MCRT: L11
MCRT on a 3D Cartesian Grid

- 3D linear cartesian grid code – optical depth integration through grid, weighting & forcing, making images, intensity moments
- Show geometry, xmax, number of cells, faces, etc
- Compute distances to faces, next face photon will hit
- Optical depth and distance traveled algorithm, show in 2D grid
- Where to add counters in optical depth routines to compute internal intensity moments (fluence, mean intensity, absorbed energy, etc)
- Subroutines of the grid code, download and run the code

The need for 3D grids

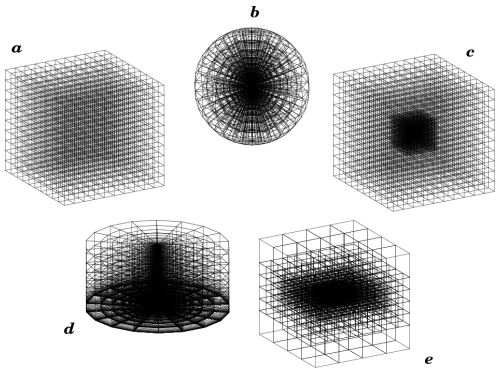
- Generate random optical depths from $\tau = -\ln \xi$, then determine the distance L to the next interaction location by solving for L in this equation:

$$\tau = \int_0^L n(s)\sigma(s)\,ds$$

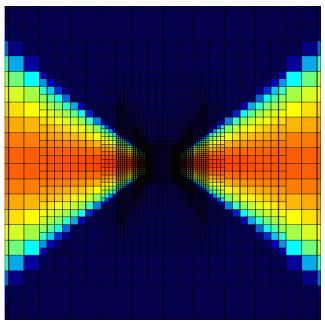
- For geometries where n and σ do not change with position, this is trivial: $L = \tau / (n\,\sigma)$
- Some structures yield analytic solutions for L, e.g., $n(r) \sim r^{-2}$
- For complex geometries cannot solve for L analytically: e.g., fractal cloud models, output densities from hydrodynamics codes, 3D skin structures, nuclear reactor geometries, etc
- Discretise the density (opacity) structure onto a 2D or 3D grid and solve for L numerically

3D Grids (Meshes)

- Evenly spaced Cartesian, spherical, cylindrical, adaptive mesh grids

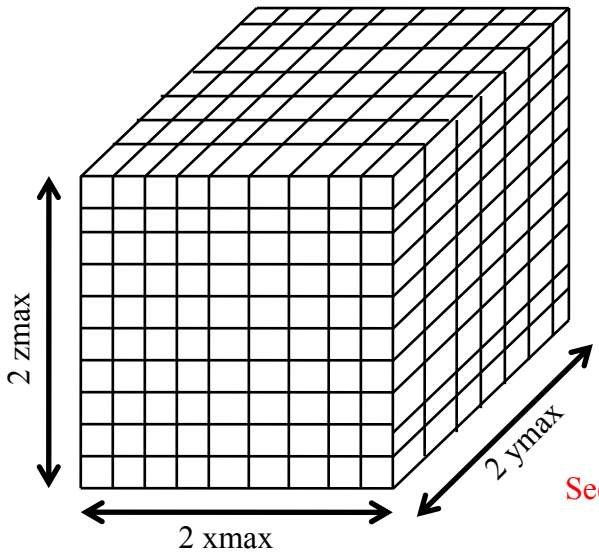


Steinacker et al (2003)



Harries et al (2004)

Linear Cartesian grid



Grid dimensions:
2xmax, 2ymax, 2zmax

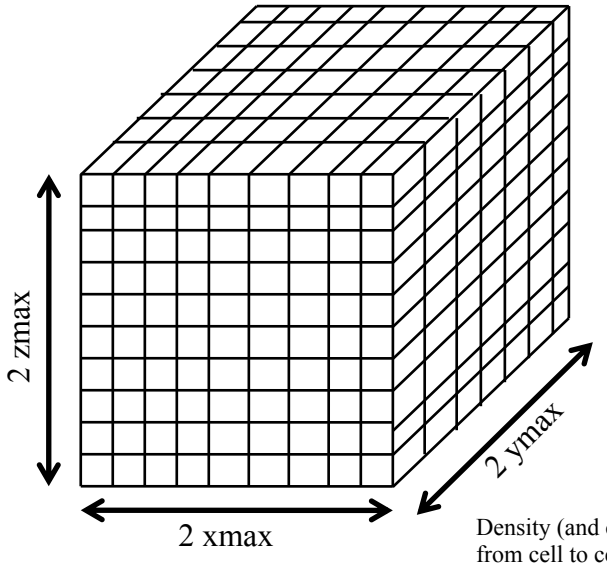
Number of x, y, z grid cells:
nxg, nyg, nzg

