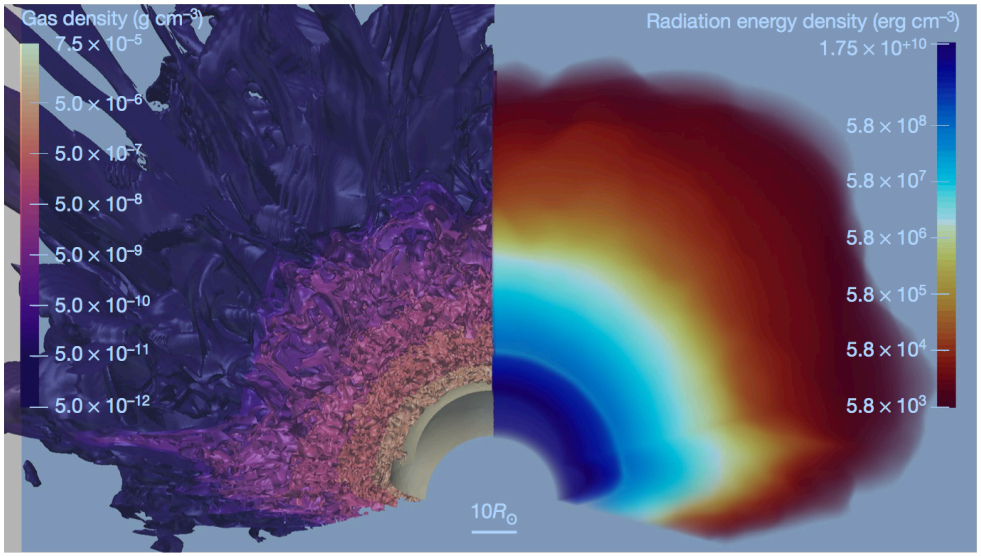
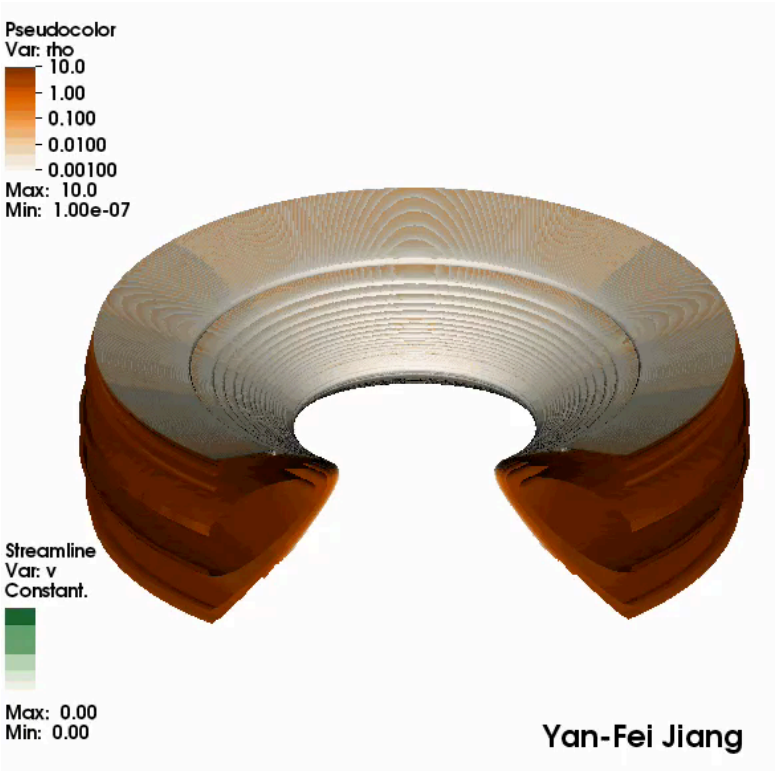
Radiation Transfer Status

- Time dependent radiation transport implemented in Athena++, roughly as described in Jiang+ 2014 for Athena
- Transfer piece solved explicitly; source terms handled (locally) implicitly
- Includes Lorentz transformations between comoving/Eulerian frames so covariant rather than second order in v/c
- Handles multiple frequency bands consistently with implicit update of source terms (temperature)
- Implements reduced speed of light approximation

