Magritte

A new Multidimensional Accelerated General-purpose Radiative Transfer code

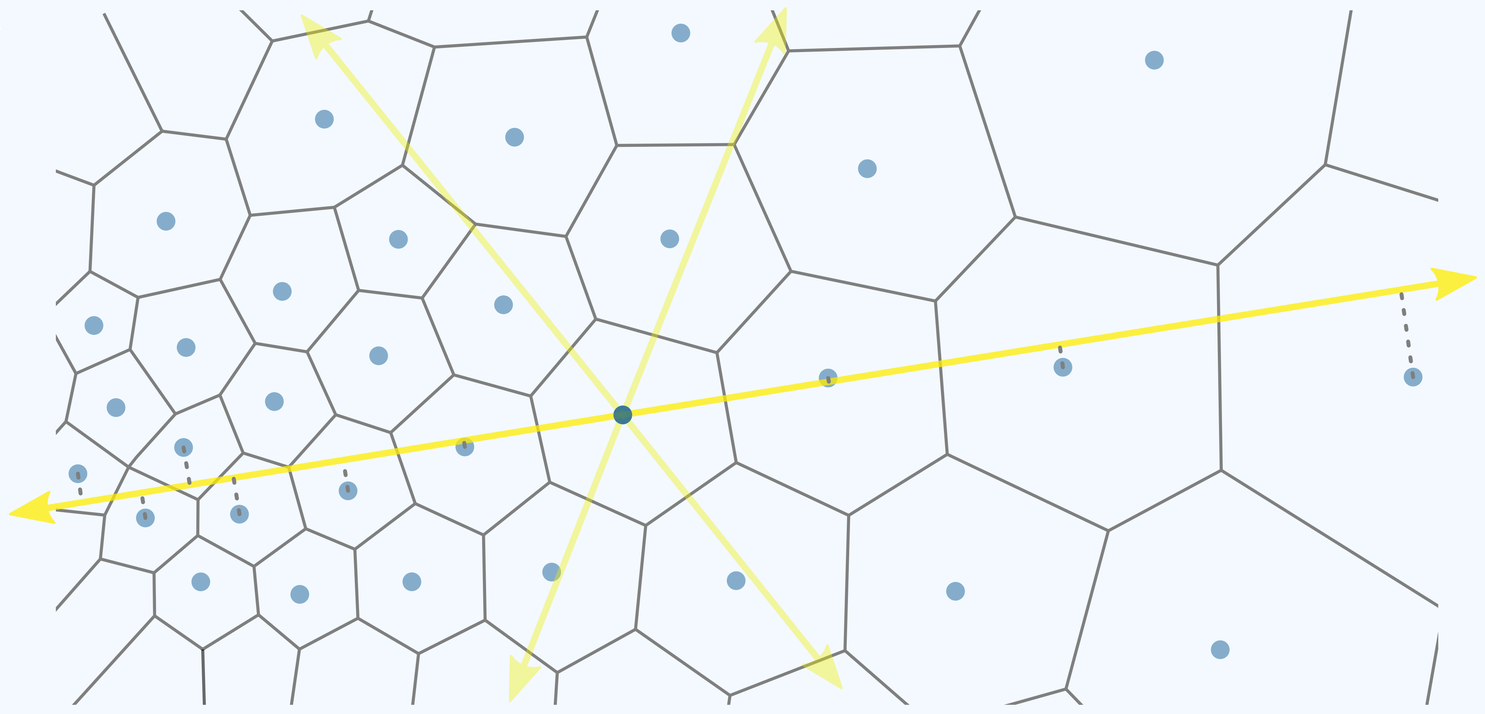
F. De Ceuster [1,2], J. Yates [1], P. Boyle [3], L. Decin [2,4] and J. Hetherington [5]

IAU General Assembly XXX, Vienna 2018

Magritte is a new deterministic radiative transfer code. It is a ray-tracing code that computes the radiation field by solving the radiative transfer equation along a fixed set of rays for each grid cell. Its ray-tracing algorithm is independent of the type of input grid and thus can handle smoothed-particle hydrodynamics (SPH) particles, structured as well as unstructered grids. The radiative transfer solver is highly parallelized and optimized to have well scaling performance on several computer architectures. Magritte also contains separate dedicated modules for chemistry and thermal balance. These enable it to self-consistently model the interdependence between the radiation field and the local thermal and chemical states. The source code for Magritte will be made publically available at github.com/Magritte-code.