Number of x, y, z faces:
nxg + 1, nyg + 1, nzg + 1

See subroutione: gridset.f

Set locations of x, y, z, faces (planes):

xface(i) = 2.*xmax*(i - 1)/nxg, where i = 1, 2, …, nxg+1



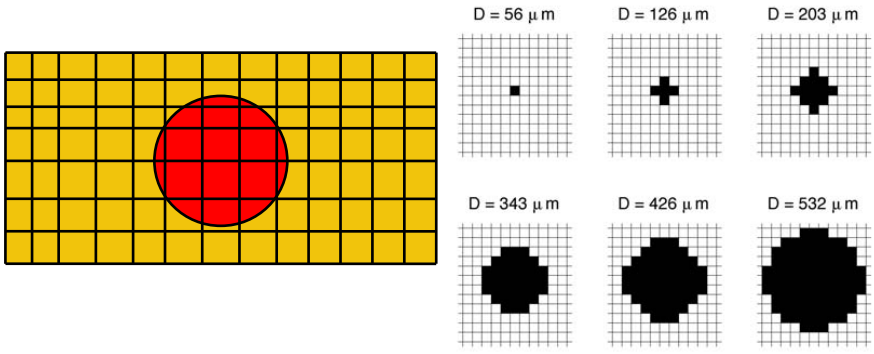
Density (and cross sections) can change from cell to cell, but are constant within a cell:

$$n = n(x, y, z),\ \sigma = \sigma(x, y, z)$$

See subroutine density.f called from gridset.f

Blood vein

- Set grid to have optical properties of skin (epidermis) but introduce a blood vein with increased absorption coefficient

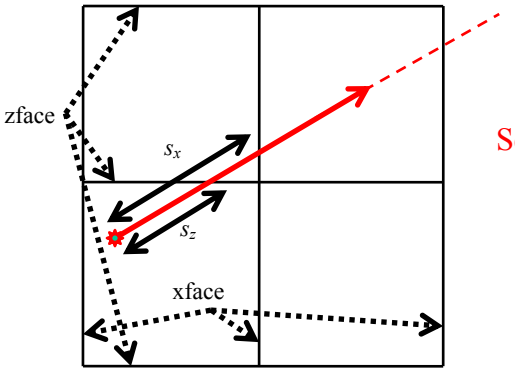


- Discretise sphere (or cylinder) onto 3D grid – Jacques (2010)

- Packet travels through grid until it reaches the random optical depth, τ_r
- Compute distances to the next *x, y, z* faces (planes) along photon line of flight:

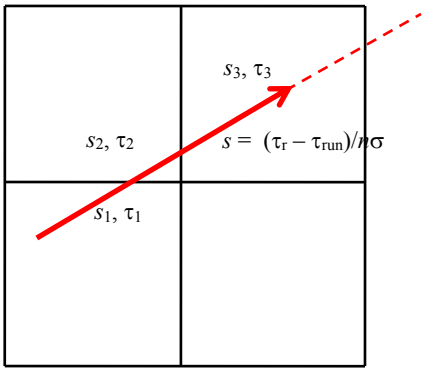
$$s_x = (\text{xface} - x)/n_x, \quad s_y = (\text{yface} - y)/n_y, \quad s_z = (\text{zface} - z)/n_z,$$

- Where s_x, s_y, s_z are distances to the next xface, yface, zface along the direction of travel, they are NOT distances parallel to *x*, *y*, *z* axes
- The next face the packet will hit is determined by: $\min(s_x, s_y, s_z)$



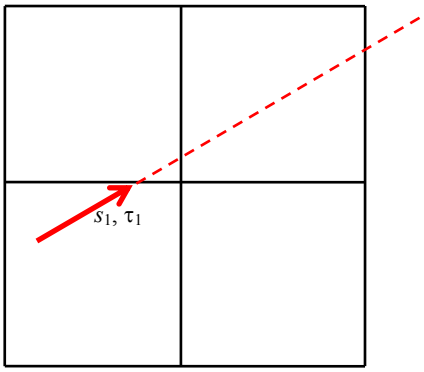
See subroutine: tauint.f

- Keep running sum of optical depth travelled: τ_{run}
- Optical depth through cell *i* to next face is $\tau_i = n_i \sigma_i s_i$
- If τ_{run} less than τ_r then move into next cell
- If τ_{run} greater than τ_r then packet interaction location will be at a distance into the cell: $s = (\tau_r - \tau_{run})/(n\sigma)$
- Scatter or absorb packet at interaction location
- Terminate packet (if absorbed), or scatter and continue random walk



