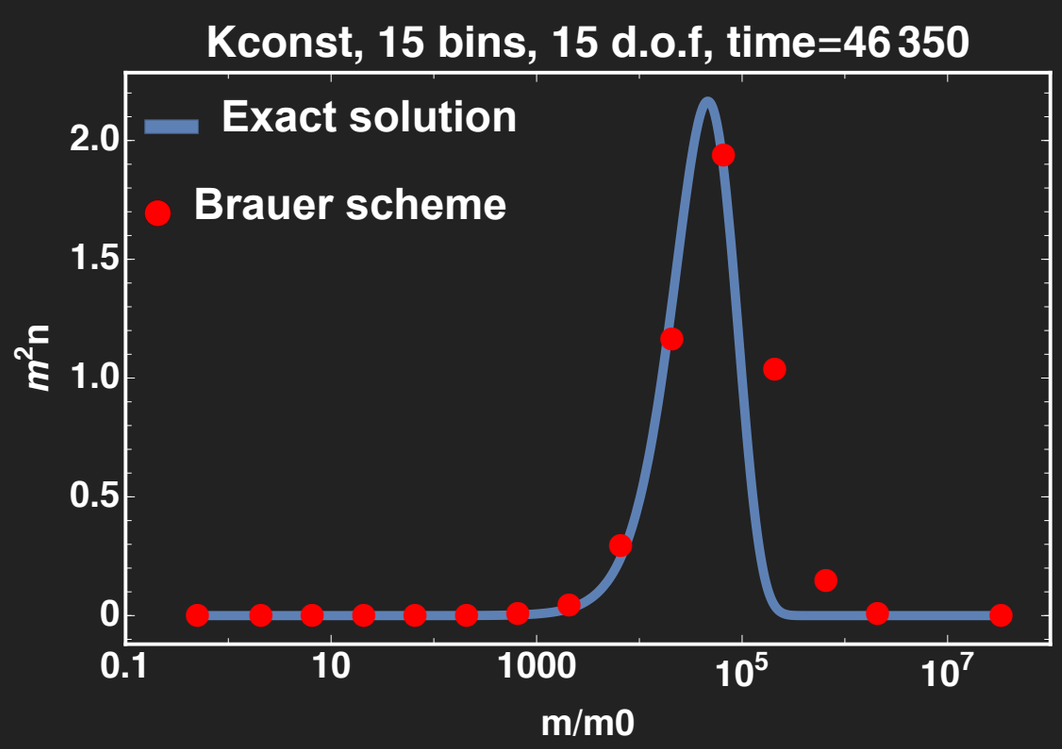


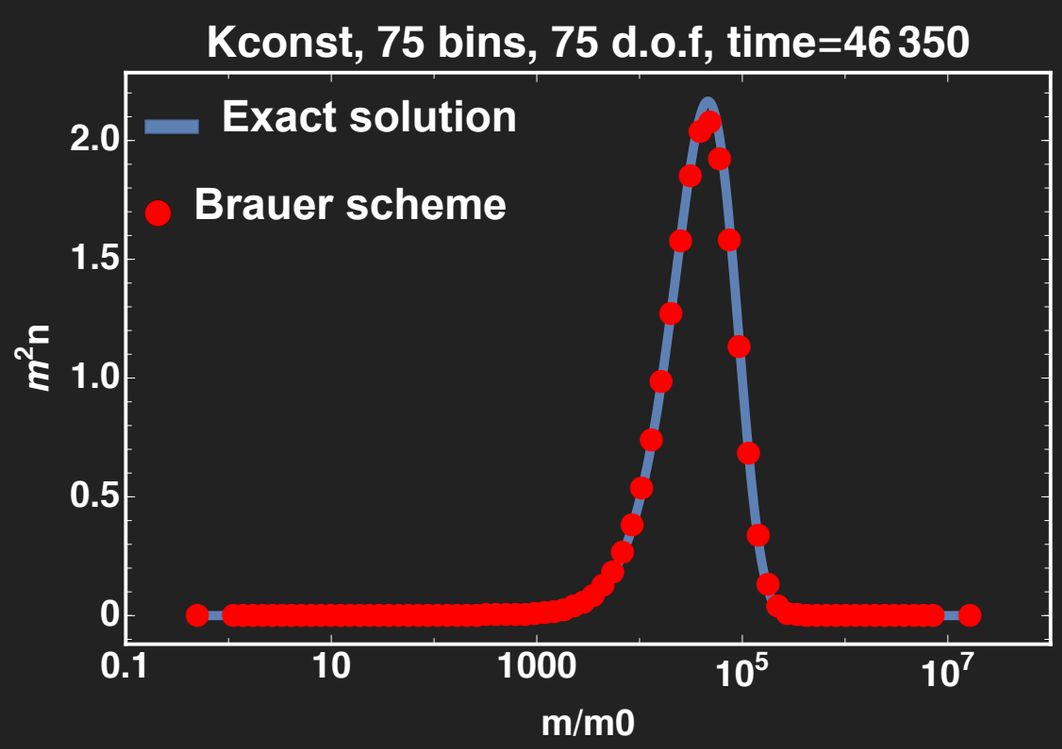
Scheme of the mass distribution of the polymer formed by the collision of i and j.