Solving the radiative transfer problem

Radiative transfer plays a key role in the dynamics, the chemistry and the energy balance of various astrophysical objects. Therefore it is essential in astrophysical modelling to properly take into account all radiative processes and their interdependence. The ever growing size and complexity of these models requires fast and scalable methods to compute the radiation field. Magritte is a new general-purpose radiative transfer solver written in modern C++. In contrast to popular (probabilistic) Monte Carlo codes, Magritte is a deterministic ray-tracer which computes the radiation field by solving the transfer equation along a fixed set of rays originating from each grid cell. Being a deterministic code allows for various optimizations and facilitates parallelization. Magritte's algorithm only uses the locations of the cell centers and the nearest neighbor lists to trace the rays. Hence it can cope with SPH particle data as well as with structured or unstructured grids. Contributions from both lines and continua are taken into account in the local emissivities and opacities. Scattering is taken into account iteratively, adding an extra source and opacity. Our method can cope with the most general anisotropic scattering formalisms.



Transfer equation along a ray

The transfer equation denotes the change of the intensity $I_\nu(\hat{\mathrm{n}})$ with distance z along ray $\hat{\mathrm{n}}$.

Line + Continuum emissivity and opacity — Scattering opacity and redistribution function

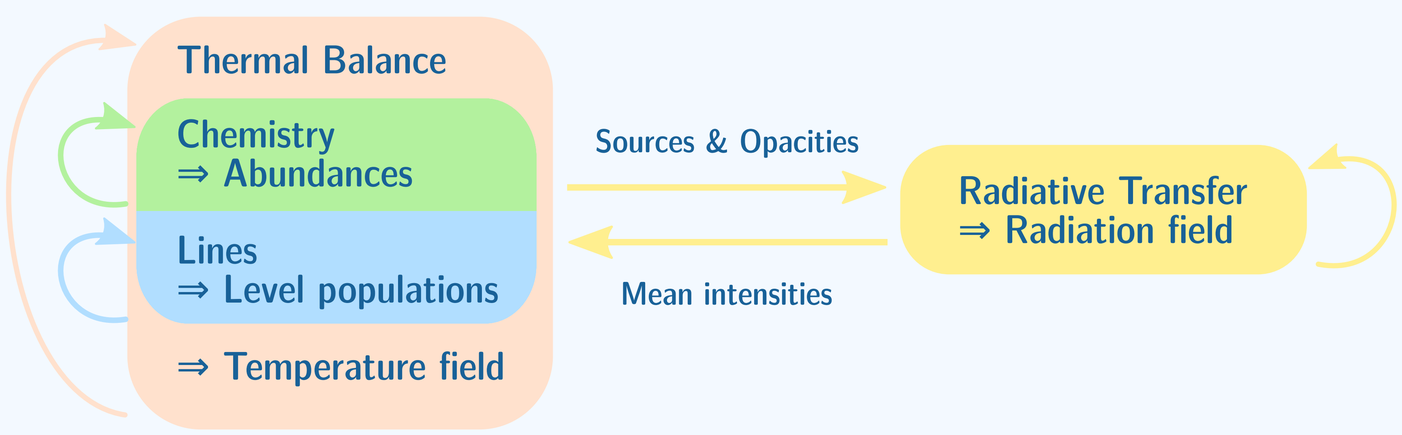
$$\frac{\mathrm{d}I_\nu(\hat{\mathrm{n}})}{\mathrm{d}z} = \eta_\nu - \left(\chi_\nu + \chi_\nu^{\mathrm{sca}}(\hat{\mathrm{n}})\right) I_\nu(\hat{\mathrm{n}}) + \int_0^\infty \mathrm{d}\nu' \oint \mathrm{d}\Omega'\, R_{\nu\nu'}(\hat{\mathrm{n}}, \hat{\mathrm{n}}')\, I_{\nu'}(\hat{\mathrm{n}}')$$

Solved directly for $I_\nu(\hat{\mathrm{n}})$ — Evaluated using previous iteration

Magritte solves this in a numerically more stable version of the second-order Feautrier form [1].