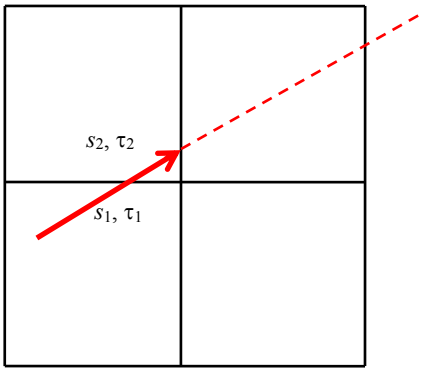
See subroutine: tauint.f

- Take step to first wall and test if $\tau_1 > \tau_r$
- No, so step into next cell (a tiny wee bit) and continue optical depth integration



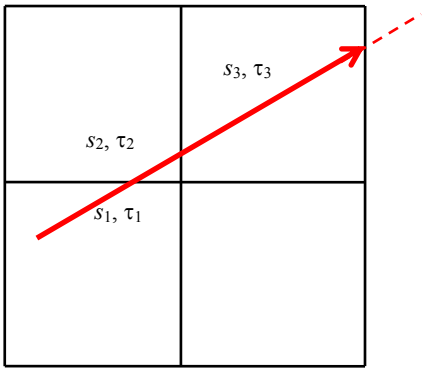
$\tau_{run} = \tau_1 < \tau_r$

- Take step to next wall and test if $\tau_{run} = \tau_1 + \tau_2 > \tau_r$
- No, so step into next cell (a tiny wee bit) and continue optical depth integration



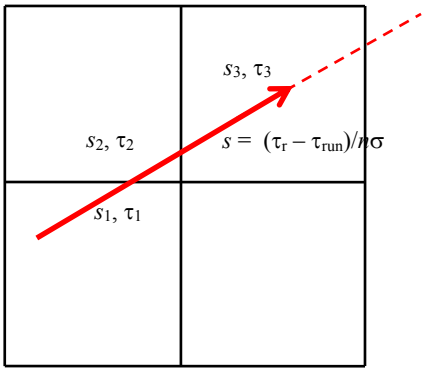
$\tau_{run} = \tau_1 + \tau_2 < \tau_r$

- Take step to next wall and test if $\tau_{run} = \tau_1 + \tau_2 + \tau_3 > \tau_r$
- Yes, so packet reaches τ_r in this cell



$\tau_{run} = \tau_1 + \tau_2 + \tau_3 > \tau_r$

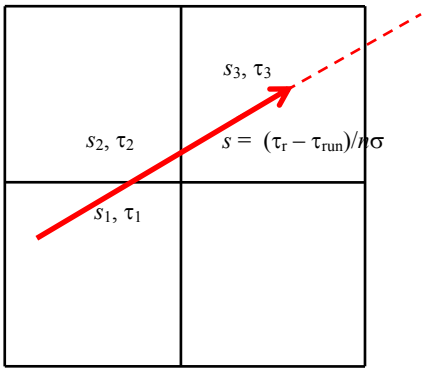
- Take step to next wall and test if $\tau_{run} = \tau_1 + \tau_2 + \tau_3 > \tau_r$
- Yes, so photon reaches τ_r in this cell
- Distance this occurs at in the cell is $s = (\tau_r - \tau_{run})/(n\sigma)$
- Update packet position using total path: $s_1 + s_2 + s$
- Scatter into new direction, absorb, "peel off," etc, etc

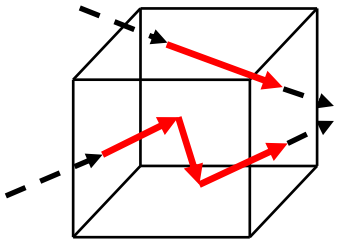


Path length estimators

- During optical depth integration through grid, keep counters in each cell for mean intensity and absorbed energy in each cell

$$J_i = \frac{L}{4\pi N \Delta V_i} \sum s \qquad E_i^{\text{abs}} = \frac{L}{N \Delta V_i} \sum n\sigma_{\text{abs}}\, s$$