Brauer et al. 2008 algorithm









$$\int mn(m,t)\mathrm{d}m = \int m^2 n(m,t)\mathrm{d}(ln(m))$$

Good accuracy is not checked with 15 grain sizes.

Finite volume method

- Equation in conservative form (Tanaka et al. 1996, Filbet & Laurençot 2003, Filbet 2008, Tine 2011, Liu 2019 ...).
- Hyperbolic equation → mass conservation.

$$\partial_t g(x,t) = -\partial_x F_{coag}[g](x,t)$$

$$F_{coag}[g](x,t) = \int_0^x \int_{x-u}^{\infty} \frac{K(u,v)}{v} g(u,t)\, g(v,t)\, \mathrm{d}u\mathrm{d}v$$

$$F_{coag}[g](x,t) = \int_0^x \int_{x-u}^{R-u} \frac{K(u,v)}{v} g(u,t)\, g(v,t)\, \mathrm{d}u\mathrm{d}v$$ F=0 at x=R

High-order algorithm

- How can we obtain the accuracy required by PHANTOM with at maximum 15 grain sizes ?
  Discontinuous Galerkin method (Liu et al. 2019).
- Accuracy is controlled simply.
- Absorption of discontinuities at interfaces.

★ $\partial_t g(x,t) + \partial_x F[g](x,t)) = 0$ → PDE

★ $\forall \phi \quad \int_{I_k} \partial_t g(x,t)\phi(\xi(x))dx + \int_{I_k} \partial_x F[g](x,t)\phi(\xi(x))dx = 0$

★ $\forall x \in I_k, \quad g(x,t) \approx g_k(x,t), \quad g_k(x,t) = \sum_i c_k^i(t)\,\phi_i(\xi_k(x))$ orthogonal polynomials

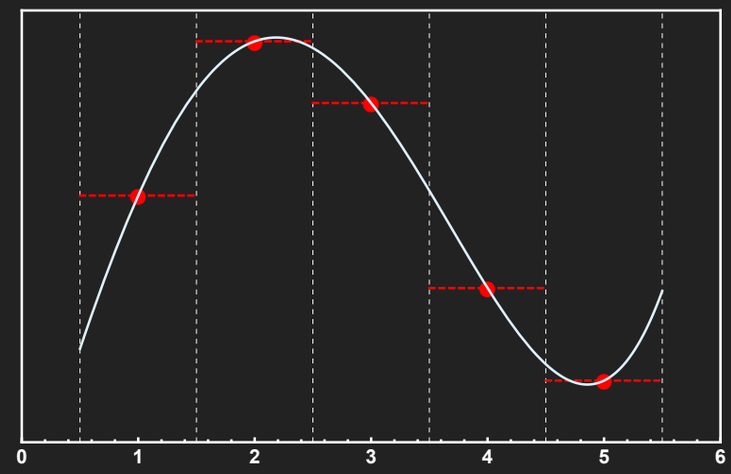
★ $\int_{I_k} \partial_t g(x,t)\phi(\xi(x))dx + F[g](x,t))\phi(\xi(x))\,|_{\partial I_k} - \int_{I_k} F[g](x,t)\partial_x\phi(\xi(x))dx = 0$