Magritte's iterative scheme

Input: Initial temperature field, elemental abundances and LTE populations



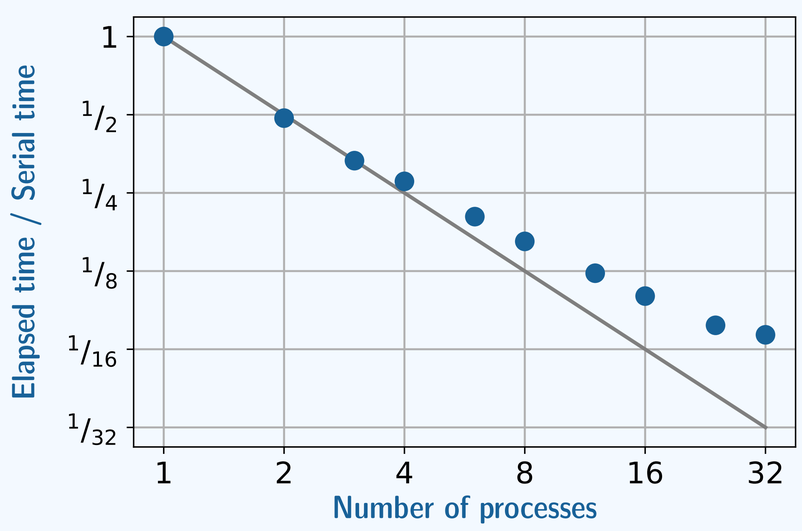
Output: Self-consistent temperature field, chemical abundances, level populations and radiation field

Note: Magritte's modular design allows for easy integration with other codes (e.g. to replace or extend the Chemistry or Thermal Balance module).

Parallelizing a deterministic ray-tracer

There are three common parallel programming paradigms: message passing, threading, and single instruction multiple data (SIMD) vectorization. Magritte uses a combination of all three to ensure performance on both shared and distributed memory architectures. The computations for different rays in our algorithm are independent within an iteration. Therefore these can easily be distributed over different processes and the results communicated at the end of each iteration. This is done using the standard message passing interface (MPI). Solving the transfer equation along a certain ray requires data from different grid cells and frequencies and thus can better be kept as local as possible in memory. Therefore, within each process, the computations for different cells are threaded using the OpenMP standard. The computations along a certain ray for different frequencies require exactly the same operations but with different values for the emissivities and opacities for each frequency. Hence these computations are ideally suited for SIMD vectorization. To achieve this in a portable way, Magritte uses the SIMD vector types provided in the Grid library [2,3]. In future versions we will also explore the possibility of offloading the whole radiative transfer solver to a graphics processing unit (GPU).

(Preliminary) strong scaling



The figure above shows the (preliminary) strong scaling behaviour of the MPI distributed parallelization over the rays for a test model containing 192 rays, 220 frequency bins and 12,133 grid cells.

Acknowledgements

FDC is supported by the EPSRC iCASE studentship programme, Intel Corporation and Cray Inc. LD acknowledges support from the ERC consolidator grant 646758 AEROSOL. This work was performed using the Cambridge Service for Data Driven Discovery (CSD3), part of which is operated by the University of Cambridge Research Computing on behalf of the STFC DiRAC HPC Facility (www.dirac.ac.uk). The DiRAC component of CSD3 was funded by BEIS capital funding via STFC capital grants ST/P002307/1 and ST/R002452/1 and STFC operations grant ST/R00689X/1. DiRAC is part of the National e-Infrastructure.

(1) Department of Physics and Astronomy, University College London, UK
(2) Department of Physics and Astronomy, Institute of Astronomy, KU Leuven, Belgium
(3) School of Physics and Astronomy, The University of Edinburgh, UK
(4) School of Chemistry, University of Leeds, UK
(5) The Alan Turing Institute, UK

References:
[1] Rybicki & Hummer (1991)
[2] Boyle et al. (2015)
[3] github.com/paboyle/Grid

frederik.deceuster@kuleuven.be
@FredDeCeuster