Radiation Transfer Status





Thus far used to simulate accretion disks, massive stars, winds, collapse of stellar cores, …

Overview of Scheme

Transfer equation:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \nabla \cdot (\hat{n} I) = S_\nu(I_\nu)$$

Broken into two pieces. First transport is handled explicitly ([reduced] speed of light sets Courant condition):

$$\frac{\partial I}{\partial t} + \tilde{c}\nabla \cdot (\hat{n} I) = 0$$

Source term is handled separately via operator splitting. Update is implicit, solved simultaneously with gas temperature:

$$\frac{\partial I_c}{\partial t} = \Gamma(\hat{n})\tilde{c}\rho \left[\kappa_P \frac{aT^4}{4\pi} + \kappa_s J_c - (\kappa_J + \kappa_s) I_c \right]$$

$$\frac{\rho k_B}{\mu(\gamma - 1) m_p}\frac{\partial T}{\partial t} = -\tilde{c}\rho \left(\kappa_P a T^4 - 4\pi \kappa_J J_c \right)$$

Note that we solve the Eulerian frame source term update using comoving frame variables e.g. $I_c = \Gamma^4 I$

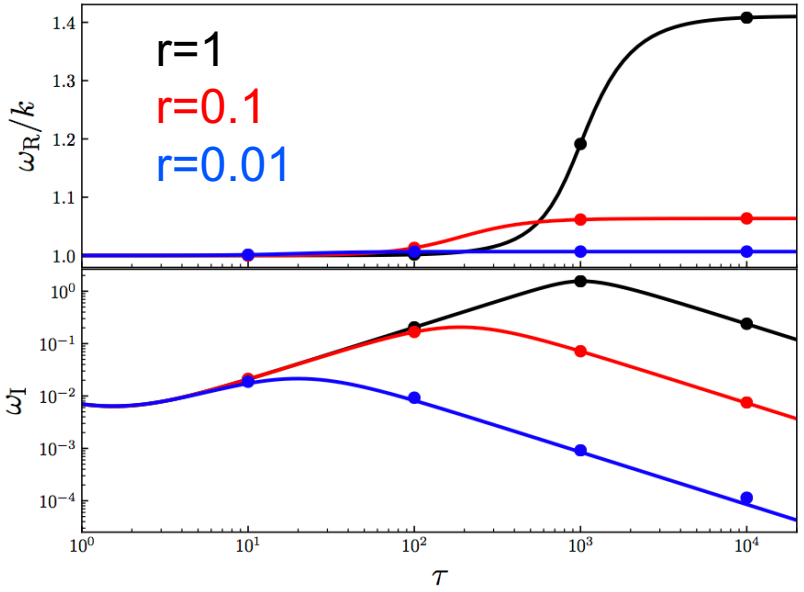
Issues with current method

- Reduced speed of light approximation is of limited utility when optical depths are large

$$t_{\rm dif} \sim \frac{\lambda_{mfp}}{c}\tau^2 \sim \frac{L}{c}\tau$$

$$t_{\rm adv} \sim \frac{L}{v}$$

- Flat spacetime only – no GR
- Only handles spherical polar coordinates in 3D
- Reliance on grey opacities
- Only radiation field in neighboring zones is used to compute intensity – grid sets preferred direction, impacting different angles differently



$$S_c = c\rho\left[\kappa_R\left(\frac{aT^4}{4\pi} - I_c\right) + \kappa_s(J_c - I_c) + (\kappa_P - \kappa_R)\left(\frac{aT^4}{4\pi} - J_c\right)\right]$$

$$\int S_c\, d\Omega = c\kappa_P\rho\left(\frac{aT^4}{4\pi} - J_c\right)$$

$$\int \hat{n} S_c\, d\Omega = c(\kappa_R + \kappa_s)\rho F_c$$

Generalizing to non-Cartesian Coordinates

The general covariant transfer equation is given by the following:

$$\frac{\partial(n^t n_t I_\nu)}{\partial t} + \frac{1}{\sqrt{-g}}\frac{\partial}{\partial x^i}\left(\sqrt{-g} n^i n_t I_\nu\right) + \frac{\partial}{\partial \nu}\left(\mathcal{F}_\nu \nu n_t I_\nu\right) - \frac{1}{\sin\zeta}\frac{\partial}{\partial \zeta}\left(\mathcal{F}_\zeta n_t I_\nu\right) + \frac{\partial}{\partial \psi}\left(\mathcal{F}_\psi n_t I_\nu\right)$$
$$= n_t\left(j_\nu - \alpha_\nu I_\nu\right).$$

$$\omega^{(a)}_{(b)(c)} = e^{(a)}_{\alpha} e^{\gamma}_{(c)} e^{\alpha}_{(b);\gamma}$$