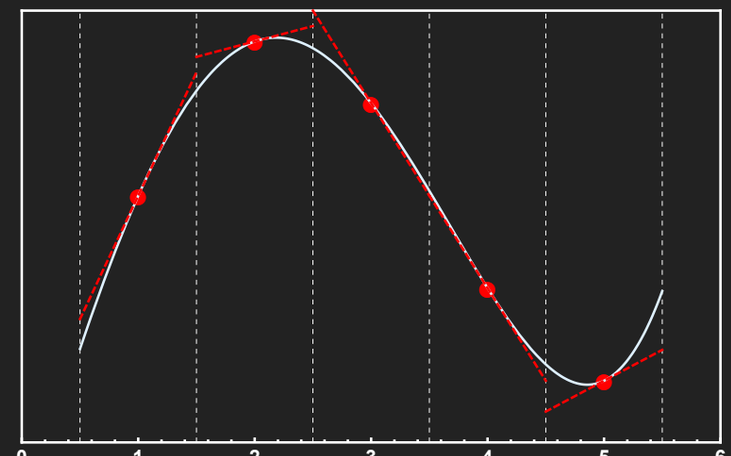
★ $\frac{\Delta x_k}{2}\int_{-1}^{1} \vec{\phi}(\xi)\vec{\phi}^T(\xi)d\xi \,.\, \frac{d\vec{c_k}(t)}{dt} + F[g_k](x,t))\vec{\phi}(\xi_k(x))\,|_{\partial I_k} - \int_{I_k} F[g_k](x,t))\partial_x\vec{\phi}(\xi_k(x))dx = \vec{0}$

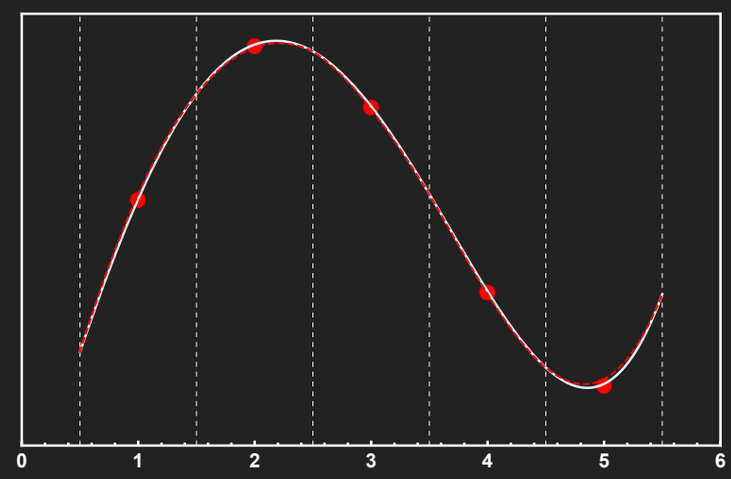
$\begin{bmatrix} d_0 & & \\ & \ddots & \\ & & d_p \end{bmatrix}$ → ODE

★ $\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} c_k^0 \\ \ddots \\ c_k^p \end{bmatrix}(t) = \frac{2}{\Delta x_k}\begin{bmatrix} 1/d_0 & & \\ & \ddots & \\ & & 1/d_p \end{bmatrix}\left(\int_{I_k} F(g_k(x,t))\partial_x \begin{bmatrix} \phi_0 \\ \ddots \\ \phi_p \end{bmatrix}(\xi_k(x))dx - F\left[g\right](x,t)\begin{bmatrix} \phi_0 \\ \ddots \\ \phi_p \end{bmatrix}\left(\xi_k(x)\right)|_{\partial I_k}\right)$

Brauer algorithm.