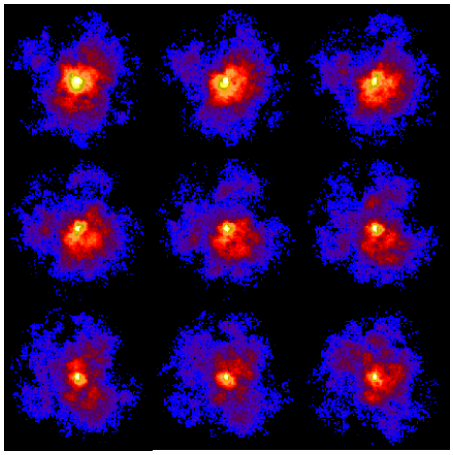
Psychoanalysis: Kenny's grid code…

- mcpolar.f – this is like your main program for the uniform density sphere. It loops over packets and tracks packets while they are in the grid. Modify this routine to make images. Test when packets exit the top of the grid – can then do internal reflection.
- Include files: grid.txt, photon.txt, etc – saves writing out same parameter declarations again and again (is he lazy or tidy?)
- iarray.f – initialize arrays to zero (is he cautious?)
- gridset.f – sets up faces, calls density.f to set density (and optical properties) in the cells
- sourceph.f – sets packet launch position and direction. Modify this routine for external illumination, collimated or gaussian beams, etc
- tauint2.f – does the optical integration through the grid and tests when packets reach the edge of the grid. Updates packet position after optical depth integration. Modify this routine to do repeating boundaries. Add in pathlength counters in this routine
- stokes.f – scatters packet, updates polarization (Stokes parameters) of packet, updates direction cosines to lab frame

Download the grid code

- From the course website download the tar file: grid.tar
- If you don't have an automatic extractor, un-tar by typing the command: tar -xvf grid.tar
- Makefile – just type "make" and the code should compile and produce the executable "mcgrid"
- Type "./mcgrid" and this will run a simulation of isotropic scattering in a uniform sphere with optical depth 10. Code writes out the density grid (unformatted output) to a file called density.dat. Screen output shows a count of the number of MC photon packets and "average number of scatterings = 57.17"

- Set up for isotropic scattering in a (discretised) sphere – compare results for average number of scatterings with your own uniform density sphere code
- Subroutine stokes.f for non-isotropic scattering: change hgg in the input.params and see how this effects average number of scatterings
- Include files – can change grid resolution in grid.txt and add in extra 3D arrays for setting scattering properties in each cell and arrays for storing counters for fluence (mean intensity)
- If you change an include file you must delete all the ".o" files before typing "make"
- Modify the code to include path length counters, external illumination, bin photons according to the (x,y) location that they leave the top of the grid, etc
- Can include fresnel reflection at upper boundary, introduce a blood vessel (need 3D arrays for the optical properties which could change from cell to cell), make images of exiting packets
- Assume detector can catch all packets no matter their exit direction



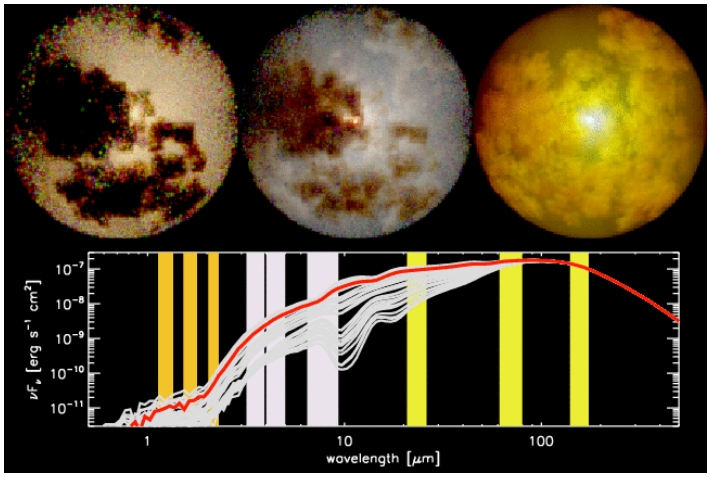
Monte Carlo scattered light
simulations of fractal clouds

NGC 7023
Reflection Nebula

3D density: viewing angle effects

Mathis, Whitney, & Wood (2002)

Heating dusty nebulae



Indebetouw, Whitney, Johnson, & Wood (2006)