Momentum coordinates
ν: frequency
ζ: polar angle
ψ: azimuthal angle

$$\mathcal{F}_\nu = -n^{(a)} n^{(b)} \omega^{(0)}_{(a)(b)},$$

$$\mathcal{F}_\zeta = -n^{(a)} n^{(b)} \left(\omega^{(3)}_{(a)(b)} - \omega^{(0)}_{(a)(b)} n^{(3)}\right),$$

$$\mathcal{F}_\psi = n^{(a)} n^{(b)} \frac{\left(n^{(2)} \omega^{(1)}_{(a)(b)} - n^{(1)} \omega^{(2)}_{(a)(b)}\right)}{\sin^2\zeta}$$

- This requires a choice of coordinates and basis (tetrad) $e^{(a)}{}_{\alpha}$. Necessary for general relativistic spacetimes but also relevant for curvalinear coordinates in flat space time.
- Basis for method that treats angles/frequency (momentum coordinates) on equal footing with space coordinates.

Example: Spherical-polar

Current version defines angles relative to fixed Cartesian axis. For this basis, all of the angle terms are zero.

$$\frac{1}{c}\frac{\partial I_\nu}{\partial t}+\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 n_r I_\nu\right)+\frac{1}{r\sin\theta}\frac{\partial(\sin\theta\, n_\theta I_\nu)}{\partial\theta} + \frac{1}{r\sin\theta}\frac{\partial(n_\phi I_\nu)}{\partial\phi}=j_\nu-\alpha_\nu I_\nu$$

However, this only works in 3D. For 1D or 2D volumes, one needs a basis that is well defined over the entire cell surface. Choosing a basis aligned with the unit vectors of the standard r, θ, ϕ:

$$e^t_{(t)}=\frac{1}{c},\ e^r_{(r)}=1,\ e^\theta_{(\theta)}=\frac{1}{r},\ e^\phi_{(\phi)}=\frac{1}{r\sin\theta}$$

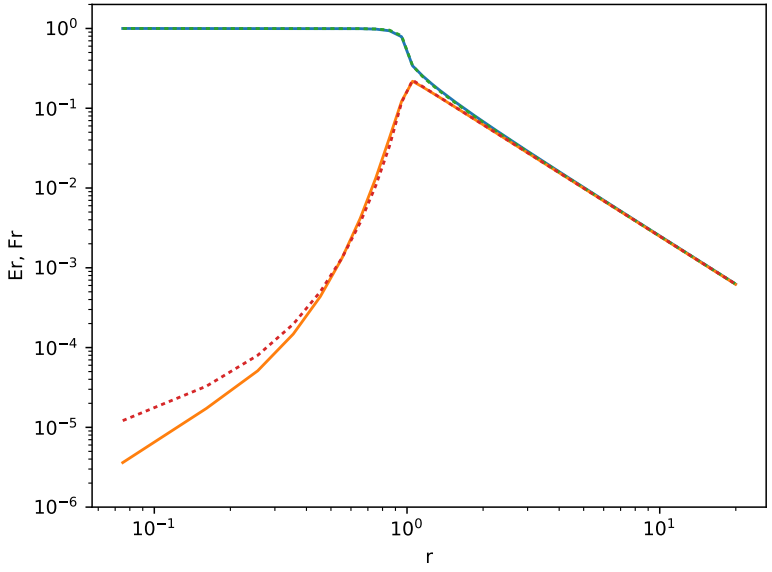
This yields:

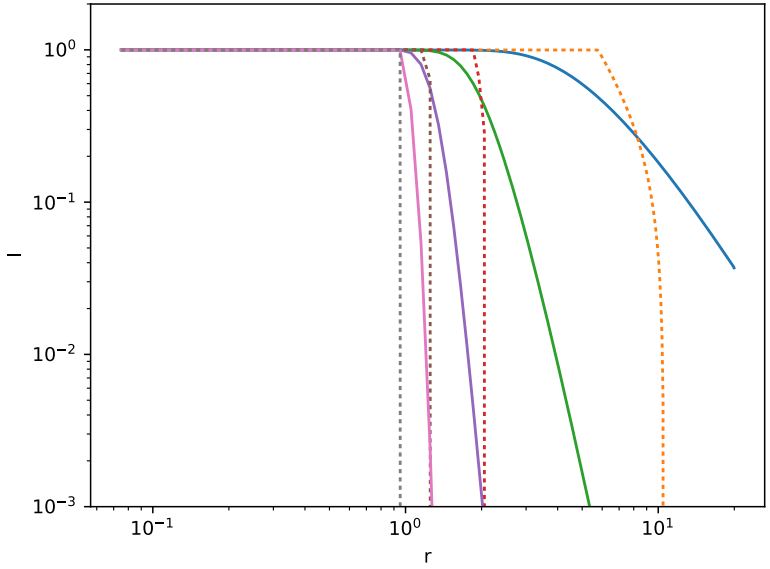
$$\frac{1}{c}\frac{\partial I_\nu}{\partial t}+\frac{1}{r^2}\frac{\partial(r^2\cos\zeta I_\nu)}{\partial r}+\frac{1}{r\sin\theta}\frac{\partial(\sin\theta\,\sin\zeta\cos\psi I_\nu)}{\partial\theta}+\frac{1}{r\sin\theta}\frac{\partial(\sin\zeta\sin\psi I_\nu)}{\partial\phi} - \frac{1}{r\sin\zeta}\frac{\partial(\sin^2\zeta\, I_\nu)}{\partial\zeta}-\frac{\sin\zeta\cos\theta}{r\sin\theta}\frac{\partial(\sin\psi\, I_\nu)}{\partial\psi}=j_\nu-\alpha_\nu I_\nu,$$

Example: Spherical-polar 1D

Method is implemented in Athena++ only in 1D (so far). Example problem with analytic solution is the radiation field interior/exterior to a homogeneous spherical emitter.

Method reproduces radiation energy density and flux but flux in different angle bins is not the step function in radius that occurs in analytic problem





Plans for the Future

- Add current version to main branch
  - needs some “tidying”
  - update of boundary condition methods
- Implement treatment for general coordinate systems
  - Needs general implementation for arbitrary coordinates?
  - Fluxes/areas for angles/frequencies computed in coordinates class
- Fully general relativistic treatment
- General covariant treatment may be equivalent to what is needed for curvalinear coordinates in flat spacetime
- Implementation of VET method
  - Should allow treatment of problems in cases where reduced speed of light fails
  - Athena implementation required two global computations: backward-Euler solution for radiation moments and short characteristics to compute Eddington tensor
  - Compute these using the multigrid infrastructure developed for self-gravity?
- General long characteristics? Higher order “upwind” schemes for above methods?

Monte Carlo in Athena++

Why bother? Several alternatives for post-processing:

- Black holes: GRMONTY (public), Pandurata (not public)
- Dust/low energy: RADMC, Hyperion, etc.

My motivations:

- Control over development/features
- Take advantages of Athena++ infrastructure e.g. particles, coordinates, atomic/molecular chemistry
- Provide publicly available tool
- MPI parallelization over mesh blocks for large simulations
- Extensions: time-dependent calculations, radiation hydro, ray tracing, neutrinos

Goals

- Performance – optimization, load balancing
- Flexibility – would like to do both static post-processing and time-dependent runs, but these motivate different methods for photon evolution
- Extensibility – provide user defined functions for customizing emission, absorption, scattering. Easy to expand code base.
- Compartmentalization – have minimal impact on non MC Athena++ classes
- Verification – robust set of tests

Components

- **Photon** – class implementing super photon/photon packet, each has energy, direction, polarization, position, weight, status, etc.
- **PhotonMover** – classes to move photons through grid; implemented as derived classes
- **MonteCarlo/MonteCarloBlock** – division similar to Mesh and MeshBlock to control initialization, photon transfer, communication and output
- **Outputs** – MCOutput/Spectra classes - monte carlo specific outputs
- **Physics implementations** – functions for scattering, absorption, emission, lorentz transformations between frames, acceleration of optically thick zones, etc.