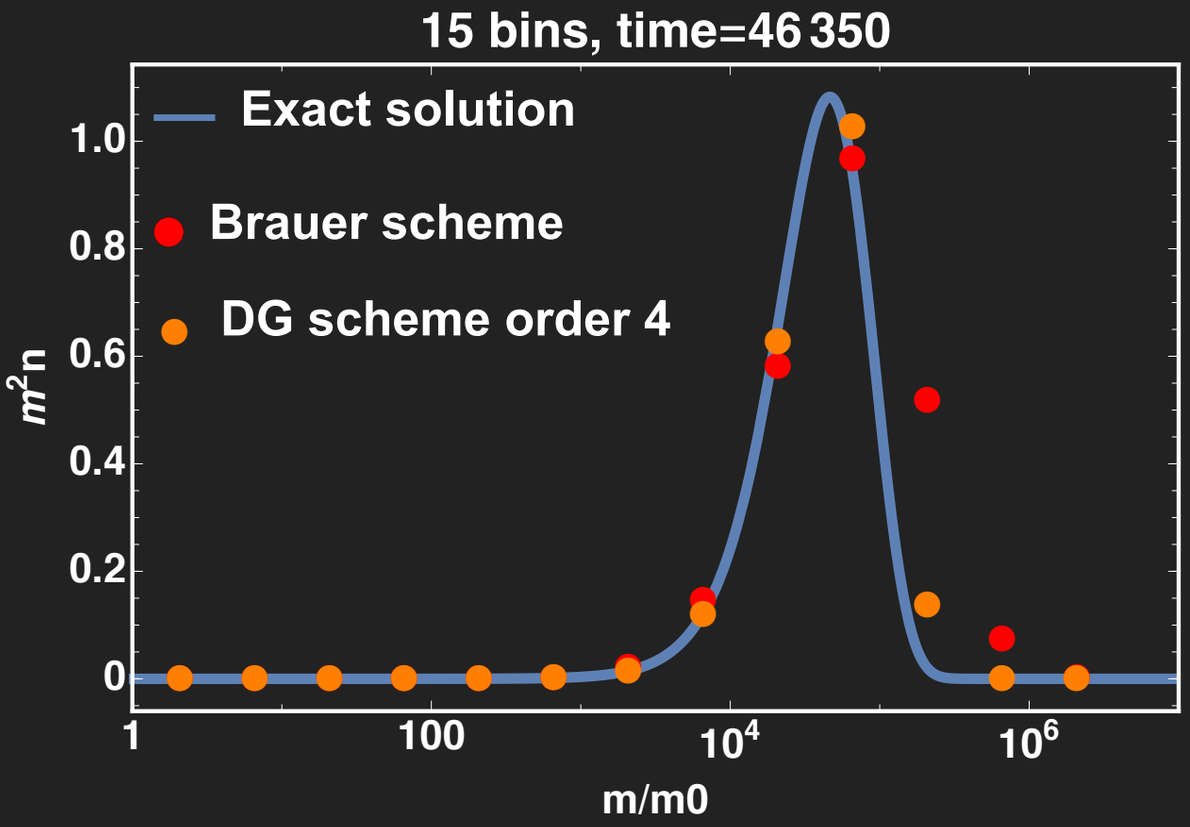


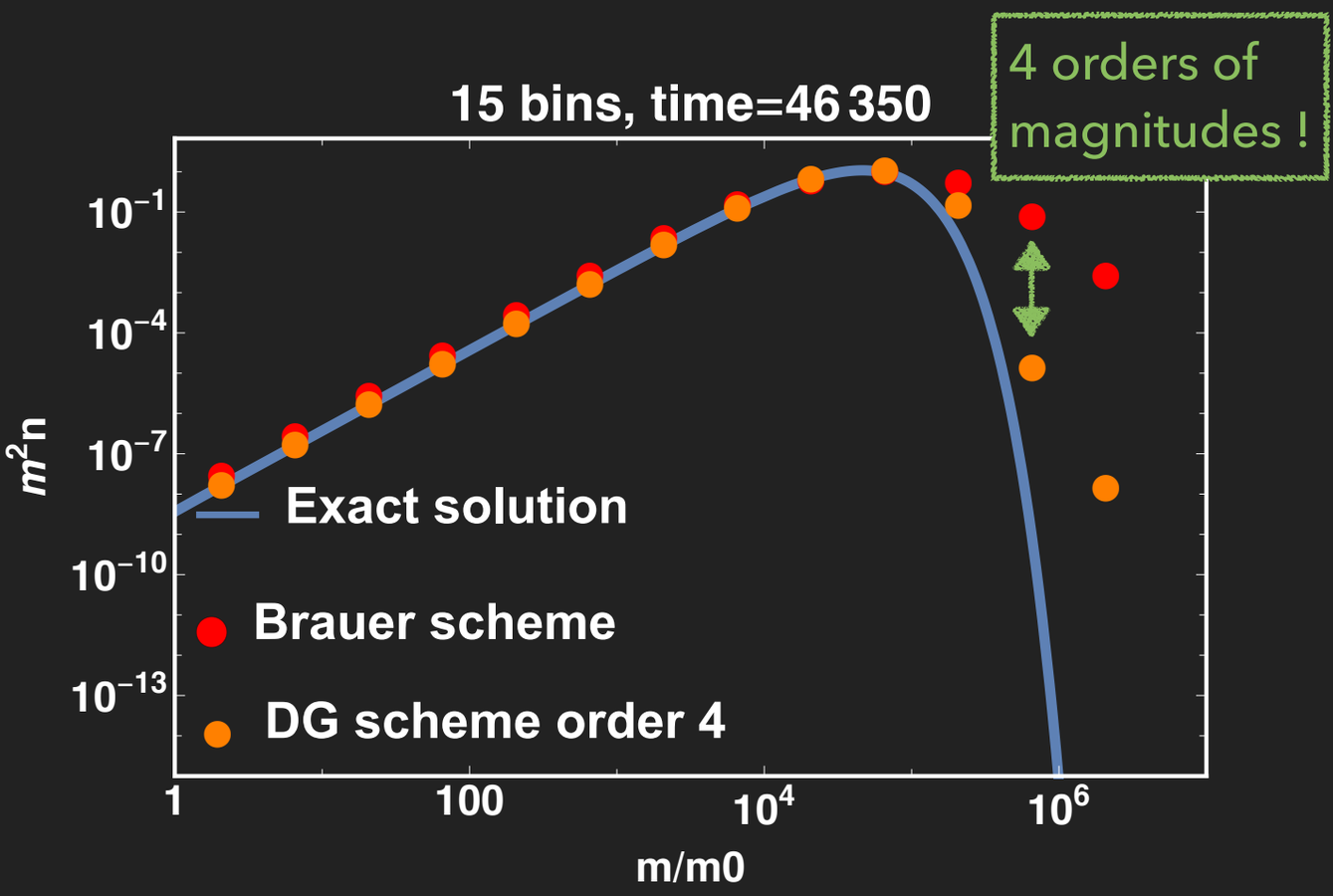


High-order algorithm

- Gauss quadrature method → integral approximation.
- **Discontinuous Galerkin + Gauss quadrature**

$$\int_a^b f(x)\omega(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$





$$\int mn(m,t)\mathrm{d}m = \int m^2 n(m,t)\mathrm{d}(ln(m))$$

Good candidate !

Requirements from PHANTOM

Requirements \ Algorithms	Ormel et al. 2007	Brauer et al. 2008	Finite volume scheme
• SPHERE/ALMA 12 orders of magnitude in mass < 15 grain sizes • ~10-100 degrees of freedom: sub-bins or MC particles • Good accuracy	?	✗	?