Input/Output

```
<montecarlo>
mcranks    = 7
nphot      = 100000
cadence    = 1000
iseed      = 121500
emin       = 3.
emax       = 3.e4
scattering = compton
emission   = freefree
absorption = freefree
polarized = false
acceleration = true
boosts     = false
```

```
<output1>
file_type  = hdf5      # Binary data dump
dt         = 1.0       # time increment between outputs
variable   = mcmom

<output2>
file_type = spec
face      = outer_x1
ne        = 100
emin      = 3.
emax      = 3.e4
ncth      = 8
nphi      = 8
x2max     = 1.570796327
cartesian_axis = true
```

Use athinput files with standard formatting – <montecarlo> block

New outputs:

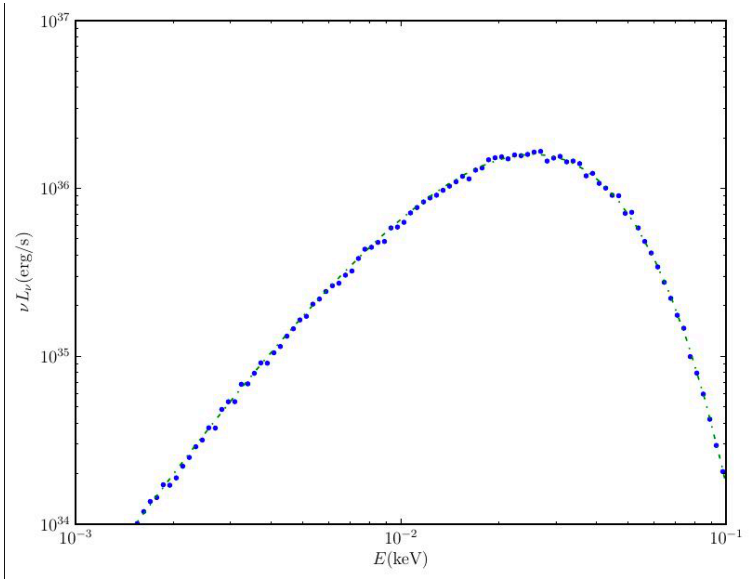
- Moments: radiation energy density, flux, pressure tensor (todo: user defined variables)
- Spectra: escaping photons binned in energy and angle

Going forward:

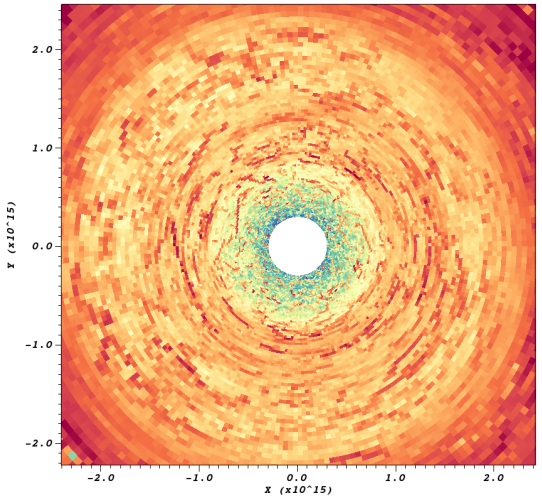
- Add images (via ray tracing) and develop python modules for visualizing outputs (e.g. generating light curves)

Examples

Spectrum from Thomson scattering +free-free absorption calculation – comparison to Feautrier solver



Radiation energy density from Athena++ rad hydro simulation snaphshot of accretion flow. 2D slice through midplane.