Optimisations

- Long time solution (Menon & Pego 2006)

$$K(x,y) = (xy)^{\lambda/2}, \quad 0 \leq \lambda < 2$$

$$\lim_{t \to +\infty} n(x,t) = \exp(-x)$$

- New polynomial basis

$$g_k(x,t) = \sum_i c_k^i(t)\tilde{\phi}_i(\xi_k(x),t)$$

$$\tilde{\phi}_i(\xi_k(x),t) = \exp\left(-\frac{a(t)\xi_k(x)}{2}\right) L_{a(t),i}(\xi_k(x))$$

$$\int_{-1}^{1} \exp(-a(t)\xi) L_{a(t),i}(\xi) L_{a(t),j}(\xi) d\xi = \delta_{ij}$$

Decrease polynomial order

- Analytical integration

$$\forall x,\ g(x,t) \approx \sum_{l=1}^{N} \sum_{i=0}^{k} c_l^i(t)\, \phi_i\left(\xi_l(x)\right) \left[\theta\left(x - x_{l-1/2}\right) - \theta\left(x - x_{l+1/2}\right)\right]$$

$$F(x) = \sum_{l'=1}^{N} \sum_{i'=0}^{k} \sum_{l=1}^{N} \sum_{i=0}^{k} c_{l'}^{i'}(t)\, c_l^i(t)$$

$$\left[\theta\left(x - x_{l'-1/2}\right) \int_{x_{l'-1/2}}^{x} \phi_{i'}\left(\xi_{l'}(u)\right) \int_{x_{l-1/2}+(x-u-x_{l-1/2})\theta(x-u-x_{l-1/2})}^{x_{l+1/2}+(x-u-x_{l+1/2})\theta(x-u-x_{l+1/2})} \frac{\phi_i\left(\xi_l(v)\right)}{v} \mathrm{d}v\mathrm{d}u - \theta\left(x - x_{l'+1/2}\right) \int_{x_{l'+1/2}}^{x} \phi_{i'}\left(\xi_{l'}(u)\right) \int_{x_{l-1/2}+(x-u-x_{l-1/2})\theta(x-u-x_{l-1/2})}^{x_{l+1/2}+(x-u-x_{l+1/2})\theta(x-u-x_{l+1/2})} \frac{\phi_i\left(\xi_l(v)\right)}{v} \mathrm{d}v\mathrm{d}u\right]$$

$$\frac{\mathrm{d}c^k}{\mathrm{d}t} = c^i W_{ij}^k c^j$$

- General Lokta-Volterra system
- Differential Riccati equation

Evaluate g in few number of points.

Interface with PHANTOM

- Interface with PHANTOM (D. Mentiplay et D. Price)

```
Files in libsmol.tar.gz:

  - libsmol.a
  - smol2other.f90
  - README

To use the library compile smol2other.f90 using the same
compiler that you use
to compile your hydro code (e.g. using gfortran):

    gfortran -c smol2other.f90

Then, when compiling your program, include the directory with
smol2other.mod.
For example

    gfortran -c -I$(SMOL_DIR) ...

Then, when linking your program, link to libsmol.a and include
the directory.
For example

    gfortran -o your_program *.o -L$(SMOL_DIR) -lsmol
```

```f90
!-----------------------------------------
! MODULE: interface with other code
!-----------------------------------------
module smol2other

   implicit none

   interface
      module subroutine grain_growth(ndusttypes,dustfrac,grainmass,graindens,dt,mtot,rhogas,temp)
         implicit none
         integer,                                intent(in)    :: ndusttypes
         double precision, dimension(ndusttypes), intent(in)    :: grainmass,graindens
         double precision,                       intent(in)    :: dt,mtot,rhogas,temp
         double precision, dimension(ndusttypes), intent(inout) :: dustfrac
      end subroutine grain_growth
   end interface


end module smol2other
```

Perspectives

- Mathematical optimisations: adapted polynomial basis, binning choice.
- Implicit form for time solver.
- Fragmentation $$\partial_t g(x,t) = -\partial_x \left(F_{coag}[g](x,t) + F_{frag}[g](x,t) \right)$$
- Simulations with PHANTOM

Thanks for listening !