Physics

Scattering – relatively easy; need to randomly sample outgoing direction, energy, polarization based on incoming energy, polarization, direction; current options are isotropic, polarized/unpolarized Thomson scattering, and polarized/unpolarized Compton scattering, and user defined; function pointers set at initialization

Absorption – relatively easy; functions for computing mean free paths to absorption and scattering; current options are free-free, electron scattering, and user defined, function pointers set at initialization

Emission – more complicated; code must be flexible to handle distributed emission, point sources, external irradiation, etc. Initialization of Photons is implemented through user defined function similar to problem generator, with support for common mechanisms (e.g. free-free emission) provided

Going forward:

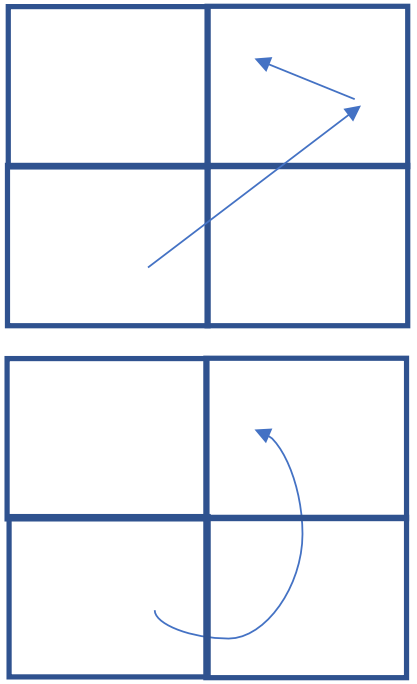
- Adding dust scattering, lyman alpha scattering, pair production, resonance scattering, etc.
- LTE and non-LTE treatments of atomic opacities

Photon Movement

Photon travels a path length equal to a number of mean-free-paths between scattering/absorption events drawn from exponential distribution

Traditional: treat movement as sequence of line segments between zones (fast for Cartesian, slower for curvalinear)

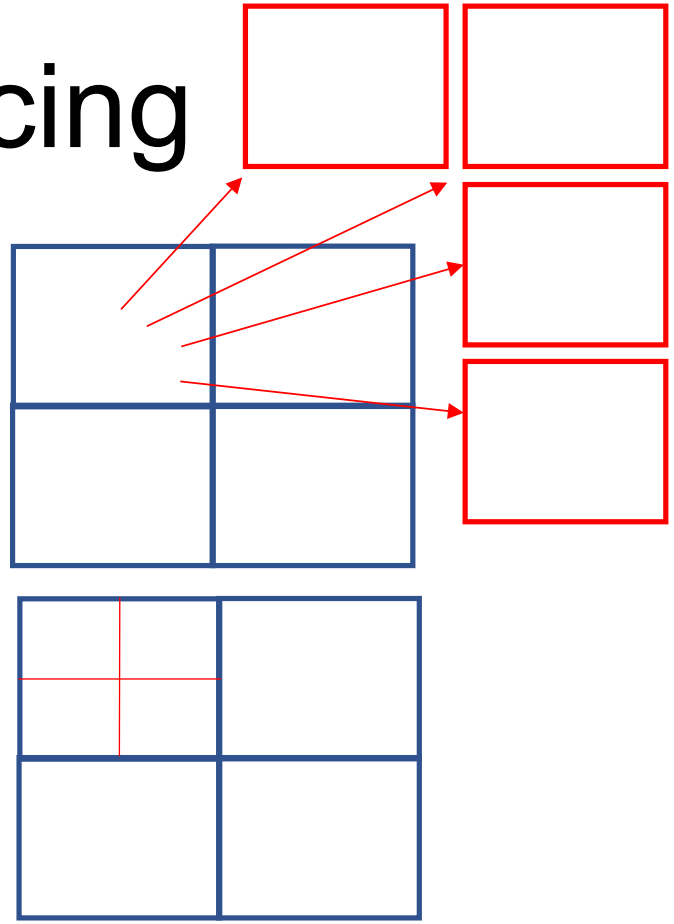
Integration: integrate photon geodesics along curved paths (Verlet algorithm form GRMONTY)



Going forward:

- Finish implementation and testing of integrator
- Can we use the same infrastructure as the particle class?
- How do we optimally implement communication between MeshBlocks? (photons may cross more than one MeshBlock per time step!)

Load Balancing



Unlike hydro, there is no reason to expect meshblocks to have same cost. Some zones (e.g. optically thick zones) could have more photons and/or longer integration

Solutions:

- Create multiple copies of same MeshBlock and run MC transfer concurrently
- Break mesh blocks up into smaller subsets for Monte Carlo

Going forward:

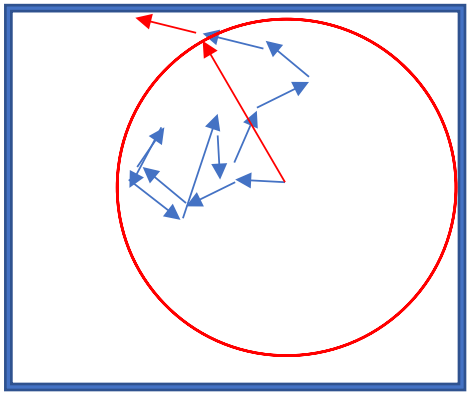
- Implement mesh refinement with cost function that includes estimate of monte carlo
- Pair this with an acceleration method

Acceleration

For large optical depths, # of scatterings scales with τ^2, but many applications with $\tau > 10^2$. Therefore computational cost dominated by zones with largest optical depth, but MC is unnecessary because diffusion approximation is good!

Common Solutions:

- Modified random walk (MRW) method
- Discrete diffusion monte carlo (DDMC) method



MRW method draws largest sphere that can fit in box and moves photon to surface with random direction. Path length is drawn from distribution of path lengths to determine absorption.

Complications/generalizations:

- Need to account for energy change associated with Compton scattering
- Need to account for advection due to flow velocities

Going forward:

- Further testing of MRW method
- Implementation of DDMC?

TODO List

Short term (end of summer 2019)

- Further development and testing of acceleration
- Full implementation of the geodesic based integrator
- MPI communication across subdomains (take advantage of particle class)
- Modification to utilize task list
- New physics: Dust and Lyman alpha scattering, molecular lines?

Midterm (end of summer 2020?)

- Time-dependent calculations
- Iterative calculations for temperature (ionization) structure
- Coupling to non-LTE calculations of rate equations?
- Discrete diffusion monte carlo?

Longterm

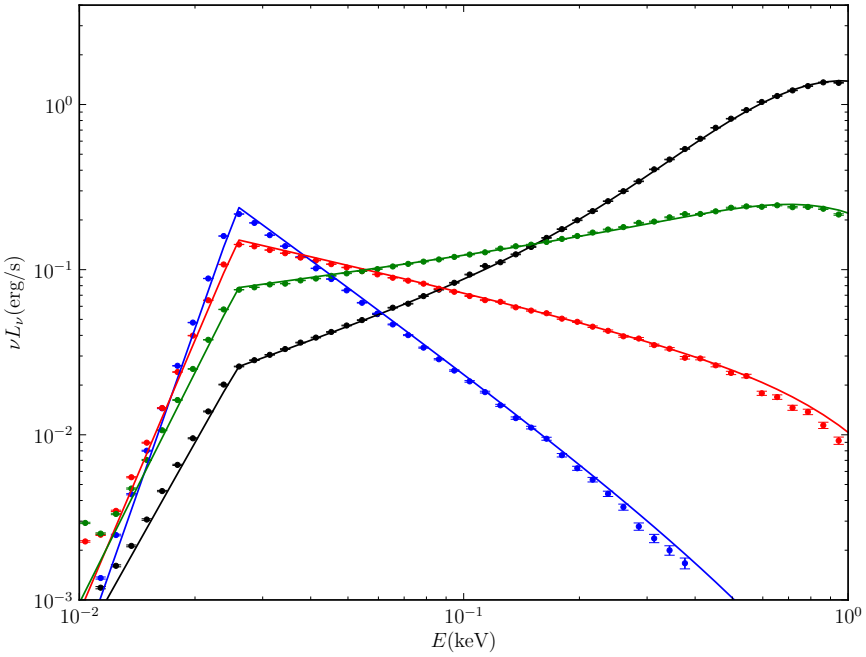
- Radiation hydro via moment method closure or implicit monte carlo
- Ray tracing module(s)

Lots of opportunities to collaborate

Acceleration

Implementing MRW with Compton scattering requires solving the Kompaneets eq. with a delta function source. Such solutions can be computed and tabulated. Photons are then draw from a distribution of final energies given a total diffusion path length (time) and initial energy.

MRW is also complicated by advection. Photons don't have infinite time to diffuse in the comoving frame befor advection in to neighboring zones with different properties. Solution: draw from a distribution of photon positions in comoving frame given a fixed advection time.



GR Tests

Verlet algorithm – integration in Schwarzschild compared to GEOKERR